

WHITEPAPER

MSAP.ai

AI Agent 스킬 설계

업무 전문가가 만들고 작은 모델이 일하는 MSAP.ai AAS

업무 전문가가 말로 정의한 반복 업무를 프론티어 AI가 결정적 코드 중심의 스킬로 만들고, 조직 안의 작은 모델이 그 스킬을 같은 결과로 실행하게 하는 설계 방법을 정리했습니다. 의사결정권자와 IT 담당자가 지시문 해석·결정적 코드·질문 예시를 기준으로 첫 파일럿 업무를 고르고 MSAP.ai AAS로 운영까지 잇는 순서를 제시합니다.

2026-11-12 · 작성자: SMART CF Team · msap.ai

목차

AI Agent 스킬 설계 — 업무 전문가가 말로 만들고 작은 모델이 일하는 MSAP.ai AAS

1장: AI Agent의 본질은 업무를 말로 옮기는 것

1.1 GPU 증설보다 업무 정의가 먼저인 AI Agent 도입 예산의 병목

1.2 정해진 코드 경로의 워크플로와 스스로 경로를 정하는 에이전트의 경계

1.3 소수의 전문 작업을 반복하는 에이전트 호출과 작은 언어 모델의 자리

1.4 만드는 쪽은 프론티어 AI, 일하는 쪽은 로컬 모델인 이종 구성과 MSAP.ai AAS의 위치

2장: 업무 전문가가 말로 정의하는 업무와 그 산출물

2.1 코드 작성은 프론티어 AI, 판단 기준은 업무 전문가 — 코딩 역량을 전제하지 않는 역할 경계

2.2 에이전트가 코드로 옮길 수 있는 업무 문장의 조건 — 입력값, 처리 규칙, 완료 상태

2.3 지시문 해석과 결정적 코드, 질문 예시를 한 장에 담는 업무 정의서 양식

2.4 업무 정의서 한 장에서 나오는 산출물 — 규칙 파일, 결정적 스크립트, 회귀 시험, 짧은 SKILL.md

3장: 만드는 곳과 일하는 곳의 분리

3.1 인터넷이 되는 개발 환경의 프론티어 AI와 폐쇄망 운영 환경의 로컬 LLM 역할 구분

3.2 비식별 데이터베이스와 목업 인터페이스로 내부 시스템을 흉내 낸 디지털 트윈 개발 환경

3.3 스킬 업로드와 스킬 허브를 통한 폐쇄망 반입 경로

3.4 고객·금액 자료를 밖으로 내보내지 않는 사내 게이트웨이와 보안 담당자의 확인 지점

4장: 프론티어 AI를 MCP로 연결해 스킬을 만드는 절차

4.1 MCP 호스트인 프론티어 AI와 MCP 서버인 MSAP.ai AAS의 연결 구조

4.2 업무 정의서를 건넨 뒤 스킬을 만들어 올리고 켜기까지의 작업 순서

4.3 MSAP.ai AAS 채팅으로 운영 모델에 실제 요청을 보내는 시험 — 만든 모델이 아닌 실행 모델 기준의 판정

4.4 실패 유형별 수정 반복과 제작 단계의 사용자 동의, 비밀값 분리 보관

5장: 작은 모델이 일하게 만드는 스킬 작성 지시 세 가지

5.1 지시 ① 지시문 해석 — 요청 첫 줄 기준의 명령 선택과 불명확한 요청의 정해진 질문 후 정지

5.2 지시 ② 결정적 코드 — 값 추출부터 재조회 검증까지 스크립트로 옮기는 범위

5.3 지시 ③ 질문 예시 — 질문 패턴과 업무 시나리오를 회귀 시험으로 고정하는 예시 설계

5.4 「결정적 코드로」 한 줄 지시의 한계 — 확인 게이트가 모델 몫으로 남은 대조 결과

6장: 보급형 GPU에서 에이전트가 쓸 만해지는 설계

6.1 토큰 생성 속도를 정하는 메모리 대역폭과 활성 파라미터가 작은 MoE 모델의 이점

6.2 활성 파라미터 3B 급 오픈소스 MoE 모델로 바꾼 보급형 GPU의 응답 속도 변화

6.3 판단은 코드, 모델은 한 줄 실행 — 코드와 로컬 LLM, 판단 모델, 사람의 역할 분담

6.4 코드 이관 뒤에도 남는 공백 세 곳과 모델 호출 전·도구 호출 전·출력 직후 훅의 차단

7장: 업무 전문가의 스킬 작성 예시

7.1 CRM 자연어 조회 스킬 — 오답 394초에서 정답 14초로 바뀐 업무 정의와 결정적 코드 이관

7.2 영업기회 등록 스킬 — SKILL.md 281줄에서 52줄로, 턴당 도구 호출 1회로 줄인 진행기

7.3 견적 초안 검증 — 첫 초안 단계부터 실행되는 판단 모델의 요청 충족 판정

7.4 서버 업그레이드와 보안 공지를 반영하는 운영 에이전트 설계안 — 영업 밖 IT 운영 업무로의 확장
8장: 만든 스킬이 쓸 만한지 재는 기준과 운영
8.1 실제 요청 기준의 합격선 — 도구 호출 3회 이하, 수 분 이내 응답, 정답
8.2 코드 실행과 판단 모델 판정, 재시도 1회 또는 사람 이관으로 이어지는 결과 검증 순환
8.3 질문 예시 회귀 시험으로 관리하는 스킬 변경 — 모델 교체와 업무 규칙 변경 때의 재확인
8.4 업무가 바뀌면 업무 정의서부터 고치는 스킬의 생애 주기와 업무 전문가의 소유
9장: 남은 한계와 도입 순서
9.1 작은 모델에 아직 어려운 영역 — 모호한 요청, 범용 대화, 코드로 옮기지 못한 판단
9.2 운영 중 에이전트의 스킬 자가 수정 위험과 쓰기 승인 대기 설정
9.3 파일럿 대상 업무의 선정 기준 — 반복 빈도, 규칙 명시 가능성, 데이터 반출 제한
9.4 부서 반복 업무 하나로 시작하는 MSAP.ai AAS 파일럿 — 업무 정의서 작성에서 부서 확산까지
부록 — 인용한 공개 자료 10건과 본문 용어 정의
Appendix A — 참고문헌 (References)
Appendix B — 용어 정의 (Glossary)

AI Agent 스킬 설계 — 업무 전문가가 말로 만들고 작은 모델이 일하는 MSAP.ai AAS

이 백서는 AI Agent 도입을 검토하는 의사결정권자와 IT 담당자를 위해, 업무 전문가가 말로 정의한 업무를 프론티어 AI가 결정적 코드 중심의 스킬로 만들고 조직 안의 작은 모델이 그 스킬을 같은 결과로 실행하게 하는 방법을 설명합니다. 1–3장은 AI Agent 도입의 출발점이 모델 크기나 GPU가 아니라 업무 정의라는 관점, 업무 전문가가 쓰는 업무 정의서와 그 산출물, 스킬을 만드는 개발 환경과 스킬이 일하는 폐쇄망 운영 환경의 분리를 다룹니다. 4–6장은 프론티어 AI를 MCP로 MSAP.ai AAS에 연결해 스킬을 만드는 절차, 작은 모델이 같은 결과를 내게 하는 지시문 해석·결정적 코드·질문 예시의 세 가지 지시, 보급형 GPU에서 쓸 만한 속도를 내는 설계를 정리합니다. 7–9장은 영업 업무의 스킬 작성 예시, 만든 스킬의 합격선과 운영 방법, 남은 한계와 도입 순서를 거쳐, 부서 반복 업무 하나를 골라 업무 정의서를 쓰고 MSAP.ai AAS 파일럿 대상으로 정하는 결정으로 이어집니다.

1장: AI Agent의 본질은 업무를 말로 옮기는 것

AI Agent(AI 에이전트)는 언어 모델이 상황에 맞춰 도구를 고르고 호출하면서 업무를 끝까지 처리하는 소프트웨어이며, 조직이 이 도구를 도입하는 목적은 사람이 반복하던 업무를 기계가 같은 결과로 대신 처리하게 하는 데 있습니다. 실제 도입 논의는 어떤 대형 언어 모델(LLM, Large Language Model)을 쓸지, 그 모델을 올릴 GPU 서버를 몇 대 살지에서 시작하는 경우가 많습니다. 모델과 장비를 먼저 고르면 무엇을 시킬지와 어떤 결과가 정답인지가 정해지지 않은 상태에서 비용부터 발생하고, 도입 효과를 잴 기준은 장비를 설치한 뒤에야 만들게 됩니다. MSAP.ai AAS(AI Agent Studio)는 이 순서를 바꾸는 도구로, 업무 전문가가 말로 정의한 업무를 프론티어 AI가 결정적 코드 중심의 스킬로 만들고 조직 안의 작은 모델이 그 스킬을 매일 같은 결과로 실행하도록 만들기와 넘기기를 잇습니다.

여기서 업무 전문가는 코드를 쓰지 않아도 자기 업무의 판단 기준과 예외를 가장 정확히 아는 사람을 말하고, 프론티어 AI는 Claude처럼 현재 가장 높은 추론 성능을 내는 상용 대형 모델을 말합니다. 결정적 코드(deterministic code)는 같은 입력에 언제나 같은 결과를 돌려주는 프로그램이며, 스킬은 SKILL.md 지시문·스크립트·참고 파일을 한 디렉터리에 묶어 에이전트가 필요할 때 불러 쓰는 업무 단위입니다[S2]. 이 장은 도입 효과를 정하는 요인이 모델 크기보다 업무 정의에 있다는 관점을 먼저 세우고, 정해진 경로를 따르는 워크플로와 스스로 경로를 정하는 에이전트의 경계, 반복 업무에서 작은 언어 모델이 맡는 역할, 만드는 모델과 일하는 모델을 나누는 이종 구성을 차례로 설명합니다. 마지막에는 의사결정권자가 MSAP.ai AAS 도입을 검토하면서 내릴 판단 네 가지를 근거 장 번호와 함께 제시합니다.

1.1 GPU 증설보다 업무 정의가 먼저인 AI Agent 도입 예산의 병목

AI Agent 도입 예산은 모델 사용료와 GPU 서버 같은 실행 기반에 쓰는 돈과, 업무를 정의하고 스킬을 설계하는 데 쓰는 돈으로 나뉩니다. GPU 서버는 운영 환경에 반드시 필요한 투자이지만, 어떤 업무를 어떤 정답 기준으로 맡길지가 정해져야 필요한 모델 크기와 서버 규모도 계산할 수 있습니다. 따라서 도입 성과를 가르는 병목은 장비의 성능보다 업무를 기계가 실행할 형태로 옮기는 작업에 있고, 예산 배정 순서도 그 작업을 앞에 두는 편이 투자 대비 효과(ROI)를 높입니다. 두 가지 도입 순서가 같은 예산으로 어떤 결과를 내는지는 다음 표와 같습니다.

구분	모델·GPU를 먼저 정하는 도입	업무 정의를 먼저 하는 도입
첫 의사결정	쓸 모델과 GPU 수량	맡길 반복 업무와 정답 기준
모델 크기 산정 근거	공개 벤치마크 점수, 다른 조직의 구성	업무 정의서의 처리 규칙과 질문 예시
첫 성과 측정 시점	장비 설치 뒤 시범 업무를 찾은 다음	스킬을 만든 직후 운영 모델로 시험할 때
실패 원인 추적	모델 탓인지 업무 정의 탓인지 가리기 어려움	실패한 질문 예시를 회귀 시험으로 다시 실행

1.1.1 큰 모델과 고가 GPU부터 정하는 도입이 업무 성과로 이어지지 않는 구조

모델 선정이 업무 정의보다 앞설 때 생기는 추가 작업

모델을 먼저 고르는 도입은 대개 공개 벤치마크 점수와 다른 조직의 구성 사례를 근거로 삼습니다. 하지만 벤치마크 점수는 범용 문제를 얼마나 잘 푸는지를 보여 줄 뿐이고, 우리 조직의 견적 규칙이나 서버 점검 절차를 정확히 처리하는지는 알려 주지 않습니다. 장비를 설치한 다음에야 시범 업무를 찾고, 업무 담당 부서에 무엇이 정답인지 묻고, 그 답을 프롬프트에 덧붙이는 작업이 뒤따릅니다. 프롬프트가 길어질수록 요청 한 건이 쓰는 입력 토큰 소비가 늘고, 같은 질문에도 실행할 때마다 도구 순서와 답이 달라져 담당자가 결과를 다시 확인해야 합니다. 즉 모델을 먼저 정한 도입은 업무 정의라는 작업을 없애지 못하고 장비 투자 뒤로 미루며, 그동안 장비는 쓸 업무가 정해지지 않은 상태로 남습니다.

토큰 생성 속도를 정하는 GPU 메모리 대역폭과 모델 크기의 관계

GPU를 늘리는 것만으로 응답 속도 문제를 풀기 어려운 이유는 언어 모델의 추론 방식에 있습니다. 언어 모델이 답을 만드는 디코드 단계는 토큰을 하나씩 생성하는 행렬-벡터 연산이라 GPU의 계산 능력을 다 쓰지 못하고, 지연 시간은 가중치와 중간값을 메모리에서 GPU 연산 장치로 옮기는 속도가 좌우합니다[S5]. 다시 말해 같은 장비에서는 토큰마다 읽어야 하는 가중치가 많은 모델일수록 느려지고, 그 속도를 유지하려면 메모리 대역폭이 큰 고가 장비가 필요합니다. 에이전트가 요청 한 건을 처리하면서 모델을 여러 번 호출하면 호출 횟수만큼 지연과 비용이 곱해집니다. 따라서 업무 처리 방식을 정하지 않은 상태에서 장비만 늘리면 호출 횟수를 줄일 기회는 놓친 채 호출 한 번의 비용만 낮추려는 투자가 됩니다.

모델·GPU 우선 도입에서 비용이 발생하는 단계 목록

모델과 GPU를 먼저 정한 도입에서 비용이 생기는 단계를 발생 순서대로 놓으면 다음 다섯 가지입니다. 이 가운데 첫 항목을 뺀 네 항목은 업무 정의를 앞에 두면 줄이거나 없앨 수 있습니다.

- GPU 서버 구매와 설치: 업무 기준 없이 모델 크기를 정하므로 과잉 투자와 성능 부족 위험이 함께 남음
- 시범 업무 탐색: 장비 설치 뒤 담당 부서를 찾아 요구 사항을 다시 수집
- 프롬프트 보강: 정답 기준을 지시문에 덧붙이면서 요청당 입력 토큰 소비 증가
- 결과 재확인: 실행할 때마다 다른 답이 나와 담당자가 결과를 다시 검토
- 원인 추적: 오답이 모델 성능 탓인지 업무 정의 누락 탓인지 가리는 분석 작업

시범 업무 탐색부터 원인 추적까지의 네 단계는 모두 업무 정의가 늦어서 생기는 비용입니다. 반면 업무 정의를 먼저 하면 정답 기준과 예외가 처음부터 문서로 남고, 그 문서가 모델 크기 산정과 성과 측정의 근거가 되어 총소유비용(TCO) 계산에도 쓸 수 있습니다. 의사결정권자가 다시 볼 것은 GPU 예산의 크기보다 그 예산을 어느 단계에서 집행하는가입니다.

1.1.2 업무를 기계가 실행할 형태로 옮기는 일이 도입 효과를 정하는 이유

같은 모델에서도 업무 정의에 따라 달라지는 에이전트 결과

같은 모델과 같은 GPU를 쓰더라도 업무를 어떤 형태로 넘겼는지에 따라 에이전트의 결과는 크게 달라집니다. 「이번 분기 영업기회 목록을 정리해 줘」 같은 한 문장만 받은 모델은 어떤 시스템을 조회할지, 어떤 조건으로 거를지, 결과를 어디에 남길지를 실행할 때마다 스스로 추론해야 합니다. 반면 같은 요청이 들어오기 전에 조회 대상·필터 조건·출력 형식·정보가 모자랄 때 되물을 질문이 코드와 규칙 파일로 정해져 있으면, 모델은 그 코드를 한 번 실행하고 결과를 전달하는 일만 맡습니다. 첫 번째 방식에서는 모델의 추론 성능이 결과를 좌우하고, 두 번째 방식에서는 업무 정의의 정확도가 결과를 좌우합니다. 두 방식은 같은 운영 모델에서도 처리 시간, 도구 호출 횟수, 정답 여부가 모두 달라지며, 구체적인 전후 수치는 조회 업무 사례(7장)에 정리했습니다.

업무를 실행 가능한 형태로 옮기는 작업의 범위

업무를 기계가 실행할 형태로 옮기는 작업은 업무 설명서를 프롬프트 한 장으로 바꾸는 일보다 범위가 넓습니다. 업무 전문가가 말로 정의한 업무 하나에서는 요청 해석 규칙·값 추출 스크립트·업무 시스템 연결 도구·질문 예시를 고정한 회귀 시험·이들을 짧게 안내하는 SKILL.md가 함께 나옵니다. Anthropic은 스킬에 넣은 스크립트가 코드만 줄 수 있는 결정적 신뢰성을 제공한다고 설명하며, PDF 양식 필드를 추출하는 스크립트가 스크립트나 PDF 내용을 모델 맥락에 싣지 않고도 늘 같은 방식으로 실행되는 예를 듭니다[S2]. 업무를 옮긴다는 것은 판단 규칙을 모델의 추론에서 꺼내 코드와 시험으로 고정하는 일이고, 이 작업의 품질이 도입 효과의 상한을 정합니다. 업무 정의서 항목과 이 산출물들의 대응 관계는 2장의 업무 정의서 양식에서 자세히 다룹니다.

업무 정의 작업을 맡는 업무 전문가와 프론티어 AI의 분업

이 작업을 누가 맡느냐도 도입 효과를 좌우합니다. 판단 기준과 예외를 가장 잘 아는 사람은 업무 전문가이지만, 대부분의 업무 전문가는 코드를 쓰지 않으며 에이전트 내부 구조를 알아야 할 이유도 없습니다. MSAP.ai AAS는 이 간격을 프론티어 AI로 메웁니다. 업무 전문가가 말로 업무를 정의하면 프론티어 AI가 MCP(Model Context Protocol)로 MSAP.ai AAS에 연결해 스킬과 코드를 만들고, 곧바로 실제 운영 모델에게 요청을 보내 결과를 시험합니다. 예산을 먼저 배정할 대상은 업무 전문가가 업무를 정의하는 시간과 그 정의를 스킬로 바꾸는 설계 작업이며, 이 두 가지가 끝난 뒤에 운영 모델의 크기와 GPU 규모를 정해도 늦지 않습니다.

> 🎯 *결정 포인트 — AI Agent 예산을 GPU 증설보다 업무 정의와 스킬 설계에 먼저 배정할 것인가. 운영 GPU 규모는 업무 정의서와 질문 예시로 운영 모델을 시험한 결과를 보고 정할 수 있습니다.*

1.2 정해진 코드 경로의 워크플로와 스스로 경로를 정하는 에이전트의 경계

「AI Agent」라는 이름 아래에는 실제로 서로 다른 두 가지 설계 방식이 섞여 있습니다. Anthropic은 2024년 12월 공개한 연구 글 「Building effective agents」에서 언어 모델과 도구를 미리 정한 코드 경로로 엮는 워크플로와, 모델이 스스로 처리 순서와 도구 사용을 정하는 에이전트를 나눠 정의했습니다[S1]. 두 방식의 경계를

알아야 반복 업무 스킬을 어느 쪽에 가깝게 설계할지 정할 수 있으며, 뒤의 장에서 계속 쓰는 「판단은 코드, 모델은 한 줄 실행」 원칙도 이 구분에서 출발합니다.

1.2.1 Anthropic 연구 글로 본 워크플로와 에이전트의 정의 차이

Anthropic이 정의한 워크플로와 에이전트

Anthropic은 워크플로를 LLM과 도구를 미리 정한 코드 경로(predefined code paths)로 조율하는 시스템으로 정의합니다. 반면 에이전트는 LLM이 처리 과정과 도구 사용을 실행 중에 스스로 정하는 시스템으로 정의합니다[S1]. 두 정의를 가르는 기준은 다음 단계를 누가 정하는가이며, 워크플로에서는 개발자가 작성한 코드가 순서를 정하고 모델은 정해진 단계 안에서 글을 만들거나 분류합니다. 에이전트에서는 모델이 단계마다 결과를 보고 다음에 부를 도구를 직접 고릅니다. 같은 업무라도 처리 순서를 코드가 쥐면 워크플로에 가깝고 모델이 쥐면 에이전트에 가까우며, 두 방식의 차이를 정리하면 다음 표와 같습니다.

구분	워크플로	에이전트
다음 단계를 정하는 주체	개발자가 작성한 코드	언어 모델
같은 요청의 실행 경로	매번 같음	실행할 때마다 달라질 수 있음
잘 맞는 업무	절차와 정답 기준이 정해진 반복 업무	단계 수를 미리 알 수 없는 탐색형 업무
모델 호출 횟수	설계 단계에서 미리 셀 수 있음	업무 난이도에 따라 늘어남

출처: Anthropic, 「Building effective agents」(2024)의 정의를 바탕으로 정리[S1]

가장 단순한 해법부터 찾으라는 Anthropic의 설계 권고

Anthropic은 같은 글에서 에이전트 시스템을 만들 때 가장 단순한 해법부터 찾고, 복잡도는 결과가 실제로 좋아지는 경우에만 더하라고 권합니다. 에이전트 방식은 유연한 대신 지연과 비용이 늘고 단계마다 오류가 쌓일 수 있으므로, 단계가 정해진 업무까지 모델에게 경로 선택을 맡길 이유가 없다는 뜻입니다. 도구 설계에 관해서는 도구 설명을 새로 합류한 개발자에게 주는 문서처럼 쓰고, 여러 예시 입력으로 시험해 모델이 실수하는 부분을 찾아 고치라고 합니다. 도구 인자를 상대 경로 대신 절대 경로로 받게 바꾸자 모델의 경로 오류가 사라진 사례도 함께 소개합니다[S1]. 이 권고는 모델의 판단에 기대는 범위를 줄이고 실수할 여지를 코드 설계로 막는 방향이며, MSAP.ai AAS가 스킬 작성에 적용하는 원칙도 같은 방향을 따릅니다.

1.2.2 반복 업무 스킬이 워크플로 쪽에 무게를 두는 근거

같은 요청에 같은 결과가 필요한 반복 업무의 조건

조직이 AI Agent에 먼저 맡기려는 업무는 견적 작성, 영업기회 등록, 서버 보안 공지 검토처럼 절차와 정답 기준이 이미 정해진 반복 업무인 경우가 많습니다. 이런 업무에서는 같은 요청에 같은 결과가 나와야 하며, 어제는 처리되고 오늘은 처리되지 않는 결과는 업무 시스템에 그대로 오류로 남아 감사 추적에도 쓸 수 없습니다. 따라서 반복 업무 스킬은 처리 순서·값 검증·확인 절차를 코드로 고정하는 워크플로 쪽에 무게를 두고, 모델에게는 요청을 해석해 어떤 명령을 부를지 고르는 일과 결과를 사람에게 전달하는 일만 남깁니다. 이 방식에서도 모델은 요청 문장을 이해하는 역할을 계속 맡으므로, 사용자는 대화창에서 평소 쓰는 문장으로 업무를 요청할 수 있습니다.

지시문 해석 단계와 결정적 코드 단계로 나눈 스킬 구조

반복 업무 스킬은 지시문 해석 단계와 결정적 코드 단계로 나뉩니다. 지시문 해석 단계는 요청의 첫 줄과 본문을 분석해 부를 명령을 고르고, 정보가 모자라면 미리 정해 둔 질문을 출력한 뒤 멈춥니다. 결정적 코드 단계에서는 값 추출·후보 조회·처리 순서·사람 확인·업무 시스템 쓰기·쓰기 후 재조회 검증을 모두 스크립트가 처리합니다. 즉 에이전트의 유연함은 요청을 이해하는 지시문 해석 단계에 남기고, 결과의 일관성은 스크립트가 처리하는 결정적 코드 단계에서 확보하는 구조입니다. Anthropic이 Agent Skills 설계에서 스크립트로 결정적 신뢰성을 얻으라고 권하는 것과 같은 설계이며[S2], 이 구조를 프론티어 AI에게 지시하는 방법은 5장의 스킬 작성 지시 세 가지로 이어집니다.

업무 성격별로 고른 적합한 스킬 설계 방식

모든 업무를 워크플로로 만들 필요는 없으며, 업무 성격에 따라 알맞은 설계 방식이 다릅니다. 업무 성격별 판단 기준은 다음 표와 같습니다.

업무 성격	업무 예	알맞은 설계 방식
절차와 정답 기준이 정해진 반복 업무	견적 작성, 영업기회 등록, 정기 점검 보고	결정적 코드 중심 스킬
절차는 정해졌지만 요청 문장이 다양한 업무	자연어 고객 관리 시스템 조회, 메일 원문 기반 등록	지시문 해석 단계와 결정적 코드의 조합
단계 수를 미리 알 수 없는 탐색형 업무	장애 원인 조사, 새 기술 자료 조사	에이전트 방식과 사람 검토의 조합

MSAP.ai AAS가 우선 대상으로 삼는 업무는 표의 첫째와 둘째 행입니다. 반면 셋째 행의 탐색형 업무는 프론티어 AI의 추론 능력이 결과를 좌우하므로 작은 모델 하나에 맡기기보다 사람 검토와 함께 운영하는 편이 안전합니다. 다만 탐색형 업무라도 반복되는 하위 단계가 보이면 그 단계부터 결정적 코드로 떼어 내 둘째 행의 방식으로 옮길 수 있습니다.

1.3 소수의 전문 작업을 반복하는 에이전트 호출과 작은 언어 모델의 자리

작은 언어 모델(SLM, Small Language Model)은 일반 소비자용 장비에 올려 한 사용자의 에이전트 요청을 실용적인 지연 시간으로 처리할 수 있는 크기의 언어 모델을 말합니다[S4]. 의사결정권자가 작은 모델에 대해 가장 먼저 묻는 것은 업무를 맡길 만큼 성능이 충분한가입니다. NVIDIA 연구진은 에이전트 안의 모델 호출이 소수의 전문 작업을 반복한다는 관찰에서 출발해 작은 모델 중심 구성을 제안했지만, 그 글은 스스로 입장 논문임을 밝힌 글이므로 실험 증명과 구분해 읽어야 합니다. 따라서 작은 모델의 실제 쓸모는 논문의 주장과 별도로 우리 업무의 결과로 판단해야 합니다.

1.3.1 NVIDIA 연구진의 입장 논문이 제시한 작은 언어 모델 중심 에이전트 구성

NVIDIA 연구진이 관찰한 에이전트 안 모델 호출의 반복성

NVIDIA Research의 Belcak 외 연구진은 2025년 6월 arXiv에 공개한 입장 논문 「Small Language Models are the Future of Agentic AI」에서 에이전트 시스템 안의 언어 모델이 적은 수의 전문 작업을 거의 변화 없이 반복한다고 주장합니다[S4]. 에이전트가 처리하는 요청 하나를 나눠 보면 요청 분류, 값 추출, 도구 인

자 작성, 결과 요약처럼 형태가 정해진 호출이 이어지고, 범용 대화 능력이 꼭 필요한 호출은 그중 일부라는 뜻입니다. 이 관찰이 맞다면 모든 호출을 가장 큰 범용 모델로 보내는 구성은 필요 이상으로 비싼 선택이 됩니다. 반복 업무 스킬이 판단을 코드로 옮기면 모델에게 남는 호출은 더 좁아지므로, NVIDIA 연구진이 말한 반복성은 스킬 설계를 거친 뒤 더 뚜렷해집니다.

작은 언어 모델에 관한 입장 논문의 핵심 주장 세 가지

NVIDIA 연구진의 입장 논문이 내세우는 주장은 세 문장으로 요약할 수 있습니다. 첫째, 작은 언어 모델은 에이전트 호출 대부분에 충분히 강력하고, 본래 에이전트 작업에 더 알맞으며, 반드시 더 경제적이라고 주장합니다. 둘째, 범용 대화 능력이 꼭 필요한 곳에는 여러 모델을 섞어 쓰는 이종 구성이 자연스러운 선택이라고 봅니다. 셋째, 큰 모델 기반으로 만든 에이전트를 작은 모델 기반으로 옮기는 일반 변환 절차를 제안합니다[S4]. 이 가운데 둘째 주장인 이종 구성은 MSAP.ai AAS가 만드는 환경과 일하는 환경을 나누는 방식과 같은 방향이며, 1.4절의 역할 분담으로 이어집니다.

NVIDIA 입장 논문의 근거 수준과 주장의 인용 범위

다만 NVIDIA 연구진의 글은 작은 모델이 특정 업무에서 큰 모델과 같은 정확도를 낸다는 것을 실험으로 증명한 연구가 아니며, 저자들도 이 글을 입장 논문(position paper)으로 소개합니다. 이 주장은 설계 방향의 근거로만 쓰고, 작은 모델이 실제 업무를 처리할 수 있는지는 제품 동작과 업무 결과로 따로 확인해야 합니다. 근거의 수준을 구분하면 다음 표와 같습니다.

주장	근거	근거 수준
에이전트 안의 모델 호출은 소수의 전문 작업을 반복함	NVIDIA 연구진 입장 논문[S4]	주장(관찰과 논증)
작은 모델이 에이전트 호출 대부분에 충분함	NVIDIA 연구진 입장 논문[S4]	주장(실험 증명 아님)
스킬 안의 스크립트는 결정적 신뢰성을 줌	Anthropic Agent Skills 설명[S2]	벤더 공식 설명
판단을 코드로 옮긴 스킬을 작은 운영 모델이 정답으로 실행함	한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영했을 때 관찰된 동작(7장)	제품 동작

1.3.2 「작은 모델은 성능이 낮다」는 우려와 같은 요청에 같은 정답을 내는가로 바꾼 성능 기준

공개 벤치마크 점수로 잰 작은 모델 성능의 의미와 한계

작은 모델의 성능이 낮다는 우려는 대부분 공개 벤치마크 점수에서 나옵니다. 수학 문제, 코딩 문제, 일반 상식 질의처럼 범용 능력을 재는 시험에서는 파라미터가 많은 모델이 높은 점수를 받는 경우가 많고, 이 차이는 반복 업무에서도 무시할 수 없는 사실입니다. 하지만 벤치마크 점수는 모델이 처음 보는 문제를 스스로 풀어야 한다는 전제에서 잰 값입니다. 스킬이 판단 규칙을 코드로 쥐고 있으면 모델이 스스로 풀어야 할 문제는 요청 문장에서 명령과 값을 고르는 일로 줄어들고, 이 일은 범용 시험 문항과 난도가 크게 다릅니다. 반복 업무용 모델을 고를 때 벤치마크 점수만 보면 업무에 필요한 크기보다 큰 모델을 고르게 되고, 그만큼 GPU 비용도 커집니다.

업무 결과 일치로 다시 정의한 반복 업무 모델의 성능 기준

반복 업무에서 모델의 성능은 같은 요청에 같은 정답을 내는가로 재야 합니다. 업무 전문가가 정한 질문 예시를 운영 모델에 넣었을 때 결과가 정답과 일치하는지, 같은 질문을 다시 넣어도 같은 결과가 나오는지, 몇 번의 도구 호출과 몇 초 안에 끝나는지가 실제 업무 투입 여부를 가릅니다. 이 기준은 Anthropic이 도구 설계에서 여러 예시 입력으로 시험해 모델의 실수를 찾으라고 권한 방법과 같은 방향입니다[S1]. 질문 예시는 스킬을 만드는 재료이면서 동시에 운영 모델의 성능을 재는 시험 문항이고, 두 성능 기준의 차이는 다음 표와 같습니다.

구분	벤치마크 점수 기준	업무 결과 기준
측정 대상	범용 문제 풀이 능력	우리 업무 질문 예시의 정답 일치
측정 문항	공개 시험 문항	업무 전문가가 정한 질문 예시
반복 실행	평가하지 않음	같은 질문에 같은 결과인지 확인
비용과 속도	별도로 측정	도구 호출 횟수와 처리 시간을 함께 기록
판단 결과	모델 간 순위	운영 투입 여부

같은 운영 모델에서 스킬 설계에 따라 달라진 조회 업무 결과

업무 결과 기준을 적용하면 작은 모델에 대한 판단이 달라집니다. 한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영했을 때, 같은 활성 파라미터 3B 급 오픈소스 MoE 모델이 판단을 모델에 맡긴 스킬에서는 오답을 내고 판단을 코드로 옮긴 스킬에서는 정답을 내는 동작을 보였습니다. 정답을 낸 스킬은 도구 호출 횟수와 처리 시간도 크게 줄었으며, 구체적인 전후 수치와 스킬 구조는 7장 사례에 정리했습니다. 따라서 의사결정권자가 확인할 것은 모델의 순위보다 우리 업무의 질문 예시를 이 모델이 정답으로 처리하는가이며, 이 확인은 GPU를 추가로 구매하기 전에 끝낼 수 있습니다.

1.4 만드는 쪽은 프론티어 AI, 일하는 쪽은 로컬 모델인 이종 구성과 MSAP.ai AAS의 위치

스킬을 만드는 일과 스킬을 매일 실행하는 일은 요구하는 능력이 다릅니다. 업무 문장을 읽고 규칙 파일·스크립트·회귀 시험을 설계하는 일은 높은 추론 능력이 필요하지만 스킬마다 한 번이면 되고, 만든 스킬을 실행하는 일은 판단이 코드에 들어 있어 작은 모델로 충분하지만 업무 요청마다 반복합니다. NVIDIA 연구진이 제안한 이종 구성도 이처럼 작업 성격에 따라 모델을 나눠 쓰는 방식이며[S4], MSAP.ai AAS는 이 분업을 인터넷이 되는 개발 환경과 조직 안 운영 환경 사이의 연결로 구현합니다.

1.4.1 스킬 제작을 맡는 프론티어 AI와 매일 실행을 맡는 조직 안 작은 모델의 역할 분담

스킬 제작을 맡는 프론티어 AI와 MCP 연결

스킬을 만드는 쪽은 인터넷에 연결된 개발 환경입니다. 업무 전문가가 Claude 같은 프론티어 AI에게 업무를 말로 설명하면, 프론티어 AI는 MCP로 MSAP.ai AAS에 연결해 스킬을 만들고 올립니다. MCP는 LLM 애플리케이션과 외부 데이터·도구를 잇는 개방형 프로토콜로, 연결을 거는 호스트, 호스트 안의 클라이언트, 기능을 제공하는 서버라는 세 역할을 정하고 JSON-RPC 2.0 메시지로 통신합니다[S3]. MSAP.ai AAS는 이 구조에서 MCP 서버를 맡아 에이전트 목록 조회, 스킬 만들기·올리기·켜고 끄기, 채팅 시작·보내기·응답 기다리기, 예약 작업과 비밀값 관리를 도구로 제공합니다. 프론티어 AI는 스킬을 만든 직후 MSAP.ai AAS 채팅으로 실제 운영 모델에게 요청을 보내 결과를 확인하고, 실패한 질문 예시를 근거로 스킬을 다시 고칠 수 있습니다.

매일 실행을 맡는 조직 안 작은 모델과 사내 게이트웨이

스킬을 실행하는 쪽은 조직 안의 운영 환경이며, MSAP.ai AAS는 오픈소스 Hermes Agent 기반 실행 환경을 웹 콘솔과 MCP 서버로 다룹니다. Hermes Agent는 스킬을 YAML 앞머리를 단 SKILL.md 디렉터리로 관리하고, 이름과 설명을 먼저 본 뒤 필요할 때만 본문과 참고 파일을 읽는 점진적 공개(progressive disclosure) 방식을 씁니다[S8]. 운영 환경의 모델 호출은 사내 게이트웨이인 LiteLLM 하나를 거치며, LiteLLM은 여러 모델 제공자를 OpenAI 호환 인터페이스 하나로 묶고 사용자·팀별 가상 키와 비용 집계를 제공합니다[S9]. 게이트웨이 뒤에서는 vLLM이 OpenAI 호환 서버로 로컬 LLM을 서빙하며 Chat Completions 형식의 도구 호출을 지원합니다[S10]. 이 구성에서는 고객·금액 자료가 조직 밖으로 나가지 않고, 모델 사용량과 비용도 게이트웨이 한 곳에서 거버넌스 기준에 맞춰 관리할 수 있습니다.

보급형 GPU에서 실행할 수 있는 MoE 운영 모델의 구조

운영 모델로는 활성 파라미터 3B 급 오픈소스 MoE 모델처럼 보급형 GPU에서도 실행할 수 있는 크기를 씁니다. MoE(Mixture of Experts, 전문가 혼합)는 층마다 라우터가 여러 전문가 가운데 일부만 골라 계산하는 구조로, 예를 들어 Mixtral 8×7B는 토큰마다 47B 파라미터에 접근할 수 있지만 추론 때는 활성 파라미터 13B만 씁니다[S6]. 같은 방식의 공개 모델인 Qwen3-30B-A3B는 전체 30.5B 파라미터 가운데 전문가 128개 중 8개를 골라 활성 3.3B만 계산하며 Apache-2.0 라이선스로 공개되어 있습니다[S7]. 앞서 본 대로 디코드 단계의 속도는 토큰마다 메모리에서 읽는 가중치 양이 좌우하므로[S5], 활성 파라미터가 작은 MoE 모델은 같은 장비에서 더 빠르게 토큰을 생성합니다. 다만 Qwen3-30B-A3B는 범주를 설명하기 위한 공개 사례이며, 어떤 장비와 모델 조합이 쓸 만한지는 6장에서 판단을 코드로 옮긴 설계와 함께 따집니다.

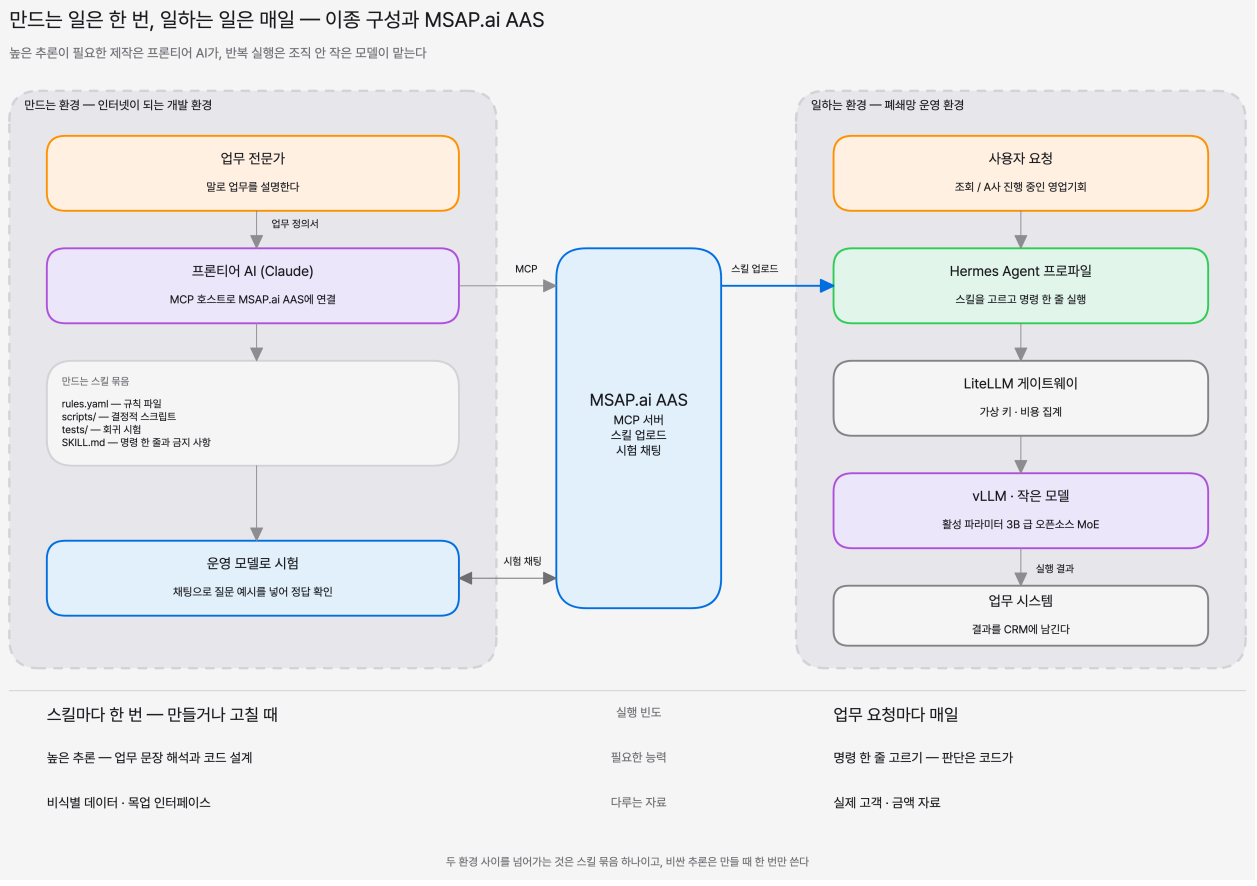


그림 1-1. 만드는 환경의 프론티어 AI와 일하는 환경의 작은 모델을 MSAP.ai AAS가 잇는 이종 구성

개발 환경은 비식별 데이터베이스와 목업 인터페이스로 내부 시스템을 흉내 낸 디지털 트윈이고, 완성한 스킬은 스킬 업로드나 스킬 허브로 폐쇄망 운영 환경에 옮깁니다. 스킬을 부르는 스킬, 크기가 큰 스킬, 한글 이름 스킬도 옮긴 뒤 그대로 동작하므로 업무 전문가는 개발 환경에서 시험한 구조를 운영 환경에서 다시 만들 필요가 없습니다. 세 주체의 역할을 나누면 다음 표와 같습니다.

구분	프론티어 AI	조직 안 작은 모델	MSAP.ai AAS
맡는 일	업무 문장 해석, 규칙 파일·스크립트·회귀 시험·SKILL.md 작성	요청 해석, 스킬 명령 한 줄 실행, 결과 전달	MCP 도구 제공, 운영 모델 시험 채팅, 스킬 업로드, 실행 환경 관리
실행 빈도	스킬을 만들거나 고칠 때	업무 요청마다	상시
실행 위치	인터넷이 되는 개발 환경	폐쇄망 운영 환경	두 환경 모두
다루는 자료	비식별 데이터와 목업 인터페이스	실제 고객·금액 자료	스킬과 실행 기록

업무 담당자가 콘솔에서 정의하는 세 대상

MSAP.ai AAS 콘솔에서 업무 담당자가 정의하는 대상은 에이전트, 스킬, 예약 작업 세 가지로 나뉩니다. 에이전트와 그 안의 프로파일은 「누가」 일하는지를 정하는 단위로 성격·모델·메신저 연결을 담고, 스킬은 「어떻게」 일하는지를 적은 문서와 스크립트이며, 예약 작업은 「언제」 실행할지를 정한 일정입니다. 세 대상을 나눠 두면 프로파일의 성격과 모델은 그대로 둔 채 스킬 하나만 고치거나, 같은 스킬을 매일 아침과 매주 월요일에 각각 실행하도록 일정만 더할 수 있습니다. 모델 호출과 에이전트 실행처럼 전문 지식이 필요한 부분은 실행 환경이 맡고 업무 규칙이 바뀐 부분만 업무 담당자가 손보는 분업이므로, IT 부서는 업무 규칙이 바뀔 때마다 개발 요청을 받지 않고 플랫폼·모델·자격증명의 안정성에 집중할 수 있습니다.

아래 그림은 정의와 관측은 MSAP.ai AAS 콘솔이 맡고 실행은 에이전트 플랫폼이 맡는 구조와, 한 콘솔에 모은 만들기·시키기·지켜보기·나누기 네 기능 묶음을 보여 줍니다.



그림 1-2. MSAP.ai AAS가 정의하는 세 대상과 한 콘솔에 모은 네 기능 묶음

이 구조에서 스킬을 올리고 켜는 일은 「만들기」 묶음에 속하므로, 프론티어 AI가 MCP로 올린 스킬도 업무 담당자가 같은 콘솔에서 확인하고 켜거나 끌 수 있습니다.

1.4.2 의사결정권자가 이 문서에서 내릴 판단 네 가지와 근거 장 안내

MSAP.ai AAS 도입 검토에서 내릴 판단 네 가지

MSAP.ai AAS 도입을 검토하는 의사결정권자가 내릴 판단은 네 가지입니다. 네 판단은 서로 떨어져 있지 않으며, 업무 정의를 먼저 한다는 관점을 받아들이면 GPU 예산, 망 구성, 스킬 작성 체계, 남는 위험의 관리 방식이 같은 순서로 이어집니다. 각 판단에 필요한 근거와 근거가 있는 장은 다음 표와 같고, 도입을 시작하는 순서는 9장(남은 한계와 도입 순서)에 정리했습니다.

판단 질문	판단에 필요한 근거	근거 장
운영 모델용 GPU를 보급형으로 계획해도 업무 결과를 낼 수 있는가	판단을 코드로 옮긴 설계에서의 처리 시간과 정답 여부	6장
프론티어 AI를 쓰면서도 운영 환경을 폐쇄망으로 유지할 수 있는가	만드는 환경과 일하는 환경의 분리, 스킬 업로드 경로	3장
코드를 쓰지 않는 업무 전문가가 스킬 작성을 맡을 수 있는가	업무 정의서 양식과 MCP 기반 제작 절차	2장, 4장
도입 뒤에도 남는 위험은 무엇이고 누가 관리하는가	아직 어려운 업무 유형과 운영 중 스킬 변경 통제	9장

GPU 예산과 폐쇄망 유지 판단을 함께 묶는 업무 정의서

표의 앞 두 판단은 업무 정의서가 있어야 답할 수 있습니다. 보급형 GPU로 충분한지는 업무 전문가가 정한 질문 예시를 작은 운영 모델로 실행해 정답 일치와 처리 시간을 확인한 결과로 정해지고, 그 질문 예시가 바로 업무 정의서의 일부입니다. 폐쇄망 유지 여부도 마찬가지로, 프론티어 AI가 다루는 개발 환경에는 비식별 데이터만 두고 실제 자료는 운영 환경에만 두는 구성이 가능한지가 업무 정의서의 입력값과 연결 시스템 목록에서 드러납니다. MCP 명세는 호스트가 도구를 부르기 전에 사용자 동의를 받고 사용자가 모든 자료 접근을 통제해야 한다는 원칙을 두고 있어[S3], 개발 환경의 프론티어 AI가 무엇에 접근하는지도 업무 정의서 단계에서 미리 정할 수 있습니다.

업무 전문가 작성 체계와 남는 위험을 판단하는 기준

뒤의 두 판단은 사람과 운영에 관한 것입니다. 업무 전문가가 스킬 작성을 맡을 수 있는지는 업무 전문가가 지시문 해석·결정적 코드·질문 예시를 담은 업무 정의서를 쓸 수 있는지로 판단하며, 코드 작성과 시험은 프론티어 AI가 맡으므로 코딩 역량은 판단 기준이 아닙니다. 남는 위험에는 단계 수를 미리 알 수 없는 탐색형 업무처럼 작은 모델에 맡기기 어려운 업무 유형과, 운영 중 에이전트가 스킬을 스스로 고치는 문제가 포함됩니다. 예를 들어 Hermes Agent는 에이전트가 도구로 스킬을 직접 만들고 고칠 수 있게 하며, 그 쓰기를 사람 검토 대기로 돌리는 설정을 함께 제공합니다[S8]. 따라서 1장의 판단을 받아들인 조직이 먼저 정할 실행 단위는 부서 반복 업무 하나와 그 업무의 업무 정의서이며, 이 한 건이 MSAP.ai AAS 파일럿 대상이 됩니다. 그 업무 정의서를 에이전트가 코드로 옮길 수 있는 문장으로 쓰는 방법은 2장에서 다룹니다.

2장: 업무 전문가가 말로 정의하는 업무와 그 산출물

MSAP.ai AAS에서 스킬을 만드는 출발점은 업무 전문가가 쓴 업무 정의 문장입니다. 업무 전문가는 영업 조회, 견적 작성, 서버 보안 조치처럼 반복되는 업무의 판단 기준을 가장 정확하게 아는 현업 담당자이고, 스킬은 그 판단 기준을 에이전트가 읽고 실행할 수 있도록 지시문·스크립트·참고 파일을 한 디렉터리에 묶은 것입니다. Anthropic은 2025년 10월 Agent Skills를 공개하면서 스킬을 SKILL.md 파일을 담은 디렉터리로 정의했고, 오픈소스 Hermes Agent도 같은 형식의 스킬을 개방형 표준과 호환되는 방식으로 읽습니다[S2][S8]. 형식이 공개 표준으로 정리되었으므로 이제 도입 성과는 그 디렉터리 안에 어떤 판단이 얼마나 정확하게 들어가느냐에 달려 있고, 그 판단을 문장으로 내놓을 수 있는 사람은 개발자보다 업무 전문가입니다. 이 장에서는 업무 전문가와 프론티어 AI의 역할 경계, 코드로 옮길 수 있는 업무 문장의 조건, 지시문 해석·결정적 코드·질문 예시를 한 장에 담는 업무 정의서 양식, 정의서 한 장에서 나오는 산출물의 구조를 다룹니다. 의사결정권자는 이 장을 근거로 우리 부서에 코딩 없이 업무 정의서를 쓸 수 있는 업무 전문가가 있는지 판단할 수 있습니다.

2.1 코드 작성은 프론티어 AI, 판단 기준은 업무 전문가 — 코딩 역량을 전제하지 않는 역할 경계

스킬 하나를 만드는 일은 판단 기준을 정하는 일과 그 기준을 코드로 옮기는 일로 나뉩니다. 판단 기준은 업무를 매일 처리하는 업무 전문가에게 있고, 코드와 시험은 Claude 같은 프론티어 AI가 짧은 시간에 작성할 수 있습니다. Anthropic은 에이전트가 쓸 도구의 설명을 팀에 새로 온 개발자에게 주는 문서처럼 쓰고, 사용 예와 경계 사례와 입력 형식을 담으라고 권합니다[S1]. 그런 문서에 들어갈 내용은 업무를 아는 사람만 쓸 수 있으므로, 스킬 제작에서 업무 전문가가 맡는 일은 개발자를 대신하는 일과 성격이 다르며 판단 기준의 소유자로서 그 기준을 문장으로 내놓는 일입니다. 아래 표는 두 역할의 경계를 요약합니다.

구분	업무 전문가	프론티어 AI
맡는 일	판단 규칙, 예외, 사람이 확인할 단계, 정답 판정	규칙 파일, 결정적 스크립트, 회귀 시험, SKILL.md 작성
필요한 역량	업무 지식과 판단 기준을 문장으로 쓰는 능력	코드 작성, 시험 설계, 스킬 형식 이해
쓰지 않아도 되는 것	프로그래밍 언어, 에이전트 내부 구조, 모델 선택	업무 판단 기준 (업무 전문가의 문장을 따른다)
결과물	업무 정의서 한 장과 질문 예시 목록	운영 환경에 올릴 스킬 디렉터리

2.1.1 업무 전문가가 서술하는 판단 규칙과 예외, 사람이 확인할 지점

업무 전문가가 문장으로 남기는 판단 규칙

업무 전문가가 쓰는 문서는 판단 규칙을 담은 평서문이며, 그 안에 코드는 한 줄도 들어가지 않습니다. 예를 들어 영업기회를 등록하는 업무라면 「메일 원문에서 수요기관명, 사업명, 예상 금액, 마감일을 읽고, 수요기관이 CRM에 이미 있으면 그 고객사를 쓰고 없으면 새로 만든다」 같은 문장이 판단 규칙입니다. 이 문장에는 어떤 값을 읽는지, 어떤 순서로 무엇을 확인하는지, 결과가 어디에 남는지가 모두 들어 있으므로 프론티어 AI가 그대로 스크립트로 옮길 수 있습니다. Anthropic은 워크플로를 미리 정한 코드 경로로 모델과 도구를 엮은 시스템으로

정의하는데, 업무 전문가의 판단 규칙은 바로 그 코드 경로의 설계도에 해당합니다[S1]. 업무 전문가에게 필요한 역량은 자기 업무를 다른 사람이 똑같이 따라 할 수 있을 만큼 구체적으로 설명하는 능력이며, 프로그래밍 역량은 이 일에 필요하지 않습니다.

규칙 밖의 경우를 다루는 예외 문장

판단 규칙만 적으면 스킬은 순조로운 경우에만 맞게 동작하고, 실제 업무의 상당 부분을 차지하는 예외에서 모델이 추측을 시작합니다. 업무 전문가가 예외를 따로 적어야 하는 이유는 예외 처리 방식이 조직마다 다르고 그 차이가 문서 어디에도 쓰여 있지 않은 경우가 많기 때문입니다. 같은 이름의 고객사가 두 곳이면 어느 쪽을 고를지, 메일에 금액이 없으면 비워 둘지 되물을지는 업무 전문가만 답할 수 있습니다. 이런 예외를 「금액이 없으면 등록하지 말고 금액을 묻는다」처럼 문장으로 남겨 두면 프론티어 AI는 그 문장을 정해진 질문을 출력하고 멈추는 코드로 바꿉니다.

사람이 결과를 확인해야 하는 단계와 그 조건

업무 전문가가 세 번째로 정하는 것은 사람이 결과를 보고 승인해야 하는 단계입니다. 조회처럼 결과를 보여 주기만 하는 업무는 확인 단계가 필요 없지만, CRM에 영업기회를 만들거나 견적을 발행하는 업무는 쓰기 직전에 사람이 한 번 확인해야 합니다. MCP(Model Context Protocol, 모델과 외부 도구를 잇는 개방형 프로토콜) 명세도 사용자가 모든 자료 접근과 작업에 명시적으로 동의하고 통제할 수 있어야 하며, 호스트가 도구를 부르기 전에 사용자 동의를 받아야 한다고 정합니다[S3]. 그래서 업무 정의서에는 「등록 전에 드라이런 결과를 보여 주고 사람이 확인하면 쓴다」처럼 확인 단계와 그 조건을 업무 전문가가 직접 적습니다.

업무 전문가가 작성하는 항목 목록

업무 전문가가 업무 정의서에 적는 항목은 다음 여덟 가지입니다. 항목마다 업무 지식만으로 채울 수 있고, 프로그래밍 용어는 하나도 필요하지 않습니다.

- 업무 목적: 이 업무가 끝나면 누가 무엇을 얻는가
- 요청 형태: 사람이 에이전트에게 어떤 첫 줄과 어떤 본문으로 요청하는가
- 입력값: 요청에서 읽어야 할 값과 각 값을 찾는 방법
- 처리 규칙: 값을 확인하고 결과를 만드는 순서
- 예외: 값이 없거나 후보가 여럿이거나 이미 처리된 경우의 대응
- 확인 단계: 사람이 결과를 보고 승인해야 하는 단계와 그 조건
- 완료 상태: 일이 끝났다고 판정할 수 있는 시스템의 상태
- 질문 예시: 실제로 들어올 법한 요청 문장과 각각의 기대 결과

2.1.2 프론티어 AI가 맡는 코드와 시험 작성, 업무 전문가에게 남는 정답 판정

프론티어 AI가 작성하는 스크립트와 회귀 시험

업무 정의서가 완성되면 프론티어 AI는 그 문장을 규칙 파일, 결정적 스크립트, 회귀 시험, 짧은 SKILL.md로 옮깁니다. 결정적 코드는 같은 입력에 언제나 같은 결과를 내는 코드를 말하며, Anthropic은 스킬에 넣은 스크립트가 모델의 토큰 생성으로는 얻을 수 없는 결정적 신뢰성을 준다고 설명합니다[S2]. 예를 들어 PDF 양식에서 필드를 뽑는 스크립트는 스크립트 본문이나 PDF 내용을 모델의 맥락에 싣지 않고도 매번 같은 방식으로 실행됩니다[S2]. 회귀 시험은 업무 전문가가 적은 질문 예시마다 기대 결과를 붙여 두고 스킬을 고칠 때마다 다시 돌려 보는 시험이며, 이것도 프론티어 AI가 질문 예시 목록에서 그대로 만들어 냅니다.

MCP로 연결된 개발 환경에서 실제 실행 모델로 하는 시험

MSAP.ai AAS는 오픈소스 Hermes Agent 기반 실행 환경을 웹 콘솔과 MCP 서버로 다루며, MCP 도구로 에이전트 목록 조회, 스킬 만들기·올리기·켜고 끄기, 채팅 시작·보내기·기다리기를 제공합니다. MCP에서 연결을 거는 LLM 애플리케이션을 호스트, 기능을 제공하는 쪽을 서버라고 부르므로, 이 구성에서는 업무 전문가가 쓰는 프론티어 AI가 호스트이고 MSAP.ai AAS가 서버입니다[S3]. 이 연결 덕분에 프론티어 AI는 스킬을 만든 직후 MSAP.ai AAS 채팅으로 운영 환경과 같은 작은 모델에게 실제 요청을 넣어 볼 수 있습니다. 즉, 업무 전문가는 코드를 읽지 않고도 자기가 쓴 질문 예시가 운영 모델에서 어떤 답으로 돌아오는지 채팅 화면에서 확인할 수 있습니다.

업무 전문가에게 남는 정답 판정

코드와 시험을 프론티어 AI가 맡아도 결과가 맞는지를 판정하는 일은 업무 전문가에게 남습니다. 회귀 시험의 기대 결과를 정하는 사람이 업무 전문가이고, 시험이 모두 통과한 뒤에도 실제 요청에서 나온 답이 업무상 맞는지 최종 판단하는 사람도 업무 전문가이기 때문입니다. Anthropic은 여러 예시 입력으로 도구를 시험해 모델이 어떤 실수를 하는지 찾아 고치라고 권하는데, 그 실수를 실수로 알아보는 기준은 업무 지식에서 나옵니다[S1]. 아래 표는 스킬 하나를 만드는 동안의 작업별 담당을 정리한 것입니다.

작업	업무 전문가	프론티어 AI	MSAP.ai AAS
업무 정의서 작성	직접 작성	빠진 항목을 질문	—
규칙 파일·스크립트 작성	검토하지 않음	작성	—
회귀 시험 작성	기대 결과 제공	시험 코드 작성	—
운영 모델로 시험	결과 판정	MCP로 채팅 요청	운영 모델로 실행
운영 환경 반입	반입 승인	스킬 묶음 준비	스킬 업로드·스킬 허브

결과적으로 업무 전문가가 직접 손대는 산출물은 업무 정의서와 질문 예시뿐이고, 나머지는 프론티어 AI와 MSAP.ai AAS가 처리합니다. 업무 전문가를 고르는 기준은 그 업무의 예외와 정답을 가장 많이 아는 사람인지 여부이고, 코딩 경험은 고려 대상에서 빠집니다.

시스템 연동과 실행 기반을 맡는 IT 담당자

업무 전문가와 프론티어 AI의 분업 바깥에는 IT 담당자의 몫이 남습니다. 어떤 모델을 쓰고, 누가 어떤 에이전트와 스킬에 접근하며, 스킬이 어느 사내 시스템과 연결될지는 업무 문장으로 정할 수 있는 영역이 아니라 실행 기반을 관리하는 사람이 정하는 영역입니다. 업무 전문가가 정한 규칙이 안전하게 실행되도록 접속 권한과 모델 게이트웨이, 업무 시스템 연결을 준비하는 일이 여기에 들어가며, 세 역할의 책임 분담은 8.4.2에서 표로 다시 정리합니다.

2.2 에이전트가 코드로 옮길 수 있는 업무 문장의 조건 — 입력값, 처리 규칙, 완료 상태

같은 업무를 설명하더라도 문장에 입력값, 처리 규칙, 완료 상태가 모두 들어 있어야 프론티어 AI가 그 문장을 코드로 옮길 수 있습니다. 세 요소 가운데 하나라도 빠지면 빠진 부분은 운영 환경의 작은 모델이 매번 추측해야 하고, 추측은 요청마다 다른 결과를 냅니다. Anthropic은 에이전트를 모델이 스스로 순서와 도구 사용을 정하는

시스템으로, 워크플로를 미리 정한 코드 경로를 따르는 시스템으로 구분합니다[S1]. 업무 문장이 모호할수록 스킬은 에이전트 쪽으로 기울고, 문장이 구체적일수록 워크플로 쪽으로 옮겨 가 같은 요청에 같은 결과를 냅니다.

2.2.1 모델 판단으로 넘어가는 모호한 업무 문장과 코드로 옮겨지는 업무 문장의 대비

모호한 문장과 구체적인 문장의 세 가지 대비

아래 표는 같은 업무를 두 가지 문장으로 쓴 예입니다. 왼쪽 문장은 사람끼리는 통하지만, 에이전트에게 주면 넷째 열에 적은 판단을 모델이 혼자 내려야 합니다. 오른쪽 문장은 입력값, 처리 규칙, 완료 상태를 모두 담고 있어 프론티어 AI가 스크립트로 옮길 수 있습니다.

업무	모호한 문장	코드로 옮길 수 있는 문장	모호한 문장에서 모델 추측으로 남는 부분
CRM 조회	「요즘 진행 중인 큰 건 좀 알려 줘」	「영업 단계가 제안 또는 협상인 영업기회를 예상 금액이 큰 순서로 10건 보여 주고, 고객사 이름이 있으면 그 고객사로 좁힌다」	「진행 중」의 단계 범위, 「큰 건」의 기준, 보여 줄 건수
영업기회 등록	「이 메일 보고 영업기회 하나 만들어 줘」	「첫 줄이 영업기회이면 메일 원문에서 수요기관명·사업명·마감일을 읽고, 수요기관이 없으면 묻고, 드라이런 결과를 확인받은 뒤 등록한다」	어떤 값을 어디서 읽을지, 값이 없을 때의 대응, 확인 없이 써도 되는지
보안 공지 반영	「보안 공지 보고 알아서 조치해 줘」	「보안 공지 가운데 서버 목록의 제품·버전에 해당하는 항목만 골라 조치 계획 초안을 만들고, 담당자 확인을 받은 뒤 반영한다」	우리 서버에 해당하는지 판정하는 기준, 조치 범위, 반영 전 확인 여부

모호한 문장이 작은 모델에게 남기는 판단

모호한 문장의 문제는 사람이 읽을 때는 드러나지 않습니다. 업무를 아는 동료는 「큰 건」이 어느 금액 이상인지, 「진행 중」에 어떤 단계가 들어가는지 이미 알고 있으므로 같은 문장에서 같은 답을 냅니다. 반면 운영 환경의 작은 모델은 그 기준을 모르므로 요청이 들어올 때마다 기준을 새로 추측하고, 추측한 기준을 확인하려고 도구를 여러 번 부르다가 응답 시간과 입력 토큰을 함께 늘립니다. 즉, 모호한 문장은 업무 판단을 모델에게 넘기는 문장이고, 판단을 넘겨받은 작은 모델은 같은 질문에도 다른 결과를 낼 수 있습니다.

구체적인 문장이 코드로 바뀌는 방식

오른쪽 문장은 문장 안의 낱말이 그대로 코드의 조건이 됩니다. 「영업 단계가 제안 또는 협상」은 조회 조건이 되고, 「예상 금액이 큰 순서로 10건」은 결과 순서와 건수가 되며, 「고객사 이름이 있으면 좁힌다」는 선택적 필터가 됩니다. Anthropic은 가장 단순한 해법부터 찾고 결과가 실제로 나아질 때만 복잡도를 더하라고 권하는데, 업무 문장을 구체적으로 쓰면 모델이 경로를 정하는 복잡한 구조 없이 정해진 코드 경로만으로 업무가 끝납니다[S1].

그래서 업무 정의서를 쓰기 전에 업무 전문가가 먼저 확인할 것은 문장 속 형용사와 부사를 숫자, 단계 이름, 시스템 상태로 바꿀 수 있는지입니다.

2.2.2 업무 문장에서 빠지기 쉬운 예외 처리와 완료 기준 점검표

업무 문장에서 자주 빠지는 예외 항목

업무 전문가가 처음 쓴 업무 문장에는 순조로운 경우의 처리 순서는 잘 들어가지만, 값이 없을 때와 후보가 여럿일 때의 대응은 자주 빠집니다. 업무 전문가에게는 너무 당연해서 적을 필요를 느끼지 못하는 판단이기 때문입니다. 하지만 작은 모델은 그 당연한 판단을 모르므로 빠진 항목이 있으면 스스로 기준을 만들고, 그 기준은 다음 요청에서 바뀔 수 있습니다. 다음 점검표는 업무 정의서를 넘기기 전에 업무 전문가가 확인할 예외 항목과, 그 항목이 빠졌을 때 작은 모델이 하는 행동을 함께 보여 줍니다.

점검 항목	업무 전문가가 답할 질문	빠졌을 때 작은 모델의 행동
값이 없을 때	필수 값이 요청에 없으면 묻는가, 비워 두는가, 멈추는가	비슷한 값을 지어내거나 다른 시스템을 뒤져 찾으려 한다
후보가 여럿일 때	같은 이름의 고객사가 둘이면 어떤 기준으로 고르는가	첫 번째 후보를 고르거나 후보를 모두 섞어 답한다
표기가 다를 때	약칭, 영문 이름, 연도가 붙은 이름을 같은 대상으로 보는가	일치하는 값이 없다고 답하거나 엉뚱한 대상을 고른다
이미 처리되었을 때	같은 영업기회가 이미 있으면 새로 만드는가, 기존 것을 쓰는가	중복 항목을 만든다
시스템이 응답하지 않을 때	재시도할지, 사람에게 넘길지	같은 도구를 계속 다시 부른다
쓰기 전 확인	어떤 작업에서 사람의 확인이 필요한가	확인 없이 쓰거나 매 단계마다 확인을 요청한다
완료 상태	어떤 시스템 상태가 되면 끝난 것으로 보는가	결과를 다시 확인하지 않고 끝났다고 답한다

완료 상태를 시스템 상태로 적는 방법

점검표의 마지막 항목인 완료 상태는 「등록했다」 같은 행위가 아니라 「CRM에서 사업명으로 다시 조회했을 때 영업기회가 한 건 나오고 금액이 요청과 같다」 같은 시스템 상태로 적어야 합니다. 행위로 적으면 모델이 쓰기 명령을 보낸 것만으로 일이 끝났다고 판단하지만, 상태로 적으면 프론티어 AI가 쓰기 뒤에 다시 조회해 결과를 검증하는 코드를 붙일 수 있습니다. MSAP.ai AAS의 스킬이 쓰기 뒤 재조회 검증까지 스크립트로 처리하는 것도 업무 정의서가 완료 상태를 이렇게 적어 두었기 때문입니다.

실수 자체를 어렵게 만드는 문장 설계

예외를 적을 때는 「주의한다」보다 「이 경우에는 이렇게 한다」로 쓰는 편이 코드로 옮기기 쉽습니다. Anthropic은 모델이 상대 경로를 잘못 쓰는 문제를 도구 인자로 절대 경로만 받도록 바꿔 없앤 사례를 들며, 인자를 바꿔 실수 자체가 어렵게 만드는 설계를 권합니다[S1]. 업무 문장도 마찬가지여서, 「고객사를 잘 확인한다」라고 쓰면 모델의 주의력에 기대게 되지만 「고객사 후보가 둘 이상이면 후보 목록을 보여 주고 번호로 고르게 한다」라고 쓰

면 실수가 일어날 수 있는 경우 자체가 줄어듭니다. 점검표의 각 질문에는 업무 전문가가 행동 하나로 끝나는 답을 적어야 합니다.

2.3 지시문 해석과 결정적 코드, 질문 예시를 한 장에 담는 업무 정의서 양식

업무 정의서는 업무 전문가가 한 장으로 채우는 문서이며, 프론티어 AI가 스킬을 만들 때 받는 유일한 업무 입력입니다. 양식의 항목은 작은 모델이 같은 결과를 내도록 프론티어 AI에게 주는 세 가지 지시, 곧 지시문 해석, 결정적 코드, 질문 예시와 하나씩 연결되어 있습니다. 지시문 해석은 요청 첫 줄과 본문을 분석해 어떤 명령을 부를지 고르고 모르면 정해진 질문을 출력하고 멈추는 단계이고, 결정적 코드는 값 추출부터 재조회 검증까지를 스크립트가 맡는 원칙이며, 질문 예시는 다양한 요청 문장을 회귀 시험으로 고정하는 일입니다. 각 지시의 설계 규칙은 5장에서 자세히 다루며, 업무 정의서는 그 세 지시에 들어갈 재료를 업무 전문가의 말로 모으는 양식입니다.

2.3.1 업무 정의서 항목과 스킬 산출물의 대응표

업무 정의서 항목과 산출물, 지시의 연결

업무 정의서의 항목은 모두 스킬 디렉터리 안의 특정 파일로 바뀝니다. 업무 전문가가 항목 하나를 비워 두면 그 항목에 대응하는 파일이 비거나 모델의 추측으로 채워지므로, 대응 관계를 알면 업무 전문가는 어느 항목에 시간을 더 써야 하는지 알 수 있습니다. 아래 표는 양식의 항목과 그 항목에서 만들어지는 산출물, 이어지는 지시를 함께 보여 줍니다.

업무 정의서 항목	업무 전문가가 적는 내용	바뀌는 산출물	이어지는 지시
요청 형태	요청 첫 줄의 낱말, 첫 줄이 없을 때의 판별 기준	지시문 해석 규칙표	① 지시문 해석
입력값	읽어야 할 값, 값의 표기 방식과 찾는 위치	값 추출 스크립트	② 결정적 코드
처리 규칙	확인 순서와 결과를 만드는 방법	진행 스크립트	② 결정적 코드
예외	값이 없거나 후보가 여럿일 때의 대응	정해진 질문 문구, 후보 선택 규칙	① 지시문 해석, ② 결정적 코드
확인 단계	사람이 확인할 단계와 보여 줄 내용	확인 게이트 코드	② 결정적 코드
완료 상태	끝났다고 판정할 시스템 상태	재조회 검증 코드	② 결정적 코드
질문 예시	실제 요청 문장과 기대 결과	회귀 시험 묶음	③ 질문 예시
금지 사항	에이전트가 해서는 안 되는 행동	SKILL.md 금지 항목	② 결정적 코드

규칙표와 스크립트로 옮겨지는 항목의 비중

표에서 보듯이 여덟 항목 가운데 여섯 항목이 결정적 코드로 바뀌고, SKILL.md 본문에 문장으로 남는 것은 명령 한 줄과 금지 사항 정도입니다. 이 비중은 의도한 설계입니다. 스크립트로 옮긴 판단은 운영 모델이 바뀌어도 같은 결과를 내지만, SKILL.md에 문장으로 남긴 판단은 그 문장을 읽는 모델의 수준에 따라 결과가 달라지기 때문입니다. Anthropic도 스킬 안의 코드는 지시문으로 남긴 절차보다 결정적인 신뢰성을 준다고 설명합니다 [S2]. 따라서 업무 전문가가 입력값, 처리 규칙, 예외, 완료 상태를 구체적으로 적을수록 운영 모델이 맡는 판단은 줄어듭니다.

질문 예시 항목이 정의서에서 가장 긴 이유

업무 정의서에서 분량이 가장 많아야 하는 항목은 질문 예시입니다. 업무 전문가는 실제로 들어올 법한 요청을 짧은 질문, 긴 질문, 오타가 섞인 질문, 약칭을 쓴 질문, 두 가지 일을 한 번에 시키는 질문처럼 여러 형태로 적고, 각 질문에 기대 결과를 한 줄씩 붙입니다. Anthropic은 여러 예시 입력으로 모델이 어떤 실수를 하는지 찾아 고치라고 권하며, 질문 예시 목록은 바로 그 시험의 입력이 됩니다[S1]. 따라서 질문 예시가 수십 건이면 프론티어 AI는 그만큼의 회귀 시험을 만들고, 이후 운영 모델을 바꾸거나 업무 규칙이 바뀔 때마다 같은 시험을 다시 실행해 결과가 그대로인지 확인할 수 있습니다.

2.3.2 7.1 조회 업무로 채운 업무 정의서 작성 예

CRM 조회 업무의 업무 정의서 작성 예

아래는 영업 담당자가 채팅으로 CRM 정보를 묻는 조회 업무를 양식에 맞춰 채운 예입니다. 같은 업무가 운영에서 어떻게 바뀌었는지는 7.1에서 다루며, 여기서는 업무 전문가가 무엇을 어느 수준까지 적는지를 보여 주기 위해 고객사 이름과 금액은 빼고 일반 표현으로 적었습니다. 처리 규칙에 적은 조회 의도 일곱 가지도 양식을 채우는 방법을 보이려고 든 예시이며, 의도의 이름과 개수는 조직의 CRM 항목과 자주 들어오는 질문에 맞춰 업무 전문가가 정합니다.

```
 업무 이름   CRM 영업 정보 조회
업무 목적   영업 담당자가 채팅으로 묻는 고객사·영업기회·견적·계약 현황을
          CRM에서 찾아 같은 질문에는 늘 같은 답을 준다
요청 형태   첫 줄이 「조회」이면 이 업무로 처리한다
          첫 줄이 없으면 본문에 고객사 이름과 조회 낱말(현황, 목록, 금액,
          담당자, 견적, 계약)이 함께 있을 때만 이 업무로 본다
입력값      고객사 이름: 정식 이름, 약칭, 영문 이름, 연도가 붙은 사업명 모두 허용
          영업 단계: 발굴, 제안, 협상, 수주, 실주 가운데 하나 이상
          기간: 「올해」「지난 분기」 같은 표현은 날짜 범위로 바꾼다
처리 규칙   1. 질문을 정해 둔 조회 의도 가운데 하나로 분류한다
             (예: 고객사 정보, 영업기회 목록, 영업기회 상세, 견적 현황,
              계약 현황, 담당자별 현황, 최근 활동)
          2. 고객사 이름은 미리 만든 이름 색인에서 찾는다
          3. 의도별로 정해 둔 항목만 표로 보여 준다
예외        고객사 후보가 둘 이상이면 후보 목록을 번호와 함께 보여 주고 고르게 한다
          고객사를 찾지 못하면 「CRM에서 그 이름을 찾지 못했습니다.
          정식 이름이나 사업명을 알려 주세요」라고 묻고 멈춘다
          의도를 분류하지 못하면 조회 의도 목록을 보여 주고 고르게 한다
확인 단계   없음 (조회만 하고 쓰지 않는다)
완료 상태   답에 나온 건수와 CRM 조회 결과 건수가 같다
금지 사항   CRM에 쓰지 않는다. 웹 브라우저로 CRM 화면을 열지 않는다
```

> 스크립트 출력에 없는 값을 덧붙이지 않는다
> 질문 예시 (1) 조회 / A사 진행 중인 영업기회 → 영업기회 목록, A사, 진행 단계
> (2) A사 견적 몇 건 나갔어? → 견적 현황, A사, 전체 기간
> (3) 에이사 담당자 누구야 → 고객사 정보, A사(약칭 일치)
> (4) 올해 수주한 거 담당자별로 → 담당자별 현황, 수주, 올해
> (5) 조회 / B → 후보 둘 이상 → 번호 선택 질문
> … (업무 전문가가 수십 건까지 늘린다)

작성 예에서 업무 전문가가 정한 판단

이 작성 예에서 업무 전문가가 직접 정한 판단은 세 가지입니다. 첫째는 요청 형태 항목의 「첫 줄이 조회이면 이 업무」라는 규칙이고, 이것이 지시문 해석 규칙표가 되어 운영 모델이 스킬을 고르는 부담을 덜어 줍니다. 둘째는 예외 항목의 후보 선택과 되묻기 문구이며, 이 문구는 모델이 지어내지 않고 코드가 그대로 출력합니다. 셋째는 금지 사항의 「웹 브라우저로 CRM 화면을 열지 않는다」로, 작은 모델이 답을 찾지 못할 때 브라우저 도구를 반복해서 부르며 시간을 쓰는 행동을 처음부터 막는 규칙입니다.

작성 예에서 프론티어 AI가 만드는 것

프론티어 AI는 이 정의서를 받아 조회 의도 규칙표, 고객사 이름 색인을 만드는 스크립트, 질문 한 문장을 받아 결과 표를 내는 조회 스크립트, 질문 예시마다 기대 의도와 기대 대상을 묶은 회귀 시험을 만듭니다. 그리고 MCP로 MSAP.ai AAS에 스킬을 올린 뒤 채팅으로 질문 예시를 운영 모델에게 넣어, 회귀 시험 결과와 실제 채팅 답이 같은지 업무 전문가에게 보여 줍니다. 업무 전문가는 그 답을 읽고 정답 여부만 판정하면 되며, 틀린 답이 있으면 정의서의 해당 항목을 고쳐 다시 넘깁니다.

2.4 업무 정의서 한 장에서 나오는 산출물 — 규칙 파일, 결정적 스크립트, 회귀 시험, 짧은 SKILL.md

업무 정의서 한 장은 운영 환경에 올라가기까지 규칙 파일, 결정적 스크립트, 회귀 시험, 짧은 SKILL.md라는 네 종류의 파일 묶음으로 바뀝니다. 이 묶음이 곧 스킬이며, Anthropic Agent Skills와 Hermes Agent는 모두 이 묶음을 SKILL.md를 중심으로 한 디렉터리로 다룹니다[S2][S8]. 파일마다 책임이 다르므로 IT 담당자는 스킬을 받을 때 각 파일이 맡은 일을 알고 있어야 검토와 운영 반입을 판단할 수 있습니다.

산출물	책임	만드는 근거가 된 정의서 항목	실행 주체
규칙 파일	첫 줄 판별, 조회 의도, 정해진 질문 문구, 별칭 목록	요청 형태, 예외	스크립트가 읽음
결정적 스크립트	값 추출, 후보 선택, 순서 진행, 확인 게이트, 쓰기, 재조회 검증	입력값, 처리 규칙, 확인 단계, 완료 상태	스크립트
회귀 시험	질문 예시별 기대 결과 대조	질문 예시	개발 환경과 운영 반입 전
SKILL.md	스킬 이름과 설명, 명령 한 줄, 금지 사항	업무 목적, 금지 사항	운영 모델이 읽음

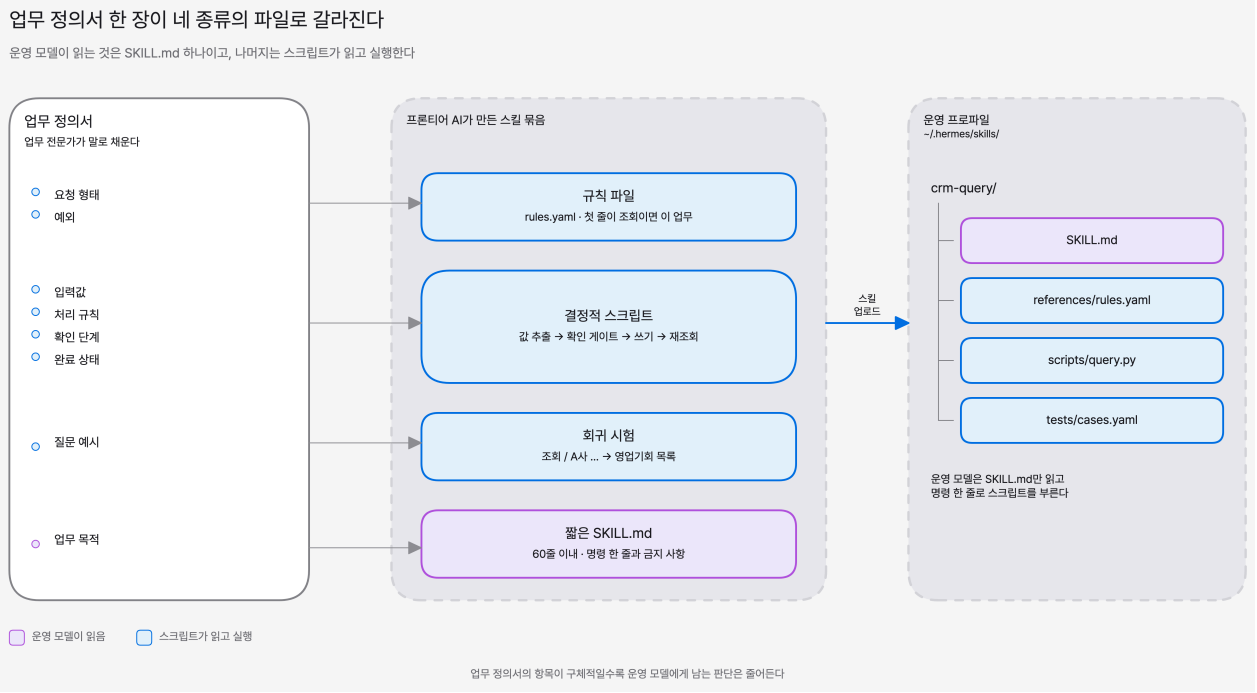


그림 2-1. 업무 전문가의 업무 정의서 항목이 규칙 파일, 결정적 스크립트, 회귀 시험, 짧은 SKILL.md로 바뀌어 운영 프로파일에 배치되는 흐름

2.4.1 SKILL.md 디렉터리 구조와 이름·설명에서 참고 파일까지 열리는 점진적 공개 3단계

SKILL.md를 중심으로 한 스킬 디렉터리 구조

스킬은 SKILL.md 파일을 담은 디렉터리이며, SKILL.md 앞머리의 YAML 블록에는 스킬 이름(name)과 설명(description)이 반드시 있어야 합니다[S2]. YAML은 사람이 읽기 쉬운 설정 파일 형식이고, 여기서는 스킬의 이름표 역할을 합니다. 디렉터리 안에는 SKILL.md 외에 스크립트를 두는 하위 디렉터리, 규칙 파일과 긴 설명을 두는 참고 파일 디렉터리, 회귀 시험 디렉터리를 함께 둘 수 있습니다. Hermes Agent는 스킬을 `~/.hermes/skills/` 아래에 두고 이 디렉터리를 스킬의 기본 저장 위치이자 정본으로 다루며, 같은 SKILL.md 형식을 씁니다[S8]. 따라서 개발 환경에서 프론티어 AI가 만든 스킬 디렉터리는 형식 변환 없이 MSAP.ai AAS의 스킬 업로드로 운영 프로파일에 들어갈 수 있습니다.

점진적 공개 3단계의 동작 방식

점진적 공개(progressive disclosure)는 에이전트가 스킬의 내용을 한꺼번에 읽지 않고 필요한 만큼만 단계적으로 여는 방식입니다. 에이전트는 처음에 모든 스킬의 이름과 설명만 시스템 프롬프트에 올려 두고, 요청과 관련 있어 보이는 스킬이 있으면 그 스킬의 SKILL.md 본문을 읽으며, 본문이 가리키는 참고 파일은 실제로 필요할 때만 엽니다[S2]. Hermes Agent 문서도 먼저 이름과 설명 목록만 보고 필요할 때 본문과 참고 파일을 읽는 같은 구조를 설명합니다[S8]. 아래 표는 세 단계에서 에이전트가 읽는 내용과 그 내용을 채우는 업무 정의서 항목을 함께 보여 줍니다.

단계	에이전트가 읽는 내용	읽는 시점	채우는 업무 정의서 항목
1단계	모든 스킬의 이름과 설명	대화를 시작할 때 늘	업무 목적, 요청 형태

단계	에이전트가 읽는 내용	읽는 시점	채우는 업무 정의서 항목
2단계	고른 스킬의 SKILL.md 본문	요청이 스킬과 관련 있어 보일 때	명령 한 줄, 금지 사항
3단계	참고 파일, 규칙 파일, 스크립트 출력	본문이 가리키고 실제로 필요할 때	입력값, 처리 규칙, 예외

1단계 설명 문장이 스킬 선택을 좌우하는 이유

세 단계 가운데 업무 전문가가 가장 신경 써야 할 단계는 1단계입니다. 에이전트가 요청을 보고 어떤 스킬을 열지 정할 때 참고하는 정보는 이름과 설명뿐이므로, 설명에 요청 첫 줄의 낱말과 업무 범위가 들어 있지 않으면 운영 모델이 엉뚱한 스킬을 열거나 아무 스킬도 열지 않을 수 있습니다. 그래서 업무 정의서의 요청 형태 항목은 SKILL.md 설명 문장의 재료가 되고, 「첫 줄이 조회이면 이 스킬」처럼 판별 기준을 설명 안에 직접 적습니다. 다만 작은 모델은 설명이 정확해도 스킬을 잘못 고를 수 있으므로, MSAP.ai AAS는 첫 줄 판별을 모델에게만 맡기지 않고 모델 호출 전에 코드로 한 번 더 확인하는 방식을 함께 씁니다.

2.4.2 작은 모델의 맥락 부담을 줄이는 짧은 SKILL.md와 참고 파일 분리

짧은 SKILL.md가 작은 모델에게 유리한 이유

SKILL.md 본문은 운영 모델이 요청마다 읽는 문서이므로 길수록 모델의 맥락을 많이 차지하고, 맥락이 길어질수록 작은 모델은 본문 안의 지시를 빠뜨리거나 순서를 바꿔 실행할 가능성이 커집니다. 맥락은 모델이 한 번에 참고하는 입력 전체를 말하며, 활성 파라미터 3B 급 오픈소스 MoE 모델처럼 운영 환경에 두는 작은 모델은 긴 지시를 끝까지 일관되게 따르는 능력이 프론티어 모델보다 약합니다. 반면 스크립트 안의 판단은 모델의 맥락에 올라가지 않고도 실행되므로, 같은 판단을 본문 문장에서 스크립트로 옮기면 모델의 부담은 줄고 결과는 일정해집니다[S2]. 그래서 MSAP.ai AAS용 스킬은 SKILL.md 본문을 60줄 이내로 두고, 그 안에 명령 한 줄과 금지 사항만 적는 것을 작성 기준으로 삼습니다.

본문과 참고 파일을 나누는 기준

본문과 참고 파일을 나누는 기준은 운영 모델이 그 내용을 매번 읽어야 하는지 여부입니다. 운영 모델이 요청마다 알아야 하는 것은 어떤 명령을 부를지와 무엇을 하면 안 되는지뿐이므로 이 두 가지만 본문에 두고, 나머지는 참고 파일이나 스크립트로 옮깁니다. 아래 표는 업무 정의서의 내용이 본문과 참고 파일, 스크립트 가운데 어디로 가는지를 정리한 것입니다.

내용	둘 곳	이유
부를 명령 한 줄	SKILL.md 본문	운영 모델이 요청마다 실행한다
금지 사항	SKILL.md 본문	운영 모델의 행동을 직접 제한한다
조회 의도 목록, 별칭 목록, 정해진 질문 문구	규칙 파일	스크립트가 읽고 모델은 읽지 않는다
값 추출, 후보 선택, 확인 게이트, 재조회 검증	결정적 스크립트	같은 입력에 같은 결과가 나와야 한다

내용	둘 곳	이유
업무 배경 설명, 예외의 상세 사례	참고 파일	사람이나 프론티어 AI가 고칠 때만 읽는다
질문 예시와 기대 결과	회귀 시험	스킬을 고칠 때마다 다시 실행한다

운영 중 스킬을 지키는 파일 분리의 효과

본문과 참고 파일을 나누어 두면 운영 중에 스킬이 바뀌는 문제도 관리하기 쉬워집니다. Hermes Agent는 에이전트가 스킬 관리 도구로 스킬을 직접 만들고 고치고 지울 수 있게 하며, 선택 설정으로 그 쓰기를 사람 검토 대기로 돌릴 수 있게 합니다[S8]. 짧은 SKILL.md와 분리된 스크립트·규칙 파일을 쓰면 업무 판단이 대부분 스크립트와 규칙 파일에 있으므로, 운영 모델이 SKILL.md를 고쳐 쓰더라도 업무 결과가 바뀌는 범위가 좁고 원본과 비교해 되돌리기도 쉽습니다. 결국 업무 정의서 한 장이 만들어 내는 이 파일 구조는 작은 모델이 같은 결과를 내도록 돕는 장치이면서, IT 담당자가 운영 중인 스킬을 검토하고 관리하는 단위가 됩니다. 업무 전문가가 정의서를 쓰고 프론티어 AI가 이 구조로 옮긴 스킬을 개발 환경에서 운영 환경으로 들여오는 경로는 폐쇄망 조직의 보안 요건과 직접 맞닿아 있으며, 그 경로가 3장의 주제입니다.

스킬이 늘어나도 입력이 일정하게 유지되는 읽기 순서

Hermes Agent의 스킬 도구는 이 점진적 공개를 함수 단위로 나눠 둡니다. 에이전트는 먼저 스킬 목록 함수(`skills_list()`)로 전체 스킬의 이름과 설명만 약 3천 토큰 범위에서 받고, 쓰기로 정한 스킬만 본문 조회 함수(`skill_view(name)`)로 읽으며, 참고 파일은 경로를 지정한 조회(`skill_view(name, path)`)로 필요할 때만 가져옵니다. 그래서 한 프로파일에 스킬을 수십 개 설치해도 요청마다 모든 본문이 모델 입력에 실리지 않고, 입력 토큰은 실제로 고른 스킬의 본문 길이에 따라 늘어납니다. 목록 단계의 부담은 이미 고정돼 있으므로, 업무 전문가가 줄일 수 있는 것은 고른 뒤에 읽히는 본문뿐이라는 점에서 짧은 SKILL.md의 이점이 한 번 더 드러납니다.

3장: 만드는 곳과 일하는 곳의 분리

MSAP.ai AAS는 AI Agent의 스킬을 만드는 개발 환경과 그 스킬로 실제 업무를 처리하는 운영 환경을 나눠 두는 구성을 전제로 설계했습니다. 인터넷이 되는 개발 환경에서는 Claude 같은 프론티어 AI가 MCP(Model Context Protocol, LLM 애플리케이션이 외부 자료와 도구를 부르는 개방형 표준 규약)로 MSAP.ai AAS에 연결해 스킬을 만들고, 폐쇄망 운영 환경에서는 사내 게이트웨이인 LiteLLM 뒤에 놓인 로컬 LLM이 그 스킬을 실행합니다[S3][S9]. 공공·금융처럼 망 분리 규정을 지키는 조직은 고객 정보나 금액 자료를 외부 AI 서비스로 보낼 수 없으므로, 가장 성능이 좋은 모델을 스킬 제작에 쓰고 싶어도 「외부 AI를 쓴다」는 사실 하나가 보안 검토의 첫 장벽이 됩니다. 이 우려는 실제 규정 위반 가능성에서 나온 정당한 우려이므로, 도입 검토는 무엇이 밖에 있고 무엇이 안에 있는지를 구성도로 확인하는 일부터 시작해야 합니다.

두 환경을 나누면 이 질문에 구조로 답할 수 있습니다. 개발 환경에서 프론티어 AI가 다루는 자료는 업무 전문가가 쓴 업무 정의서와 비식별 시험 자료뿐이고, 두 환경 사이를 넘어가는 것은 업무 규칙과 결정적 코드를 담은 스킬 파일 묶음 하나이며, 고객 자료와 운영 모델 호출은 처음부터 끝까지 폐쇄망 안에 남습니다. 따라서 의사결정권자가 이 장에서 판단할 것은 운영 데이터를 밖으로 내보내지 않고도 프론티어 AI를 스킬 제작에 쓸 수 있는가 하나입니다. 이 장에서는 두 환경에 놓이는 모델·데이터·도구를 비교하고, 실데이터 없이 스킬을 시험하는 디지털 트윈 개발 환경, 완성 스킬의 반입 절차, 운영 환경의 모델 호출 경로를 차례로 정리했습니다. 보안 담당자가 승인 전에 확인할 항목은 3.4절의 보안 점검표에 모았으므로, 그 표 하나로 현재 망 분리 정책 안에서 파일럿을 시작할 수 있는지를 판단할 수 있습니다.

3.1 인터넷이 되는 개발 환경의 프론티어 AI와 폐쇄망 운영 환경의 로컬 LLM 역할 구분

개발 환경과 운영 환경은 같은 MSAP.ai AAS와 오픈소스 Hermes Agent를 쓰지만, 각 환경에 연결된 모델과 데이터는 서로 다릅니다. 개발 환경의 모델은 업무 정의서를 읽고 규칙 파일·결정적 스크립트·회귀 시험을 작성하는 프론티어 AI이고, 운영 환경의 모델은 완성된 스킬의 명령을 한 줄씩 실행하는 활성 파라미터 3B 급 오픈소스 MoE(Mixture of Experts, 전문가 혼합) 모델입니다. 에이전트 안에서 언어 모델이 하는 일은 적은 수의 전문 작업을 거의 변화 없이 반복하는 것이므로 작은 모델로도 대부분의 호출을 감당할 수 있고, 범용 대화 능력이 꼭 필요한 일에만 큰 모델을 섞는 이종 구성이 자연스럽다는 주장이 공개돼 있습니다[S4]. 다만 이 주장은 실험 결과가 아니라 입장 논문의 견해이므로, 작은 모델이 같은 결과를 내는지는 MSAP.ai AAS 채팅으로 운영 모델에 직접 요청을 넣어 확인해야 합니다.

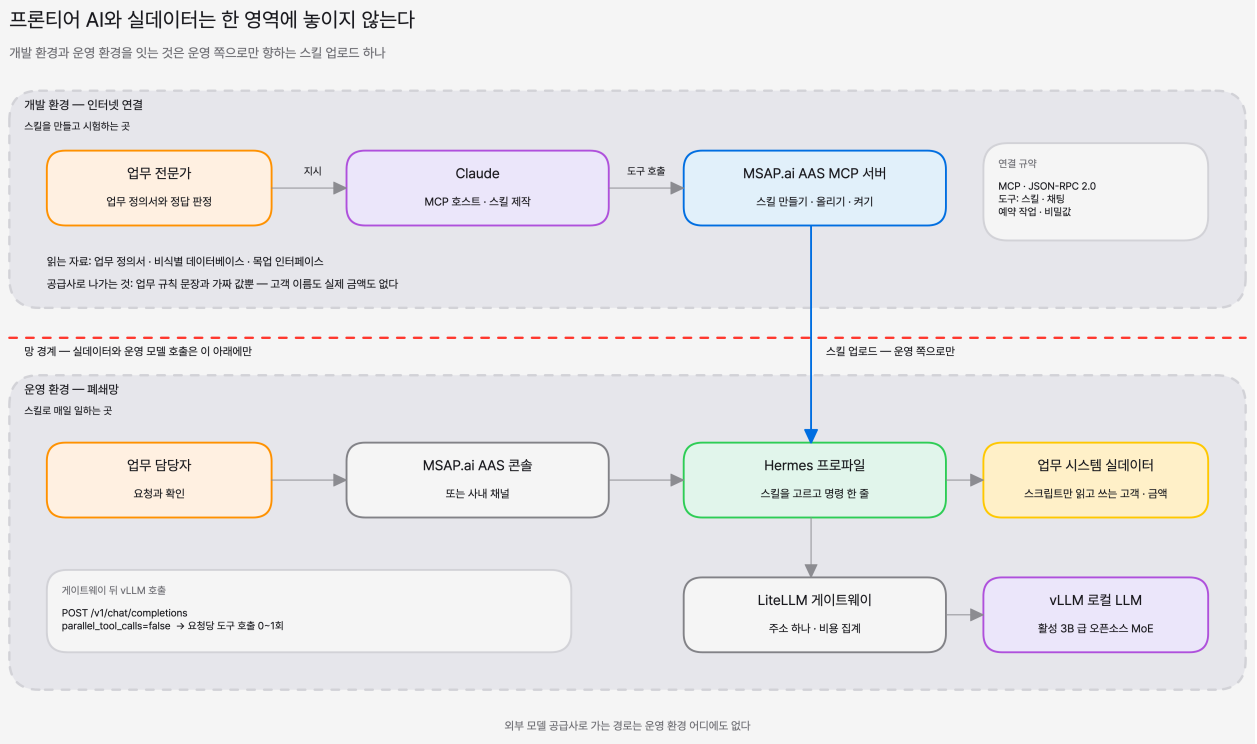


그림 3-1. 개발 환경과 운영 환경의 AI Agent 구성. 개발 환경에서는 업무 전문가의 지시를 받은 Claude가 MCP로 MSAP.ai AAS MCP 서버를 부르고, 운영 환경에서는 MSAP.ai AAS 콘솔의 Hermes 프로파일이 LiteLLM을 거쳐 vLLM으로 서빙하는 로컬 LLM을 부른다.

그림 3-1의 왼쪽은 인터넷이 되는 개발 환경입니다. 업무 전문가가 Claude에게 업무 정의서를 건네면 Claude가 MCP 호스트로서 MSAP.ai AAS MCP 서버의 도구를 불러 스킬을 만들고 올리고 켜며, 시험 요청은 같은 MCP 연결로 MSAP.ai AAS 채팅에 보냅니다. 오른쪽은 폐쇄망 운영 환경으로, 업무 담당자의 요청이 MSAP.ai AAS 콘솔이나 사내 채널로 들어오면 Hermes 프로파일이 스킬을 고르고 모델 호출은 LiteLLM을 거쳐 vLLM 로컬 LLM에 닿으며, 업무 시스템의 실데이터는 이 경계 안에서만 읽고 씁니다. 두 영역 사이를 잇는 화살표는 스킬 업로드 하나뿐이고, 그 화살표는 운영 환경 쪽으로만 향합니다. 그림에서 의사결정권자가 확인할 것은 프론티어 AI와 실데이터가 같은 영역에 한 번도 함께 놓이지 않는다는 사실입니다.

3.1.1 모델과 데이터, 도구가 놓이는 자리로 비교한 두 환경의 데이터 경계

개발 환경의 프론티어 AI와 비식별 자료

개발 환경은 프론티어 AI가 스킬을 만드는 데 필요한 것만 갖춘 별도 영역입니다. MCP에서 연결을 거는 쪽을 호스트, 기능을 내놓는 쪽을 서버라고 부르는데[S3], 개발 환경의 프론티어 AI는 호스트로 동작해 MSAP.ai AAS가 서버로 내놓는 도구, 즉 에이전트 목록 조회·스킬 만들기·스킬 올리기·켜고 끄기·채팅·예약 작업·비밀값 관리를 부릅니다. 이때 프론티어 AI가 읽는 자료는 업무 전문가가 쓴 업무 정의서, 비식별 데이터베이스의 시험 값, 목업 인터페이스의 응답 형식에 한정됩니다. 개발 환경에서 프론티어 AI 공급사로 나가는 내용은 업무 규칙을 설명한 문장과 가짜 값으로 채운 시험 자료이며, 그 안에는 고객 이름도 실제 금액도 없습니다.

운영 환경의 로컬 LLM과 실데이터

운영 환경은 폐쇄망 안에 MSAP.ai AAS, Hermes Agent, LiteLLM 게이트웨이, vLLM 서빙 서버를 모두 두고 외부 모델 공급사로 가는 경로를 만들지 않는 영역입니다. 업무 시스템의 고객 자료와 금액 자료는 스킬 안의

결정적 스크립트가 직접 읽고 쓰며, 모델은 어떤 명령을 실행할지 고르고 결과를 사람에게 전달하는 일만 맡습니다. 예를 들어 Qwen3-30B-A3B는 전체 30.5B 파라미터 가운데 토큰마다 3.3B만 계산에 쓰고 Apache-2.0 라이선스로 공개된 모델인데, 운영 환경의 모델은 이처럼 활성 파라미터가 3B 급인 오픈소스 MoE 범주에서 고릅니다[S7]. 운영 환경은 보급형 GPU에서도 돌 수 있는 작은 모델과 실데이터를 같은 망 안에 두는 구성이며, 이 모델을 고르는 기준은 6장에서 다룹니다.

환경별 구성 요소 비교

두 환경의 차이는 모델·데이터·도구·사람·비밀값 다섯 항목에서 모두 나타납니다. 아래 표에서 가운데 열과 오른쪽 열에 같은 실데이터가 동시에 나오는 행이 하나도 없다는 점이 이 구성의 핵심입니다.

항목	개발 환경 (인터넷 연결)	운영 환경 (폐쇄망)
모델	Claude 등 프론티어 AI — 스킬 제작	활성 파라미터 3B 급 오픈소스 MoE 모델 — vLLM 서빙
모델 호출 경로	프론티어 AI(MCP 호스트) → MSAP.ai AAS MCP 서버	Hermes 프로파일 → LiteLLM → vLLM
데이터	비식별 데이터베이스, 목업 인터페이스	업무 시스템의 고객·금액 실데이터
도구	스킬 만들기·올리기·켜고 끄기, 채팅, 예약 작업, 비밀값 관리	스킬 안의 결정적 스크립트, 업무 시스템 연결 도구
사람	업무 전문가 — 업무 정의와 정답 판정	업무 담당자 — 요청과 확인, IT 담당자 — 운영 관리
비밀값	개발용 시험 계정	운영 비밀값 관리에 따로 등록
남는 결과	스킬 파일 묶음, 시험 기록	업무 처리 결과, 모델 호출 기록

표 3-1. 개발 환경과 운영 환경의 구성 요소 비교 (MCP 역할은 [S3], 게이트웨이 구성은 [S9] 기준)

망 분리 정책과 두 환경의 관계

이 구성은 망 분리 정책을 바꾸지 않습니다. 개발 환경은 조직이 이미 운영하는 인터넷 영역의 개발 시스템과 같은 등급으로 다루고, 운영 환경은 기존 폐쇄망 업무 시스템과 같은 등급으로 다루면 됩니다. 두 망 사이에서 움직이는 것은 소스 코드 반입과 같은 성격의 파일 묶음이므로, 조직이 이미 쓰는 자료 반입 승인 절차를 그대로 적용할 수 있습니다. 따라서 도입 결정에 필요한 정책 변경은 없고, 보안 부서가 새로 검토할 대상은 반입되는 스킬 파일의 내용과 운영 환경 안의 모델 호출 통제 두 가지로 좁혀집니다.

> 🎯 *결정 포인트 — 프론티어 AI는 업무 정의서와 비식별 시험 자료만 보고, 실데이터는 폐쇄망 안의 로컬 LLM과 결정적 스크립트만 다룹니다. 이 경계가 우리 조직의 망 분리 규정과 맞는지가 파일럿 승인의 첫 판단 기준입니다.*

3.1.2 개발 환경에서 운영 환경으로 넘어가는 것은 스킬뿐인 구성

스킬 파일 묶음의 구성

두 환경 사이를 넘는 유일한 산출물은 스킬이며, 스킬은 SKILL.md 파일을 담은 디렉터리입니다[S2]. Hermes Agent는 이 디렉터리를 정해진 스킬 디렉터리에 두고 그곳을 스킬의 정본으로 취급하므로, 반입은 디렉터리 하나를 운영 환경의 같은 위치에 옮기는 일과 같습니다[S8]. 묶음 안에는 SKILL.md 말고도 업무 전문가의 판단 기준을 옮긴 규칙 파일, 값 추출과 확인 게이트를 맡는 결정적 스크립트, 질문 예시를 회귀 시험으로 고정한 시험 파일이 함께 들어갑니다. 스크립트를 넣은 스킬은 코드만 줄 수 있는 결정적 신뢰성을 얻는데, 같은 입력에 늘 같은 결과를 내는 이 성질 덕분에 개발 환경에서 통과한 시험을 운영 환경에서도 같은 기준으로 다시 돌릴 수 있습니다[S2].

경계 통과 항목과 남는 항목

경계를 넘는 항목과 넘지 않는 항목은 미리 목록으로 정해 두어야 반입 심사가 빨라집니다. 아래 표는 MSAP.ai AAS 구성에서 반입 대상과 비반입 대상을 나눈 기준입니다.

항목	경계 통과	이유
SKILL.md (이름·설명·짧은 실행 지시)	통과	운영 모델이 읽는 스킬의 입구
규칙 파일 (판단 기준·질문 예시)	통과	업무 전문가가 정한 기준을 코드가 읽는 형태로 옮긴 것
결정적 스크립트·플러그인 훅	통과	값 추출·확인 게이트·쓰기·재조회 검증을 실행
회귀 시험 파일	통과	반입 뒤 운영 모델로 같은 시험을 다시 돌리기 위함
비식별 시험 자료	통과하지 않음	개발 환경 전용, 운영 환경에는 실데이터가 있음
운영 비밀값 (업무 시스템 계정·키)	통과하지 않음	운영 환경의 비밀값 관리에 따로 등록
고객·금액 실데이터, 모델 호출 기록	통과하지 않음	폐쇄망 밖으로 나가지 않음

표 3-2. 개발 환경과 운영 환경 사이의 경계 통과 항목

운영에서 발견한 실패를 개발 환경으로 되돌리는 방법

운영 중에 스킬이 틀린 답을 내면 그 원인을 고치는 작업은 다시 개발 환경에서 이뤄지는데, 이때도 실데이터는 밖으로 나가지 않아야 합니다. 업무 전문가는 실패한 요청을 원문 그대로 옮기지 않고, 고객 이름과 금액을 지운 질문 예시로 다시 적어 개발 환경의 회귀 시험에 추가합니다. 예를 들어 「특정 고객사의 견적 금액을 물었는데 다른 영업기회를 골랐다」는 실패는 「같은 이름의 영업기회가 둘일 때 되묻지 않고 첫 번째를 골랐다」는 비식별 문장으로 바뀝니다. 운영에서 개발로 되돌아가는 것은 업무 전문가가 정리한 비식별 문장이며, 이 문장이 다음 판 스킬의 질문 예시가 됩니다.

3.2 비식별 데이터베이스와 목업 인터페이스로 내부 시스템을 흉내 낸 디지털 트윈 개발 환경

디지털 트윈 개발 환경은 운영 업무 시스템과 같은 모양의 데이터와 인터페이스를 비식별 값으로 다시 만든 개발용 복제 환경입니다. 프론티어 AI가 스킬을 만들려면 실제로 호출해 볼 대상이 있어야 하는데, 운영 시스템을 인

터넷 쪽으로 열 수 없으므로 같은 응답을 돌려주는 대역을 개발 환경에 세워 두는 것입니다. 도구 설명에 사용 예·경계 사례·입력 형식을 담고 여러 예시 입력으로 시험해 모델의 실수를 찾아 고치라는 설계 지침이 있는데, 디지털 트윈은 그 시험을 실데이터 없이 반복하게 해 주는 장치입니다[S1]. 따라서 파일럿 전에 IT 담당자가 정할 범위는 시험할 업무가 실제로 읽고 쓰는 테이블과 인터페이스가 어디까지인가입니다.

3.2.1 실데이터 없이 스킬을 만들고 시험하는 디지털 트윈 구성 요소

비식별 데이터베이스가 재현하는 운영 데이터의 형태

비식별 데이터베이스는 운영 데이터베이스의 테이블 구조와 컬럼 이름, 값의 형식을 그대로 두고 값만 가짜로 바꾼 데이터베이스입니다. 고객사 이름은 「가나다 기관」 같은 임의 이름으로, 금액은 실제 분포를 흉내 낸 임의 숫자로 바꾸되, 같은 이름의 고객이 둘인 경우나 값이 비어 있는 행처럼 스킬이 부딪힐 예외 상황은 일부러 남겨 둡니다. 비식별 데이터베이스는 단순히 개인정보를 지운 사본보다 한 단계 더 손이 갑니다. 예외 상황이 빠진 시험 자료로 만든 스킬은 개발 환경에서는 모두 통과하고 운영 환경에서 처음 보는 경우에 막히기 때문입니다.

목업 인터페이스가 재현하는 업무 시스템의 응답

목업 인터페이스는 CRM·결재·자산 관리 같은 내부 업무 시스템의 호출 규약을 흉내 내 정해진 응답을 돌려주는 가짜 서버입니다. 스킬의 결정적 스크립트는 운영 환경에서 부를 인터페이스와 똑같은 주소 형식·인자·응답 구조로 목업을 호출하므로, 스크립트 코드는 접속 대상 설정만 바꾸면 운영 환경에서 그대로 돕니다. 이때 목업이 돌려주는 오류 응답과 빈 결과도 운영 시스템과 같아야 하는데, 이 차이를 줄이는 준비는 3.2.2에서 점검표로 다룹니다. 목업 인터페이스는 스크립트가 운영 환경에서 받을 응답을 개발 단계에서 미리 받아 보게 하는 시험 대상입니다.

컨테이너로 띄우는 개발용 MSAP.ai AAS와 Hermes Agent

개발 환경의 MSAP.ai AAS와 Hermes Agent는 컨테이너로 띄워 운영 환경과 같은 판을 맞춥니다. 프론티어 AI는 MCP로 이 개발용 MSAP.ai AAS에 연결해 스킬을 만들고, 같은 연결로 MSAP.ai AAS 채팅에 시험 요청을 보내 개발용 에이전트가 실제로 스킬을 실행하는지 확인합니다. 개발용 에이전트가 쓰는 모델도 운영 환경과 같은 활성 3B 급 오픈소스 MoE 모델로 맞추면, 시험 결과가 운영 환경의 결과를 미리 보여 주는 신호가 됩니다. 반면 개발용 에이전트에 프론티어 AI를 연결해 시험하면 시험은 쉽게 통과하지만 운영 모델에서 같은 결과가 나온다는 보장이 사라집니다.

지금까지 설명한 네 구성 요소를 흉내 내는 대상과 만드는 방법으로 정리하면 아래 표와 같습니다. 네 요소 모두 실데이터를 담지 않는다는 점이 개발 환경을 인터넷 쪽에 둘 수 있는 조건입니다.

구성 요소	흉내 내는 대상	만드는 방법	실데이터 포함
비식별 데이터베이스	운영 데이터베이스의 테이블 구조·값 형식·예외 행	구조는 복제, 값은 임의 생성, 예외 행은 의도적으로 추가	없음
목업 인터페이스	업무 시스템의 호출 규약·정상 응답·오류 응답	운영 인터페이스 명세로 가짜 서버 작성	없음
개발용 MSAP.ai AAS·Hermes Agent	운영 환경의 콘솔·에이전트 프로파일	운영 환경과 같은 판을 컨테이너로 실행	없음

구성 요소	흉내 내는 대상	만드는 방법	실데이터 포함
개발용 실행 모델	운영 환경의 로컬 LLM	같은 범주의 활성 3B 급 MoE 모델 연결	없음

표 3-3. 디지털 트윈 개발 환경의 구성 요소

3.2.2 디지털 트윈과 운영 시스템의 인터페이스 차이를 줄이는 준비 항목

인터페이스 차이가 운영 환경에서 드러나는 방식

디지털 트윈과 운영 시스템의 차이는 대부분 필드 이름이나 응답 형식처럼 작은 데서 생기고, 그 차이는 반입 뒤 첫 시험에서 오류로 드러납니다. 상대 경로 대신 절대 경로를 요구하도록 도구 인자를 바꾸자 모델의 경로 오류가 사라졌다는 사례처럼, 모델의 실수는 모델 능력보다 도구가 받는 입력 형식이 모호할 때 더 자주 나옵니다 [S1]. 따라서 목업 인터페이스를 만들 때는 정상 응답보다 경계 상황의 응답을 운영과 똑같이 맞추는 데 시간을 더 써야 합니다. 아래 점검표는 IT 담당자가 파일럿 전에 운영 인터페이스 명세와 목업을 나란히 놓고 확인할 항목입니다.

준비 항목	맞출 내용	어긋나면 운영 환경에서 나타나는 증상
필드 이름	대소문자·밑줄·한글 필드 이름까지 운영과 동일	스크립트가 값을 찾지 못해 빈 결과 반환
응답 형식	목록 감싸는 구조, 빈 값 표기(null·빈 문자열·필드 생략)	결과가 있는데도 「없음」으로 판정
오류 응답	권한 없음·대상 없음·중복의 응답 코드와 메시지	오류를 정상으로 보고 다음 단계로 진행
날짜·금액 표기	날짜 형식, 천 단위 구분, 부가세 포함 여부	금액 계산과 비교 결과가 틀림
한글 처리	한글 이름 검색의 공백·괄호·약칭 처리	같은 고객을 다른 고객으로 인식
건수 상한	페이지 나눔 방식과 한 번에 돌려주는 최대 건수	일부 결과만 보고 판단
인증 방식	값은 다르되 인증 절차와 만료 응답은 동일	반입 직후 모든 호출이 인증 오류

표 3-4. 디지털 트윈과 운영 시스템의 인터페이스 차이를 줄이는 준비 항목 점검표

일곱 항목 가운데 필드 이름과 오류 응답은 명세서를 나란히 놓으면 쉽게 찾지만, 날짜·금액 표기와 한글 처리, 건수 상한은 명세서에 잘 드러나지 않아 놓치기 쉽습니다. 부가세 포함 여부가 다르면 스크립트는 오류 없이 틀린 금액을 비교하고, 한 번에 돌려주는 건수가 목업보다 적으면 스크립트는 일부 결과만 보고 「대상 없음」이라고 답합니다. 두 경우 모두 오류 메시지가 나오지 않으므로, 이 세 항목은 운영 시스템에서 실제 응답 표본을 받아 목업 응답과 글자 단위로 대조하는 편이 안전합니다.

파일럿 범위에 맞춘 디지털 트윈 준비량

디지털 트윈을 운영 시스템 전체 규모로 만들 필요는 없습니다. 파일럿 대상 업무가 하나라면 그 업무가 부르는 인터페이스 몇 개와 테이블 몇 개만 흉내 내면 되고, 업무를 늘릴 때마다 해당 부분을 덧붙이면 됩니다. 다만 위 점검표의 일곱 항목은 업무 크기와 상관없이 모두 확인해야 하는데, 한 항목이라도 어긋나면 개발 환경의 합격 판정이 운영 환경에서 뒤집히기 때문입니다. 그래서 IT 담당자의 파일럿 준비 범위는 「대상 업무가 쓰는 인터페이스 목록 + 그 인터페이스의 점검표 일곱 항목」으로 정하는 것이 가장 작고 정확합니다.

3.3 스킬 업로드와 스킬 허브를 통한 폐쇄망 반입 경로

완성된 스킬은 MSAP.ai AAS의 스킬 업로드나 폐쇄망 안의 스킬 허브를 거쳐 운영 환경에 들어갑니다. 스킬 업로드는 콘솔에서 스킬 디렉터리 묶음을 직접 올려 특정 에이전트에 켜는 경로이고, 스킬 허브는 검토를 마친 스킬을 조직 내부 저장소에 게시한 뒤 필요한 에이전트가 설치해 가는 경로입니다. Hermes Agent의 스킬 허브는 공식 선택 스킬이나 GitHub 저장소 같은 원천에서 스킬을 찾아 설치하며 설치할 때마다 보안 검사를 거치므로, 반입 경로 자체에 검사 단계가 들어 있습니다[S8]. 보안 부서가 승인하려면 이 경로가 문서로 정해져 있어야 하므로, 반입 단계와 단계별 담당·기록을 아래처럼 미리 고정합니다.

3.3.1 완성 스킬의 반입 절차와 설치 때 거치는 보안 검사

스킬 업로드 경로와 스킬 허브 경로의 쓰임

두 경로는 반입하는 스킬의 성격에 따라 나눠 씁니다. 한 부서의 에이전트 하나에만 쓰는 스킬을 파일럿으로 시험할 때는 스킬 업로드가 빠르고, 여러 부서의 에이전트가 같은 스킬을 쓰거나 판을 관리해야 할 때는 스킬 허브가 알맞습니다. 스킬 허브를 쓰면 어떤 에이전트가 어느 판의 스킬을 설치했는지가 저장소에 남으므로, 문제가 생긴 판을 한 번에 찾아 되돌리기 쉽습니다. 파일럿 단계는 스킬 업로드로 시작하고, 같은 스킬을 두 번째 부서가 쓰기 시작할 때 스킬 허브로 옮기는 순서가 자연스럽습니다.

반입 단계와 단계별 기록

반입 절차는 개발 환경의 시험 통과에서 시작해 운영 환경의 재시험으로 끝납니다. 개발 환경에서 이미 시험을 통과했더라도 운영 환경의 실데이터와 실제 인터페이스로 같은 시험을 다시 돌려야 3.2.2의 차이가 남아 있는지 확인할 수 있습니다. 아래 표의 여섯 단계는 MSAP.ai AAS 반입에서 쓰는 순서이며, 남는 기록 열은 보안 부서가 사후 감사 때 요구할 증빙입니다.

단계	하는 일	담당	남는 기록
1. 개발 환경 시험	회귀 시험과 질문 예시 전체를 개발용 실행 모델로 통과	업무 전문가	시험 결과
2. 묶음 확정	스킬 디렉터리를 묶고 판 번호와 파일 해시를 기록	IT 담당자	판 번호·해시
3. 반입 승인	조직의 자료 반입 절차로 파일 묶음 승인	보안 담당자	반입 승인서
4. 업로드·게시	운영 MSAP.ai AAS에 스킬 업로드 또는 스킬 허브	IT 담당자	업로드 기록

단계	하는 일	담당	남는 기록
	게시		
5. 설치 검사	스킬 허브 경로로 설치할 때 보안 검사 실행[S8]	시스템	검사 결과
6. 운영 재시험	시험 대상 에이전트에 켜고 같은 회귀 시험을 운영 모델로 실행	업무 전문가	시험 결과

표 3-5. 완성 스킬의 폐쇄망 반입 단계표

반입 뒤 운영 재시험의 판정 기준

6단계의 운영 재시험은 개발 환경 시험과 같은 문항을 같은 합격 기준으로 판정합니다. 결과가 달라지면 원인은 대부분 디지털 트윈과 운영 시스템의 인터페이스 차이이므로, 스킬을 운영 환경에서 직접 고치지 않고 차이를 목업에 반영한 뒤 개발 환경에서 다시 만들어 반입합니다. 이렇게 하면 운영 환경의 스킬은 언제나 개발 환경의 특정 판과 해시가 일치하는 상태로 유지됩니다. 따라서 보안 담당자는 운영 환경의 스킬 파일 해시를 반입 승인서의 해시와 대조하는 것만으로 승인되지 않은 변경이 있는지 확인할 수 있습니다.

3.3.2 스킬을 부르는 스킬과 한글 이름 스킬의 반입 후 동작 유지

여러 스킬이 엮인 구조와 큰 스킬의 반입

업무가 커지면 하나의 스킬이 다른 스킬을 부르거나, 규칙 파일과 스크립트가 수십 개인 큰 스킬이 생깁니다. MSAP.ai AAS는 스킬을 부르는 스킬, 참고 파일이 많은 큰 스킬, 한글 이름 스킬을 운영 환경으로 옮겨 LiteLLM 게이트웨이 뒤의 로컬 모델로 실행해도 그대로 동작합니다. 큰 스킬이 작은 모델에 부담이 되지 않는 이유는 점진적 공개 구조에 있는데, 모델은 먼저 스킬의 이름과 설명만 보고 관련이 있을 때 SKILL.md 본문을, 그 안에서 필요할 때만 참고 파일을 엽니다[S2][S8]. 스킬 묶음의 크기가 커져도 한 번의 요청에서 모델이 읽는 분량은 필요한 부분으로 한정됩니다.

한글 이름 스킬과 한글 첫 줄 판별

업무 전문가는 스킬 이름을 「견적」·「영업기회」처럼 자기 업무 용어로 붙이는 경우가 많고, 요청 첫 줄의 한글 낱말로 어느 스킬을 부를지 정하는 규칙도 흔히 씁니다. 그래서 반입 때는 스킬 이름과 파일 이름의 한글이 운영 환경에서 같은 글자로 읽히는지, 그리고 첫 줄 판별 규칙이 같은 결과를 내는지를 확인 항목에 넣습니다. 한글 파일 이름은 운영체제와 압축 도구에 따라 같은 글자가 다른 바이트로 저장될 수 있으므로, 2단계에서 기록한 해시를 운영 환경에서 다시 계산해 대조하면 이 문제를 반입 시점에 찾을 수 있습니다.

반입 대상 유형	반입 때 확인할 것	반입 뒤 동작
스킬을 부르는 스킬	불리는 스킬이 같은 에이전트에 함께 켜져 있는가	그대로 동작
큰 스킬 (참고 파일·스크립트 다수)	참고 파일 경로가 스킬 디렉터리 안의 상대 위치로만 적혀 있는가	그대로 동작, 필요할 때만 참고 파일을 엶[S2]

반입 대상 유형	반입 때 확인할 것	반입 뒤 동작
한글 이름 스킬	스킬 이름·파일 이름이 같은 글자로 읽히는가, 해시가 일치하는가	그대로 동작
첫 줄 판별 규칙	첫 줄 낱말 목록이 규칙 파일에 들어 있는가	같은 질문 예시에 같은 스킬 선택

표 3-6. 반입 대상 유형별 동작 확인표

3.4 고객·금액 자료를 밖으로 내보내지 않는 사내 게이트웨이와 보안 담당자의 확인 지점

운영 환경에서 일어나는 모든 모델 호출은 OpenAI 호환 사내 게이트웨이 LiteLLM 한 곳을 거쳐 vLLM으로 서빙하는 로컬 LLM에 닿습니다. 호출이 한 곳으로 모이면 보안 부서는 어떤 에이전트가 어떤 모델을 얼마나 불렀는지를 그 한 곳에서 보고, 외부로 가는 경로가 없다는 사실도 게이트웨이 설정 하나로 확인할 수 있습니다. 여기에 MCP 명세가 요구하는 사용자 동의 원칙과 Hermes Agent의 스킬 쓰기 승인 설정을 더하면, 「외부 AI를 쓴다」는 우려에 대한 답이 구조와 통제 항목으로 정리됩니다[S3][S8][S9].

3.4.1 OpenAI 호환 LiteLLM 게이트웨이 하나로 묶는 로컬 LLM 호출 경로

LiteLLM 게이트웨이로 모이는 모델 호출

LiteLLM은 자체 구축형 서버로 100개가 넘는 LLM 제공자, MCP 도구, A2A 에이전트를 OpenAI 호환 인터페이스 하나로 묶는 오픈소스 게이트웨이입니다. 사용자나 팀마다 가상 키를 발급해 쓸 수 있는 모델·예산·속도 상한을 정하고, 비용을 집계하며, 여러 배포 사이에서 부하를 나누고 한 배포가 멈추면 다른 배포로 우회하거나 재시도하고, 호출 기록을 관측 도구로 보냅니다[S9]. MSAP.ai AAS 운영 환경에서는 이 게이트웨이에 폐쇄망 안의 vLLM 배포만 등록하므로, 게이트웨이의 모델 목록 자체가 외부 제공자로 가는 경로가 없다는 증빙이 됩니다. LiteLLM이 지원하는 외부 제공자 연결 기능은 쓰지 않고, 가상 키·예산·기록·장애 우회 기능만 사내 모델 앞에서 씁니다.

vLLM이 내놓는 OpenAI 호환 인터페이스와 도구 호출

vLLM은 `/v1/completions`, `/v1/chat/completions` 같은 OpenAI 호환 경로를 내놓는 모델 서빙 서버이며, 채팅 요청에서 도구 호출을 지원합니다[S10]. 요청에 `parallel_tool_calls=false`를 주면 요청 하나에서 도구 호출이 0회나 1회로 제한됩니다[S10]. 판단은 스크립트가 하고 모델은 명령 한 줄을 고르는 MSAP.ai AAS 스킬 구조에서는 이 제한이 오히려 실행 결과를 예측하기 쉽게 만듭니다. 그래서 Hermes 프로파일은 vLLM을 직접 부르지 않고 LiteLLM의 주소 하나만 바라보며, 운영 모델을 바꾸거나 배포를 늘릴 때도 게이트웨이 설정만 고치면 됩니다. 이 구성에서는 모델 교체가 스킬 파일의 변경으로 이어지지 않고, 반입한 스킬의 해시도 그대로 유지됩니다.

호출 경로 단계

요청 하나가 운영 환경 안에서 거치는 단계는 다음과 같습니다. 모든 단계가 폐쇄망 안에서 끝나고, 실데이터를 다루는 단계와 모델이 개입하는 단계가 나뉘어 있다는 점을 표에서 확인할 수 있습니다.

단계	구성 요소	하는 일	다루는 데이터
1. 요청 접수	MSAP.ai AAS 콘솔·사내 채널	업무 담당자의 요청을 해당 에이전트에 전달	요청 문장
2. 스킬 선택·명령 실행	Hermes 프로파일	첫 줄 규칙으로 스킬을 고르고 결정적 스크립트 실행	요청 문장, 스크립트 결과
3. 모델 호출 통제	LiteLLM	가상 키 확인, 예산·속도 상한 적용, 호출 기록, 장애 시 우회[S9]	모델 입력과 출력
4. 생성	vLLM 로컬 LLM	OpenAI 호환 채팅 요청 처리, 도구 호출 0~1회[S10]	모델 입력과 출력
5. 업무 시스템 읽기·쓰기	스크립트·업무 시스템 연결 도구	사람 확인을 거친 뒤 쓰기, 쓴 결과를 다시 조회해 검증	고객·금액 실데이터
6. 응답	MSAP.ai AAS 콘솔·사내 채널	스크립트 결과 원문을 업무 담당자에게 전달	처리 결과

표 3-7. 운영 환경의 모델 호출 경로 단계표

대화 모델을 게이트웨이 모델로 제한하는 관리자 화면

이 호출 경로는 화면에서도 같은 규칙으로 드러납니다. MSAP.ai AAS 관리자 화면의 대화 모델 목록에는 운영자가 정한 이름 규칙에 맞는 게이트웨이 모델만 나타나고, 임베딩·음성·이미지 모델이나 규칙 밖 이름의 모델은 숨겨집니다. 주 모델을 바꾸는 권한은 관리자에게만 있으므로, 업무 담당자가 대화 창에서 승인되지 않은 모델을 골라 고객·금액 자료를 보내는 경로가 화면 단계에서 막힙니다. 아래 그림은 사용자·부서 관리, 역할 템플릿, 대화 모델 목록을 한 화면에 모은 관리자 기능을 보여 줍니다.



그림 3-2. MSAP.ai AAS 관리자 화면의 사용자·부서, 역할 템플릿, 대화 모델 목록 관리

보안 담당자는 뒤의 표 3-8에서 「운영 모델 호출이 게이트웨이 한 곳만 거치는가」를 확인할 때 이 대화 모델 목록을 게이트웨이 설정과 나란히 대조하면 됩니다.

3.4.2 MCP 명세의 사용자 동의 원칙과 가상 키 기반 예산 통제, 호출 기록

MCP 명세의 사용자 동의 원칙

MCP 명세 2025-06-18 개정본은 사용자가 모든 자료 접근과 작업에 명시적으로 동의하고 그 범위를 통제할 수 있어야 하며, 호스트는 도구를 부르기 전에 사용자 동의를 받아야 한다고 정합니다[S3]. 도구 호출은 곧 임의 코드 실행이므로 신중하게 다뤄야 한다는 원칙도 같은 명세에 있습니다[S3]. MSAP.ai AAS 구성에서 이 원칙은 두 영역에 각각 적용됩니다. 개발 환경에서는 프론티어 AI가 MSAP.ai AAS의 스킬 만들기·올리기 도구를 부를 때 업무 전문가가 호출마다 확인하고, 운영 환경에서는 업무 시스템에 쓰는 단계 앞에 스크립트가 쥔 사람 확인 게이트를 둡니다.

가상 키와 예산, 호출 기록으로 나누는 게이트웨이 통제

게이트웨이 통제는 에이전트마다 가상 키를 따로 발급하는 데서 시작합니다. 키가 나뉘어 있으면 에이전트별로 쓸 수 있는 모델과 예산, 속도 상한을 따로 정할 수 있고, 한 에이전트에서 이상 호출이 생겨도 그 키만 막으면 됩니다[S9]. 호출 기록은 관측 도구로 보내 보존하므로, 감사 때 어느 에이전트가 언제 어떤 모델을 몇 번 불렀는지를 업무 처리 결과와 맞춰 볼 수 있습니다. 게이트웨이 한 곳에서 접근 통제, 비용 통제, 감사 추적이 함께 이뤄집니다.

운영 중 스킬 변경의 사람 검토 대기 설정

Hermes Agent는 에이전트가 `skill_manage` 도구로 스스로 스킬을 만들고 고치고 지울 수 있게 하는데, 운영 환경에서 이 기능이 그대로 켜져 있으면 반입 승인을 받은 스킬이 승인 없이 바뀔 수 있습니다[S8]. 선택 설정 `skills.write_approval`을 켜면 이런 쓰기가 사람의 검토 대기로 넘어가므로, 운영 환경의 스킬은 반입 절차를 거친 판에서 벗어나지 않습니다[S8]. MSAP.ai AAS는 여기에 더해 도구 호출 전 훅으로 운영 중 스킬 수정을 막고, 바뀐 스킬은 원본 매니페스트 기준으로 되돌립니다. 그래서 3.3.1에서 기록한 해시 대조와 이 설정을 함께 쓰면, 운영 환경의 스킬이 승인된 판과 같은지를 언제든 확인할 수 있습니다.

보안 담당자의 승인 전 점검표

아래 점검표는 앞의 내용을 보안 담당자가 승인 전에 하나씩 확인할 수 있는 항목으로 묶은 것입니다. 아홉 항목 가운데 하나라도 확인되지 않으면 그 항목을 보완한 뒤 파일럿을 시작합니다.

점검 항목	확인 방법	근거
개발 환경에 실데이터가 없는가	비식별 데이터베이스와 목업 인터페이스의 값 표본 검토	3.2 구성
반입 묶음이 승인된 판인가	운영 환경 스킬 파일 해시와 반입 승인서 해시 대조	3.3.1 반입 단계
설치 때 보안 검사를 거쳤는가	스킬 허브 설치 검사 결과 확인	[S8]
운영 모델 호출이 게이트웨이 한 곳만 거치는가	LiteLLM 모델 목록에 외부 제공자가 없는지 확인	[S9]

점검 항목	확인 방법	근거
에이전트별 가상 키와 예산 상한이 있는가	게이트웨이의 키·예산·속도 상한 설정 확인	[S9]
모델 호출 기록이 보존되는가	관측 도구에 기록이 쌓이는지 확인	[S9]
쓰기 도구 호출 전 사람 확인이 있는가	스크립트의 확인 게이트와 도구 호출 동의 설정 확인	[S3]
운영 중 스킬 수정이 막혀 있는가	`skills.write_approval` 설정과 스킬 수정 차단 훅 확인	[S8]
비밀값이 스킬 파일 밖에 있는가	스킬 묶음 안에 계정·키 문자열이 없는지 검사, 운영 비밀값 관리 등록 확인	3.1.2 경계 항목

표 3-8. 보안 담당자의 승인 전 점검표

이 점검표가 모두 확인되면, 프론티어 AI는 업무 정의서와 비식별 시험 자료만 보고 실데이터와 운영 모델 호출은 폐쇄망 안에 남는다는 구성이 설정과 기록으로 증명됩니다. 따라서 3장 처음에 세운 질문, 즉 운영 데이터를 밖으로 내보내지 않고도 프론티어 AI를 스킬 제작에 쓸 수 있는가에 대한 답은 이 표의 확인 결과로 내릴 수 있습니다. 다만 이 구성은 스킬이 운영 모델에서도 같은 결과를 낸다는 전제 위에 서 있으므로, 그 전제를 확인하는 제작·시험 절차가 함께 있어야 합니다. 그 절차는 프론티어 AI를 MCP로 MSAP.ai AAS에 연결해 스킬을 만들고 운영 모델로 시험하는 순서로 이어집니다.

4장: 프론티어 AI를 MCP로 연결해 스킬을 만드는 절차

MCP(Model Context Protocol)는 LLM 애플리케이션이 외부 자료와 도구를 같은 규약으로 부를 수 있게 만든 개방형 프로토콜입니다[S3]. MSAP.ai AAS는 이 규약의 서버 역할을 맡아, 업무 전문가가 쓰는 프론티어 AI가 대화 도중에 스킬을 만들고 올리고 켜는 일을 직접 할 수 있게 합니다. 스킬을 만드는 모델과 스킬을 실행하는 모델이 서로 다르다는 점이 이 절차를 따로 정해야 하는 이유입니다. 프론티어 AI가 만든 스킬이 프론티어 AI 자신에게서 잘 동작했다는 사실은 조직 안의 활성 파라미터 3B 급 오픈소스 MoE 모델에서도 같은 결과가 나온다는 보장이 되지 못합니다. 이 장에서는 연결 구조, 스킬 제작 순서, 실행 모델 기준의 시험, 실패를 고치는 반복과 제작 단계의 통제를 차례로 정리해 IT 담당자가 파일럿 환경을 구성하고 의사결정권자가 운영 반입 조건을 판단할 수 있게 합니다.

4.1 MCP 호스트인 프론티어 AI와 MCP 서버인 MSAP.ai AAS의 연결 구조

업무 전문가의 대화가 스킬 파일로 바뀌려면 대화를 받는 프론티어 AI가 MSAP.ai AAS의 기능을 직접 호출할 수 있어야 합니다. 이 연결을 사람이 손으로 잇는다면 프론티어 AI가 만든 파일을 내려받아 서버에 복사하고, 다시 채팅 화면에서 시험하는 일을 반복해야 합니다. 반면 MCP로 연결하면 파일 등록과 시험 요청이 모두 프론티어 AI의 도구 호출이 되므로, 업무 전문가는 대화창 하나에서 제작부터 시험까지 이어서 진행합니다. 연결 구성과 권한 범위는 아래 두 항에서 MCP 표준 용어로 정의합니다[S3].

4.1.1 JSON-RPC 2.0 기반 MCP의 호스트와 클라이언트, 서버 역할

MCP 명세가 정한 세 역할

MCP 명세(Model Context Protocol Specification)은 연결에 참여하는 쪽을 호스트, 클라이언트, 서버 세 역할로 나누고 그 사이의 메시지를 JSON-RPC 2.0 형식으로 주고받게 합니다. JSON-RPC 2.0은 요청과 응답을 JSON 객체 하나로 표현하는 가벼운 원격 호출 규약이며, 메서드 이름과 인자, 결과를 정해진 필드에 담습니다. 호스트는 연결을 시작하는 LLM 애플리케이션이고, 클라이언트는 호스트 안에서 서버 하나와의 연결을 관리하는 연결부이며, 서버는 맥락과 기능을 내놓는 쪽입니다[S3]. 이 백서에서는 이 세 용어를 명세의 뜻 그대로 쓰며, 다른 장에서도 같은 뜻으로 읽으면 됩니다.

역할	MCP 명세의 정의	이 구성에서 맡는 대상	운영상 책임
호스트	연결을 시작하는 LLM 애플리케이션	업무 전문가가 쓰는 프론티어 AI(예: Claude Code, Claude 데스크톱 앱)	도구 호출 전 사용자 동의 확보
클라이언트	호스트 안에서 서버 하나와 연결을 유지하는 연결부	프론티어 AI 애플리케이션에 등록한 MSAP.ai AAS 연결 설정	연결 대상과 인증 정보 관리
서버	맥락과 기능을 제공하는 쪽	MSAP.ai AAS MCP 서버	노출 기능 범위와 권한 검사

표 출처: MCP Specification 2025-06-18[S3]

MCP 서버 기능 세 가지와 MSAP.ai AAS의 선택

MCP 서버가 내놓을 수 있는 기능은 리소스, 프롬프트, 도구 세 가지입니다. 리소스는 모델이 읽을 자료이고, 프롬프트는 미리 짜 둔 메시지나 작업 흐름의 틀이며, 도구는 모델이 호출해 실행하는 함수입니다[S3]. MSAP.ai AAS가 스킬 제작에 내놓는 기능은 대부분 도구에 속합니다. 프론티어 AI는 「이 프로파일에 스킬을 올린다」, 「운영 모델에게 이 질문을 보낸다」 같은 동작을 함수 호출로 실행하고, 그 결과를 다시 대화에 받아 다음 판단에 씁니다. 도구는 서버 쪽에서 실제 동작을 일으키는 기능이므로 명세는 도구를 임의 코드 실행과 같은 무게로 다루라고 요구하며, 이 요구는 4.4.2의 제작 통제로 이어집니다[S3].

호스트가 바뀌어도 유지되는 연결 구성

MCP가 표준 규약이라는 점은 도입 판단에서 실무적인 의미가 있습니다. 호스트 쪽 애플리케이션이 Claude Code이든 다른 MCP 지원 애플리케이션이든 서버 쪽 구성은 바뀌지 않으므로, 조직은 프론티어 AI 공급사를 바꿀 때 MSAP.ai AAS 쪽 설정을 다시 만들 필요가 없습니다. 연결부도 표준이므로 보안 담당자는 호스트별 전용 연동 방식을 따로 검토하지 않고 MCP 명세의 동의·통제 원칙 하나를 기준으로 점검할 수 있습니다[S3]. 다만 호스트가 바뀌면 동의 화면의 모양과 동의 기록 방식은 애플리케이션마다 다르므로, 파일럿에서 쓸 호스트 애플리케이션은 미리 하나로 정해 두는 편이 점검 범위를 줄입니다.

4.1.2 MSAP.ai AAS가 MCP 서버로 내놓는 도구 범위 — 스킬 생성과 업로드, 채팅, 예약 작업, 비밀값 관리

제작 단계와 이어지는 다섯 가지 도구 묶음

MSAP.ai AAS는 오픈소스 Hermes Agent 기반 실행 환경을 웹 콘솔과 MCP 서버 두 경로로 다룹니다. MCP 서버로 내놓는 도구는 에이전트 목록 조회, 스킬 관리, 채팅, 예약 작업, 비밀값 관리 다섯 묶음이며, 묶음마다 스킬 제작의 어느 단계에서 쓰이는지가 정해져 있습니다. 예를 들어 에이전트 목록은 제작을 시작할 때 시험 대상 에이전트와 프로파일을 고르는 데 쓰고, 채팅 도구는 스킬을 올린 뒤 운영 모델에게 실제 요청을 넣는 시험에 씁니다. 아래 표는 도구 이름 대신 기능으로 다섯 묶음을 정리한 것입니다.

도구 묶음	주요 기능	쓰이는 제작 단계	권한·통제
에이전트 목록	연결된 에이전트와 프로파일 조회	시작: 시험 대상 고르기	조회 전용
스킬 관리	단일 SKILL.md 생성, 스크립트를 포함한 폴더 업로드·적용, 켜기·끄기, 조회·수정·삭제	제작·등록·활성화	업로드는 에이전트 소유자만, 스크립트 포함 시 명시 확인
채팅	세션 시작, 요청 보내기, 진행 상황 기다리기, 승인·되묻기 응답	시험	승인 요청은 한 번·세션·항상·거부 중 선택
예약 작업	정해진 시각에 실행할 작업 생성·조회·즉시 실행	운영 전 반복 시험	프로파일 단위
비밀값 관리	등록 여부 확인, 화면 입력 요청	연결 준비	값은 스킬 파일 밖에 보관

스킬 관리 도구의 두 가지 등록 경로

스킬 관리 묶음에서 가장 자주 쓰는 기능은 생성과 업로드입니다. 생성은 SKILL.md 파일 하나로 끝나는 짧은 스킬을 프로파일에 바로 만드는 기능이고, 업로드는 SKILL.md와 스크립트·규칙 파일이 함께 든 폴더를 묶어 올리는 기능입니다. Anthropic Agent Skills 규격에서 스킬은 SKILL.md를 담은 디렉터리이고 스크립트를 넣으면 에이전트가 그 스크립트를 도구처럼 실행하므로[S2], 결정적 코드를 담는 스킬은 거의 언제나 업로드 경로를 탑니다. 업로드한 묶음은 바로 켜지지 않고 검사를 거친 뒤 적용 단계에서 프로파일에 반영되며, 스크립트가 들어 있으면 적용 요청에 스크립트 실행을 허용한다는 확인이 따로 필요합니다. 그래서 스킬 업로드는 파일 복사 한 번이 아니라 검사와 확인을 포함한 두 단계 작업으로 다루어야 합니다.

채팅·예약 작업·비밀값 도구의 제작 단계 역할

채팅 도구는 세션을 열고 요청을 보낸 뒤 결과를 기다리는 세 동작으로 나뉩니다. 기다리기 동작은 진행 중, 완료, 승인 요청, 되묻기, 오류 같은 상태를 돌려주고, 진행 중에는 운영 모델이 부른 도구의 이름과 상태를 함께 보여 주므로 시험하는 쪽이 도구 호출 수를 바로 셀 수 있습니다. 예약 작업 도구는 크론(cron, 정해진 시각에 작업을 반복 실행하는 일정 표기) 형식으로 작업을 등록하며, 아침마다 같은 질문 묶음을 다시 돌려 스킬이 계속 같은 결과를 내는지 확인하는 데 씁니다. 비밀값 관리 도구는 업무 시스템 접속 정보 같은 값을 스킬 파일과 대화 밖에 두도록 설계되어 있으며, 이 원칙은 4.4.2에서 제작 통제 항목으로 다시 정리합니다. 한편 Hermes Agent는 스킬을 정해진 기본 디렉터리에 두고 이를 정본으로 삼으므로[S8], MSAP.ai AAS의 스킬 도구는 이 디렉터리를 프로파일 단위로 다루는 관리 기능이라고 이해하면 됩니다.

MSAP.ai AAS가 읽는 범위와 바꾸는 범위

MCP 도구와 웹 콘솔이 손대는 범위는 Hermes Agent의 계층마다 다릅니다. 런타임 계층에서는 게이트웨이 상태와 모델 연결을 읽기만 하고 플러그인·설정·버전은 바꾸지 않으며, 프로파일 계층에서는 성격과 모델 정의를 공식 API로 만들고 내보내고 지웁니다. 스킬과 예약 작업 계층에서는 스킬 문서를 편집하고 예약 작업을 등록·수정하며, 맨 위의 MCP 서버·외부 시스템 계층은 스킬이 부르는 호출 관계와 자격증명의 이름·위치만 읽고 값과 연결 설정은 고객 환경에 그대로 둡니다. 실행 주체가 각 계층에 남아 있으므로 콘솔을 붙이거나 떼어도 운영 중인 예약 작업은 계속 실행됩니다. 아래 그림은 계층별로 MSAP.ai AAS가 읽는 것과 바꾸는 것을 나란히 보여 줍니다.



그림 4-1. Hermes Agent 계층별로 MSAP.ai AAS가 읽는 범위와 바꾸는 범위

따라서 IT 담당자는 MCP 서버로 여는 도구 범위를 승인할 때, 런타임 설정과 외부 시스템 연결은 이 도구들로 바뀌지 않는다는 점을 전제로 둘 수 있습니다.

4.2 업무 정의서를 건넨 뒤 스킬을 만들어 올리고 켜기까지의 작업 순서

업무 전문가가 2장에서 정리한 업무 정의서를 프론티어 AI에게 건네면, 그다음부터 스킬이 프로파일에서 켜지기까지는 대화 단위가 정해진 작업입니다. 순서를 고정해 두지 않으면 프론티어 AI가 SKILL.md에 긴 설명을 먼저 쓰고 스크립트를 뒤늦게 덧붙이거나, 시험 대상이 아닌 프로파일에 스킬을 올리는 일이 생깁니다. 정의서 제시, 규칙·스크립트·시험 생성, 짧은 SKILL.md 확정, 등록과 활성화의 순서로 대화를 나누고 단계마다 산출물을 확인합니다. Agent Skills의 점진적 공개 구조, 즉 이름·설명을 먼저 보고 본문과 참고 파일은 필요할 때만 여는 방식도 이 순서와 맞물립니다[S2].

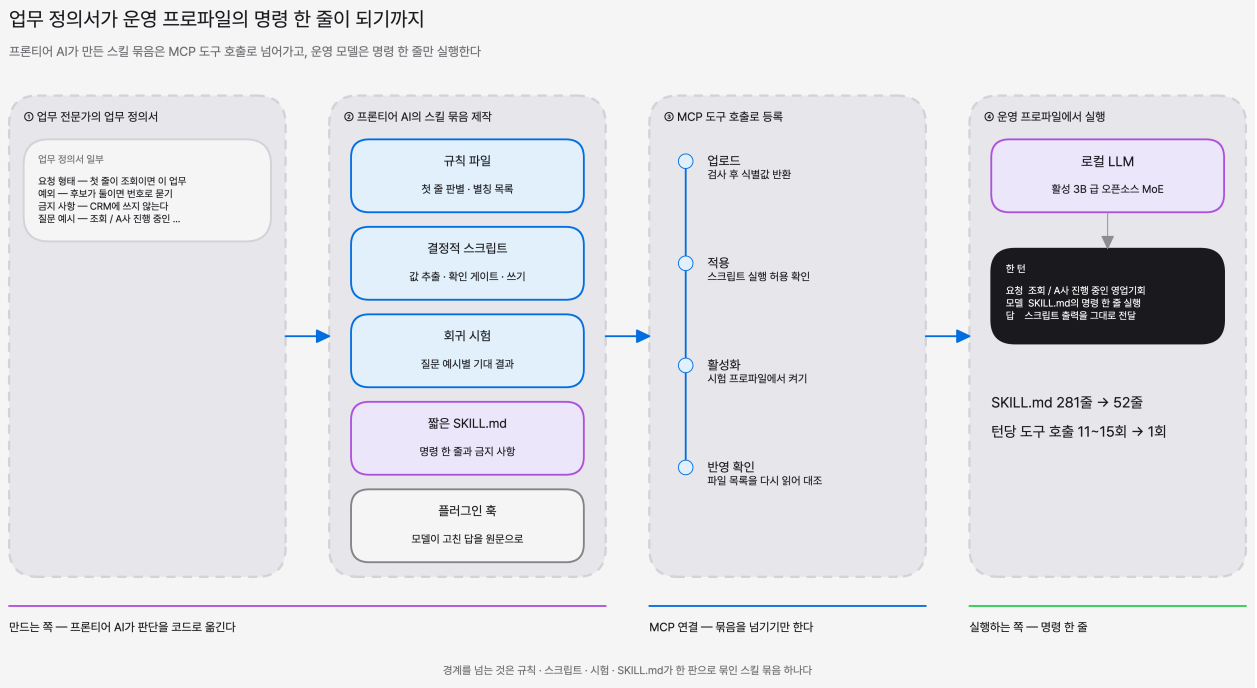


그림 4-2. 프론티어 AI가 만든 스킬 묶음이 MCP를 거쳐 운영 프로파일로 전달되는 순서

그림은 만드는 쪽과 실행하는 쪽의 경계에서 무엇이 넘어가는지를 보입니다. 왼쪽의 프론티어 AI는 업무 정의서를 읽고 판단 규칙과 스크립트, 시험을 파일로 만들며, 가운데의 MCP 연결은 도구 호출로 그 파일 묶음을 MSAP.ai AAS에 넘기는 일만 맡습니다. 오른쪽의 운영 프로파일에서는 활성 파라미터 3B 급 오픈소스 MoE 모델이 SKILL.md에 적힌 명령 한 줄을 실행하고, 판단은 등록된 스크립트가 맡습니다.

4.2.1 업무 전문가와 프론티어 AI가 주고받는 대화 단위와 단계별 산출물

업무 정의서 제시와 판단 목록 확인

첫 대화 단위는 업무 정의서를 건네고 프론티어 AI에게 판단 목록을 되받는 일입니다. 업무 정의서에는 지시문 해석 규칙, 결정적 코드로 옮길 판단, 질문 예시가 들어 있어야 하며, 프론티어 AI는 이를 읽고 「요청에서 어떤 값을 뽑는가」, 「어떤 경우에 되묻는가」, 「어떤 순서로 시스템을 조회하고 쓰는가」를 목록으로 돌려줍니다. 업무 전문가는 이 목록을 보고 빠진 판단이나 잘못 이해한 업무 용어를 바로잡습니다. 업무 전문가가 코드를 읽을 필요

는 없고, 목록이 자신이 평소 머릿속으로 내리는 판단과 같은지만 확인하면 됩니다. 이 확인을 건너뛰면 뒤 단계에서 스크립트가 아무리 정확해도 업무와 다른 일을 정확하게 수행하는 스킬이 만들어집니다.

규칙·스크립트·시험 생성 단계

두 번째 대화 단위에서 프론티어 AI는 판단 목록을 규칙 파일과 스크립트, 회귀 시험으로 옮깁니다. 규칙 파일은 첫 줄 판별 규칙이나 값 추출 패턴처럼 업무 전문가가 나중에 고칠 수 있는 설정이고, 스크립트는 그 규칙을 읽어 조회·확인·쓰기를 순서대로 실행하는 결정적 코드이며, 회귀 시험은 업무 정의서의 질문 예시를 하나씩 입력으로 넣어 기대 결과와 비교하는 시험 묶음입니다. Anthropic은 스크립트가 맥락에 내용을 싣지 않고도 늘 같게 실행되는 결정적 신뢰성을 준다고 설명하며[S2], 도구를 설계할 때는 여러 예시 입력으로 시험해 모델의 실수를 찾아 고치라고 권합니다[S1]. 이 단계의 합격 조건은 「프론티어 AI가 만든 회귀 시험이 모두 통과하는가」이며, 시험이 없으면 다음 단계로 넘어가지 않습니다.

짧은 SKILL.md 확정과 대화 단위별 산출물

세 번째 대화 단위는 SKILL.md를 짧게 확정하는 일입니다. SKILL.md 앞머리에는 `name`과 `description`이 반드시 있어야 하고[S2], 본문에는 실행 모델이 부를 명령 한 줄과 하지 말아야 할 동작만 남기며 긴 설명은 참고 파일로 옮깁니다. 예를 들어 한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영한 영업기회 등록 스킬은 SKILL.md를 281줄에서 52줄로 줄이자 실행 모델의 턴당 도구 호출이 11~15회에서 1회로 줄었습니다. 아래 표는 세 대화 단위와 마지막 등록 단위의 산출물을 정리한 것입니다.

대화 단위	업무 전문가가 하는 일	프론티어 AI의 산출물	다음 단위로 넘어가는 조건
① 정의서 제시	업무 정의서 전달, 판단 목록 확인	판단 목록(값 추출·되묻기·순서·확인·쓰기)	업무 전문가가 목록 승인
② 규칙·스크립트·시험 생성	질문 예시 보강	규칙 파일, 결정적 스크립트, 회귀 시험	회귀 시험 전부 통과
③ SKILL.md 확정	명령 한 줄과 금지 사항 확인	짧은 SKILL.md, 참고 파일	본문이 명령·금지 사항만 담음
④ 등록·활성화	시험 대상 프로파일 지정, 스크립트 실행 허용 확인	업로드 묶음, 적용·활성화 결과	시험 프로파일에서 스킬 켜짐 확인

4.2.2 MCP 도구 호출로 스킬을 등록하고 시험 대상 에이전트에 활성화하는 절차

시험 대상 에이전트와 프로파일 고정

등록 절차의 첫 단계는 에이전트 목록 도구로 시험 대상 에이전트와 프로파일을 확인하고 이후 모든 호출에 같은 값을 쓰는 일입니다. MSAP.ai AAS에서는 스킬과 비밀값, 예약 작업이 모두 프로파일 단위로 묶이므로, 대상 프로파일을 잘못 고르면 시험은 통과했는데 운영 프로파일에는 스킬이 없는 상태가 생깁니다. 파일럿에서는 운영 프로파일과 같은 모델·같은 플러그인 구성을 가진 시험 프로파일을 하나 두고, 프론티어 AI에게 이 프로파일 이름을 업무 정의서와 함께 처음부터 알려 줍니다. 그러면 제작 대화 전체에서 등록 대상이 한 곳으로 고정되고, 다른 업무의 프로파일을 건드리는 실수도 막을 수 있습니다.

업로드·검사·적용·활성화 네 단계

스킬 등록은 업로드, 검사, 적용, 활성화의 네 단계로 진행합니다. 먼저 프론티어 AI가 SKILL.md를 맨 위에 둔 폴더를 묶어 업로드하면 MSAP.ai AAS가 묶음을 검사하고 식별값을 돌려줍니다. 다음으로 그 식별값으로 적용을 요청하는데, 묶음에 스크립트가 있으면 스크립트 실행을 허용한다는 확인을 함께 보내야 적용됩니다. 마지막으로 켜기 도구로 해당 프로파일에서 스킬을 활성화하고 스킬 조회 도구로 실제 반영된 파일 목록을 다시 읽어 업로드한 묶음과 같은지 확인합니다. Hermes Agent의 스킬 허브도 설치할 때마다 보안 검사를 거치는데[S8], MSAP.ai AAS의 업로드 경로 역시 검사와 명시 확인을 거치므로 개발 환경의 등록 절차가 3장에서 다룬 폐쇄망 반입 절차와 같은 모양을 갖습니다.

단계	호출하는 기능	확인할 결과	실패 시 조치
1 업로드	스킬 폴더 묶음 업로드	검사 통과, 식별값 반환	SKILL.md 위치·앞머리 필드 확인
2 적용	식별값으로 적용 요청(스크립트 포함 시 실행 허용 확인)	프로파일에 스킬 반영	확인 누락 여부, 소유자 권한 확인
3 활성화	프로파일 단위 스킬 켜기	활성 상태	같은 이름의 옛 스킬 충돌 확인
4 반영 확인	스킬 조회·파일 읽기	반영된 파일과 업로드 묶음 일치	다시 업로드

같은 이름의 옛 스킬과 묶음 단위 교체

수정한 스킬을 다시 올릴 때는 파일 하나만 고쳐 넣기보다 묶음 전체를 다시 업로드하는 편이 안전합니다. 규칙 파일과 스크립트, 회귀 시험은 서로 맞물려 있어 한쪽만 바뀌면 시험은 옛 규칙을 기준으로 통과하고 실제 실행은 새 규칙을 따르는 어긋남이 생기기 때문입니다. 같은 기능을 하는 옛 스킬이 켜진 채 남아 있으면 실행 모델이 요청을 어느 스킬로 보낼지 헷갈릴 수 있으므로, 새 스킬을 켤 때 옛 스킬은 끄기 도구로 함께 끕니다. 즉 등록의 단위는 파일이 아니라 「규칙·스크립트·시험·SKILL.md가 한 판으로 묶인 스킬 묶음」이며, 이 원칙이 8장의 변경 관리와 이어집니다.

4.3 MSAP.ai AAS 채팅으로 운영 모델에 실제 요청을 보내는 시험 — 만든 모델이 아닌 실행 모델 기준의 판정

스킬이 켜진 뒤의 시험은 프론티어 AI가 MCP 채팅 도구로 MSAP.ai AAS 운영 모델에게 실제 요청을 보내고 그 응답을 받아 판정하는 방식으로 진행합니다. 프론티어 AI가 스킬의 스크립트를 자기 환경에서 직접 돌려 보는 방법도 있지만, 그 결과는 운영 모델이 같은 스킬을 받았을 때의 동작을 알려 주지 않습니다. NVIDIA 연구진의 입장 논문이 주장하듯 에이전트 안의 언어 모델은 적은 수의 전문 작업을 거의 변화 없이 반복하며[S4], 운영에서 스킬을 반복 실행하는 쪽은 작은 모델이므로 합격 판정의 기준도 그 모델이어야 합니다. 이 절의 시험 절차는 판정 기준을 실행 모델로 고정하는 데 목적이 있습니다.

4.3.1 프론티어 AI가 직접 실행한 결과를 합격 근거로 쓰지 않는 이유

만든 모델과 실행 모델의 능력 차이

프론티어 AI는 SKILL.md의 설명이 길고 순서가 모호해도 맥락을 메워 가며 올바른 순서로 도구를 부릅니다. 반면 활성 파라미터 3B 급 MoE 모델은 같은 설명을 받으면 단계를 건너뛰거나, 확인 없이 쓰기를 시도하거나, 스크립트 출력을 자기 말로 고쳐 전달할 수 있습니다. Anthropic은 「Building effective agents」에서 미리 정한 코드 경로로 LLM과 도구를 엮는 워크플로와 LLM이 스스로 순서와 도구 사용을 정하는 에이전트를 구분하는데 [S1], 프론티어 AI에게는 에이전트 방식으로도 충분한 스킬이 작은 모델에게는 워크플로 쪽으로 판단을 옮겨야 하는 스킬이 됩니다. 따라서 프론티어 AI가 직접 실행해 얻은 정답은 그 스킬이 「프론티어 AI에게서 된다」는 증거일 뿐이며, 운영 반입을 판단하는 근거로 쓰지 않습니다.

서버에서 직접 만들고 시험하는 방식과의 차이

스킬을 만드는 또 다른 방법은 운영 서버에 직접 접속해 명령줄 환경에서 프론티어 AI로 스킬을 만들고 바로 실행해 보는 것입니다. 이 방법은 서버 파일을 오가는 시간과 토큰이 많이 들고, 무엇보다 시험을 수행하는 모델이 프론티어 AI가 되어 실행 모델 기준의 판정이 사라집니다. MCP로 MSAP.ai AAS에 연결하면 만드는 모델은 프론티어 AI이고 시험받는 모델은 실제 운영 모델이라는 역할 분담이 연결 구조 자체에 들어가므로, 업무 전문가가 따로 신경 쓰지 않아도 판정 기준이 어긋나지 않습니다. MCP 연결은 편의 기능인 동시에 판정 기준을 지키는 장치입니다.

만든 모델과 실행 모델의 결과 차이 예시

한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영했을 때 관찰된 동작을 보면 두 모델의 차이가 수치로 드러납니다. 영업기회 등록 스킬은 판단을 스크립트로 옮기기 전에는 실행 모델이 턴마다 도구를 11~15회 부르며 191~207초를 썼고, 판단을 코드로 옮긴 뒤에는 도구 호출 1회와 18초·33초 두 턴으로 끝났습니다. 이때 합격의 기준으로 삼은 것은 실행 모델이 낸 결과 JSON이 프론티어 AI가 같은 명령을 실행한 결과와 같은지였습니다. 프론티어 AI의 결과는 정답표로 쓰고, 합격 여부는 실행 모델이 그 정답표를 정해진 시간과 호출 수 안에서 재현하는지로 가립니다.

비교 항목	프론티어 AI가 직접 실행	운영 모델이 MSAP.ai AAS 채팅으로 실행
판정에서의 역할	기대 결과(정답표) 생성	합격 판정 대상
긴 SKILL.md 처리	맥락으로 순서를 메움	단계 누락·순서 뒤바뀜 가능
확인 게이트	스스로 지킴	코드가 쥐지 않으면 건너뛸 수 있음
스크립트 출력 전달	대체로 원문 유지	자기 말로 고쳐 쓸 수 있음
관찰 예(영업기회 등록, 판단 이관 전)	결과 JSON 정상	턴당 도구 11~15회, 191~207초

4.3.2 도구 호출 수와 소요 시간, 정답 여부를 시험 자리에서 확인하는 절차

채팅 시험의 다섯 단계

시험은 세션 시작, 요청 보내기, 기다리기, 응답 처리, 기록의 다섯 단계로 진행합니다. 프론티어 AI는 시험 프로파일에 채팅 세션을 열고 업무 정의서의 질문 예시를 하나씩 보낸 뒤, 기다리기 도구를 반복 호출하며 상태가 완료가 될 때까지 진행 상황을 받습니다. 진행 중 상태에는 운영 모델이 부른 도구의 이름과 상태가 함께 표시되므로 프론티어 AI는 이를 세어 도구 호출 수를 기록하고, 세션 시작부터 완료까지의 시간을 소요 시간으로 적습니

다. 다만 중간에 승인 요청이나 되묻기 상태가 오면 그 응답은 업무 전문가가 정한 대로만 하며, 프론티어 AI가 임의로 승인하지 않게 업무 정의서에 응답 규칙을 적어 둡니다.

합격선 세 지표의 기록

시험 기록에는 도구 호출 수, 소요 시간, 정답 여부 세 지표를 남깁니다. MSAP.ai AAS가 예시로 쓰는 합격선은 실제 요청 하나에 도구 호출 3회 이하, 몇 분 안의 응답, 정답이며, 이 기준은 8장에서 운영 판정 기준으로 다시 다룹니다. 정답 여부는 응답 문장을 눈으로 읽어 판단하지 않고, 프론티어 AI가 만든 정답표와 운영 모델의 결과 값을 항목별로 비교해 판정합니다. 쓰기 작업이 있는 스킬은 응답 문장만 보지 않고 업무 시스템을 다시 조회해 실제로 기록이 생겼는지까지 확인해야 하는데, 응답이 「등록했습니다」라고 말해도 실제 쓰기가 일어나지 않은 경우가 있기 때문입니다.

시험 기록 양식

질문 예시 하나가 시험 한 건이며, 시험마다 아래 양식의 한 줄을 남깁니다. 예약 작업 도구로 같은 질문 묶음을 정해진 시각에 다시 돌리면 이 기록이 날짜별로 쌓여 스킬이 시간이 지나도 같은 결과를 내는지 보여 줍니다. 아래 양식의 기대 결과 열은 프론티어 AI의 실행 결과에서, 나머지 열은 운영 모델의 시험에서 채웁니다.

항목	기록 내용	합격선(예시)
질문 예시 번호·원문	업무 정의서의 질문 그대로	–
기대 결과	프론티어 AI 실행 결과의 핵심 값	–
운영 모델 결과	응답의 핵심 값, 재조회 결과	기대 결과와 일치
도구 호출 수	진행 상태에 표시된 호출 수	3회 이하
소요 시간	세션 시작부터 완료까지	몇 분 안
상태 이력	승인·되묻기·오류 발생 여부	계획된 되묻기만 허용
실패 유형	지시문 해석·코드·예시 부족 중 하나	실패 시 기재

4.4 실패 유형별 수정 반복과 제작 단계의 사용자 동의, 비밀값 분리 보관

운영 모델 시험에서 실패한 스킬은 실패 원인을 분류해 고친 뒤 같은 질문 예시로 다시 시험합니다. 이 반복이 업무 전문가의 대화만으로 이어지려면 실패를 5장에서 다루는 스킬 작성 지시 세 가지 가운데 어디가 부족했는지로 나누어야 합니다. 또한 제작 반복 동안 프론티어 AI는 MSAP.ai AAS의 도구를 수십 번 호출하므로, MCP 명세가 요구하는 사용자 동의와 비밀값 보관 원칙을 제작 단계부터 지켜야 운영 반입 승인을 받을 수 있습니다 [S3].

4.4.1 시험 실패를 지시문 해석, 코드, 예시 부족으로 나눠 고치는 수정 반복

실패 유형 세 가지의 증상

시험 기록의 실패는 대부분 세 유형 가운데 하나로 나뉩니다. 첫째, 지시문 해석 실패는 운영 모델이 요청을 다른 스킬로 보내거나, 모호한 요청을 되묻지 않고 추측으로 진행하는 경우입니다. 둘째, 코드 실패는 모델이 스크립트 대신 스스로 순서를 엮어 도구 호출 수가 합격선을 넘거나, 확인 게이트를 건너뛰거나, 스크립트 출력을 고쳐

전달하는 경우입니다. 셋째, 예시 부족은 업무 정의서에 있는 질문 꼴에서는 통과하지만 같은 뜻을 다른 말로 물으면 실패하는 경우입니다. 이렇게 증상으로 유형을 정해 두면 업무 전문가는 코드를 읽지 않고도 프론티어 AI에게 어느 부분을 고치라고 지시할 수 있습니다.

유형별 수정 위치

유형이 정해지면 고칠 파일도 정해집니다. 지시문 해석 실패는 첫 줄 판별 규칙표와 되묻기 문구를 담은 규칙 파일을 고치고, 코드 실패는 모델에게 남아 있던 판단을 스크립트로 옮기며, 예시 부족은 실패한 질문을 질문 예시에 더하고 회귀 시험에 고정합니다. Anthropic은 도구 인자를 바꿔 실수 자체가 어려워지게 하는 설계를 권하며, 상대 경로 대신 절대 경로를 요구하자 경로 오류가 사라진 사례를 듭니다[S1]. 코드 실패를 고칠 때도 같은 원칙을 적용해 모델이 실수할 수 있는 선택지를 스크립트 인자에서 없애는 쪽으로 고칩니다.

실패 유형	시험 기록의 증상	수정 위치	다시 시험할 범위
지시문 해석	다른 스킬로 감, 추측으로 진행, 계획에 없는 되묻기	첫 줄 판별 규칙표, 정해진 질문 문구	실패 질문 + 라우팅 관련 질문 예시 전체
결정적 코드	도구 호출 수 초과, 확인 게이트 누락, 출력 고쳐 쓰기	스크립트(판단 이관), SKILL.md 명령 한 줄	회귀 시험 전체
질문 예시 부족	같은 뜻 다른 표현에서만 실패	질문 예시 추가, 회귀 시험 고정	추가한 예시 + 기존 예시 전체

수정 뒤 회귀 시험 전체 재실행

수정한 뒤에는 실패한 질문만 다시 보내지 않고 회귀 시험 전체를 다시 돌립니다. 라우팅 규칙 한 줄을 고치면 이전에 통과하던 다른 질문이 다른 스킬로 가는 일이 생길 수 있기 때문입니다. 수정 반복의 순서는 프론티어 AI 환경에서 회귀 시험 통과, 묶음 전체 재업로드와 적용, 운영 모델 채팅 시험으로 고정합니다. 다만 같은 유형의 실패가 세 번 넘게 되풀이되면 업무 정의서 자체가 업무를 잘못 나눈 것일 수 있으므로, 업무 전문가가 정의서를 다시 쓰는 단계로 돌아갑니다.

4.4.2 도구 호출 전 사용자 동의와 비밀값 분리 보관 기반의 제작 단계 통제

MCP 명세의 동의 원칙과 제작 단계 적용

MCP 명세는 사용자가 모든 자료 접근과 작업에 명시적으로 동의하고 통제해야 하며, 호스트는 도구를 부르기 전에 사용자 동의를 받아야 한다고 정합니다[S3]. 스킬 제작에 이 원칙을 적용하면 동의가 필요한 호출은 세 종류입니다. 스킬을 올리고 적용하는 호출, 스킬을 켜거나 끄는 호출, 운영 모델이 승인을 요청했을 때 그 승인에 답하는 호출입니다. 이 가운데 승인 응답은 한 번, 세션 전체, 항상, 거부 가운데 하나를 고르게 되어 있는데, 제작 단계에서는 「항상」을 고르지 않고 시험 건마다 「한 번」으로 답하도록 업무 정의서에 적어 두면 시험 중에 쓰기 권한이 넓어지는 일을 막을 수 있습니다.

비밀값을 스킬 파일 밖에 두는 원칙

업무 시스템 접속 정보 같은 비밀값은 스킬 파일, 업무 정의서, 대화 어디에도 적지 않습니다. MSAP.ai AAS는 비밀값을 프로파일 단위로 보관하며, 제작 단계에서는 프론티어 AI가 등록 여부 확인과 입력 요청 기능만 쓰고 실제 값은 사용자가 MSAP.ai AAS 화면에서 직접 입력하는 방식을 권합니다. 대화에 이미 나온 값을 그대로 등록하는 기능도 있지만, 그 값은 프론티어 AI 공급사와 호스트 애플리케이션의 대화 기록에 남으므로 제작 단계에

서는 쓰지 않습니다. 비밀값을 스킬 묶음 밖에 두면 3장에서 다룬 폐쇄망 반입 때 스킬 묶음을 그대로 옮겨도 개발 환경의 접속 정보가 운영 환경으로 따라 들어가지 않습니다.

운영 반입 전 제작 통제 점검표

보안 담당자와 IT 담당자는 운영 반입을 승인하기 전에 아래 항목을 함께 확인합니다. 점검 항목은 모두 MCP 명세의 동의·통제 원칙과[S3] Hermes Agent의 설치 시 보안 검사 원칙에[S8] 맞춰 정했으며, 시험 기록과 MSAP.ai AAS의 스킬 조회 결과만으로 확인할 수 있습니다. 이 점검표를 통과한 스킬은 제작 과정에서도 운영과 같은 수준의 통제를 받았다는 근거를 갖추게 됩니다.

점검 항목	확인 방법	기준
도구 호출 전 동의	호스트 애플리케이션의 동의 설정	업로드·적용·켜기 호출은 매번 사용자 확인
스크립트 실행 허용	적용 기록	스크립트가 든 묶음은 명시 확인 후 적용
업로드 권한	에이전트 소유자 설정	시험 프로파일 소유자만 업로드
승인 응답 범위	시험 기록의 상태 이력	「항상」 선택 0건
비밀값 위치	스킬 묶음·업무 정의서 검색	접속 정보 문자열 0건, 프로파일 비밀값으로만 보관
묶음 일치	스킬 조회 결과와 업로드 묶음 비교	파일 목록·내용 일치
시험 결과	시험 기록표	합격선(도구 3회 이하·몇 분 안·정답) 충족

이 점검표의 마지막 항목인 시험 결과가 합격선을 넘으려면, 프론티어 AI가 처음부터 운영 환경의 작은 모델에 맞춘 스킬을 만들어야 합니다. 프론티어 AI는 따로 지시받지 않으면 자기 수준에서 읽기 쉬운 스킬을 만들기 때문에, 업무 정의서와 함께 건넬 스킬 작성 지시가 필요하며 그 세 가지 지시가 5장의 주제입니다.

5장: 작은 모델이 일하게 만드는 스킬 작성 지시 세 가지

MSAP.ai AAS에서 업무 에이전트가 쓰는 스킬은 SKILL.md 파일과 스크립트, 참고 파일을 담은 디렉터리이며, Anthropic이 공개한 Agent Skills 형식과 오픈소스 Hermes Agent의 스킬 형식을 그대로 따릅니다[S2] [S8]. 스킬은 개발 환경에서 프론티어 모델이 MCP로 MSAP.ai AAS에 연결해 만들고, 운영 환경에서는 활성 파라미터 3B 급 오픈소스 MoE 모델이 실행합니다. 핵심 문제는 만드는 모델과 실행하는 모델의 능력 차이입니다. 프론티어 모델은 따로 지시받지 않으면 자기 수준에서 읽기 쉬운 스킬을 만들고, 그런 스킬은 긴 설명을 읽고 여러 도구를 차례로 골라 엮는 일을 실행 모델에 맡깁니다. 작은 모델은 그 과정에서 순서를 건너뛰거나 빠진 값을 추측하고, 같은 요청에도 결과가 매번 달라집니다.

Anthropic의 「Building effective agents」는 LLM과 도구를 미리 정한 코드 경로로 엮은 시스템을 워크플로, 모델이 스스로 순서와 도구 사용을 정하는 시스템을 에이전트로 나누고, 결과가 실제로 좋아질 때만 복잡도를 더하라고 권합니다[S1]. 작은 모델이 실행할 스킬은 이 구분에서 워크플로 쪽으로 무게를 옮겨야 같은 결과를 냅니다. NVIDIA 연구진도 에이전트 안의 언어 모델은 적은 수의 전문 작업을 거의 변화 없이 반복하므로 작은 모델이 에이전트 호출 대부분에 알맞다는 입장을 밝혔습니다[S4]. 다만 이 논문은 실험 결과를 보고한 연구가 아니라 방향을 제안한 입장 논문이므로, 작은 모델이 실제로 같은 결과를 내게 하는 장치는 스킬 설계 안에 따로 넣어야 합니다.

업무 전문가가 프론티어 모델에게 스킬을 만들라고 말할 때 반드시 함께 적어야 하는 지시는 세 가지이고, 받은 스킬을 검토할 때 확인할 항목은 두 가지입니다. 아래 표는 세 지시가 각각 어떤 실패를 막고 스킬 안에 어떤 산출물을 남기는지 정리한 것이며, 이 장의 판단은 조직의 스킬 작성 지침에 세 지시와 검토 항목 두 가지를 넣을지입니다. 각 지시를 다루는 5.1~5.3에는 업무 전문가가 프론티어 모델에게 그대로 붙여 쓸 수 있는 지시 문장 예시가 하나씩 들어 있으므로, 자기 업무의 명령 이름과 값만 바꾸면 바로 업무 정의서에 옮길 수 있습니다.

지시	프론티어 모델에게 적는 내용	막는 실패	스킬에 남는 산출물
① 지시문 해석	요청 첫 줄과 본문으로 부를 명령을 고르고, 모르면 정해진 질문을 출력하고 멈춘다	추측으로 엉뚱한 업무를 시작함	판별 규칙표, 되묻기 문구
② 결정적 코드	값 추출부터 재조회 검증까지 스크립트가 하고, 모델은 명령 한 줄 실행과 원문 전달만 한다	같은 요청에 다른 결과, 승인 없는 쓰기	진행기 스크립트, 60줄 이내 SKILL.md
③ 질문 예시	질문 패턴과 업무 시나리오를 많이 주고 예시마다 회귀 시험으로 고정한다	변경 뒤 알아채지 못한 결과 변화	예시 입력 목록, 기대 결과 묶음

5.1 지시 ① 지시문 해석 — 요청 첫 줄 기준의 명령 선택과 불명확한 요청의 정해진 질문 후 정지

지시문 해석은 사용자가 보낸 요청 문장을 읽고 스킬 안의 여러 명령 가운데 어느 것을 부를지 고르는 단계입니다. 결정적 코드만 있고 해석 단계가 없으면 스킬은 「값을 넣어 주세요」만 되묻는 입력 양식이 되고, 반대로 해석을 모델의 판단에 맡기면 작은 모델이 요청에 섞인 다른 업무의 낱말에 끌려 엉뚱한 명령을 실행합니다. 따라서 지시 ①은 해석 단계를 스킬 맨 앞에 두되 판별 규칙은 규칙표와 정규식 같은 코드로 쓰고, 판별할 수 없으면 정해진 질문을 출력하고 멈추라고 요구합니다.

5.1.1 요청 첫 줄과 본문을 분석해 부를 명령을 고르는 해석 단계 설계

요청 첫 줄을 명령 이름으로 읽는 규칙

MSAP.ai AAS의 업무 스킬은 요청의 첫 줄을 가장 강한 판별 신호로 다룹니다. 첫 줄이 「견적」이나 「영업기회」처럼 미리 정해 둔 명령 낱말이면, 해석 단계는 본문에 어떤 낱말이 있든 그 명령을 고릅니다. 첫 줄을 우선하는 이유는 업무 요청의 본문에 다른 업무의 낱말이 자주 섞이기 때문입니다. 예를 들어 영업기회 등록을 요청하는 메일 본문에는 고객사 이름과 사업명 외에도 지난 견적 번호나 제품 이름이 함께 들어 있어서, 본문 전체를 모델에게 읽혀 업무를 고르게 하면 견적 작성 스킬이 불릴 수 있습니다. 첫 줄 우선 규칙은 사용자 입장에서도 「첫 줄에 업무 이름을 쓴다」는 한 줄짜리 약속이므로, 교육 부담 없이 판별 정확도를 높이는 방법입니다.

본문에서 명령에 필요한 값을 읽는 규칙

첫 줄로 명령이 정해지면 본문은 그 명령이 쓸 값을 꺼내는 데 씁니다. 고객사, 사업명, 금액, 날짜, 서버 이름 같은 값은 「고객사:」 같은 라벨, 날짜와 금액의 정규식, 고객사 이름 색인으로 스크립트가 찾습니다. 모델이 본문을 요약해서 값을 넘기는 대신 스크립트가 원문에서 값을 직접 읽어 다음 단계로 전하므로, 요약 과정에서 글자가 바뀌거나 숫자가 빠지는 일이 생기지 않습니다. 라벨이 없는 자유 문장에서는 정규식이 값을 찾지 못하는 경우가 생기는데, 이때 스크립트는 값을 짐작해 채우지 않고 5.1.2의 되묻기 규칙으로 넘어갑니다.

명령 낱말 없는 요청은 신호어 규칙표로 판별

첫 줄이 정해 둔 명령 낱말이 아니면 스크립트가 본문의 신호어를 규칙표와 대조합니다. 규칙표는 「취약점」, 「CVE」, 「패치」 같은 낱말 묶음마다 명령 하나를 짝지어 둔 표이며, 두 명령이 같은 점수로 맞거나 아무 명령도 맞지 않으면 고르지 않고 되묻습니다. 해석 결과는 언제나 명령 하나를 선택하거나 되묻는 두 가지 중 하나이고, 모델이 두 명령을 섞어 실행하는 경우는 생기지 않습니다. 아래 표는 이 규칙이 실제 요청 문장을 어떤 명령으로 옮기는지 보여 줍니다.

요청 예시 (첫 줄 / 본문 일부)	해석 근거	선택 명령
견적 / ○○공사 모니터링 구축 건, 라이선스 20식	첫 줄 명령 낱말	견적 작성
영업기회 / 첨부 공고 기준 등록, 지난 견적 번호 참고	첫 줄 명령 낱말 (본문의 「견적」은 판별에 쓰지 않음)	영업기회 등록
서버 1에 OS 취약점 조치됐어?	신호어 「취약점」 단독 일치	패치 적용 조회
지난달 공지된 CVE는 몇 대에 적용됐어?	신호어 「CVE」 단독 일치	패치 적용 조회
이거 처리해 줘	일치하는 명령 없음	되묻기

해석 규칙을 코드로 두었을 때 업무 전문가가 얻는 통제권

Anthropic은 도구 설명을 새로 온 개발자에게 주는 문서처럼 쓰고 사용 예와 경계 사례, 입력 형식을 담으라고 권합니다[S1]. 해석 규칙표는 그 권고를 사람이 읽는 설명에서 끝내지 않고 실행되는 코드로 옮긴 형태입니다. 작은 모델은 규칙표를 해석할 필요가 없고, 해석 단계가 낸 선택 결과에 따라 정해진 명령 한 줄을 실행하기만 합니다. 결과적으로 명령 선택의 정확도는 모델 크기보다 규칙표의 품질로 정해지며, 업무 전문가는 오분류가 생긴 요청 문장을 보고 규칙표의 신호어를 직접 고칠 수 있습니다.

업무 전문가가 지시 ①로 적는 문장 예시 — 명령 세 개를 가진 영업 지원 에이전트

업무 전문가가 지시 ①을 줄 때 코드나 정규식을 직접 쓸 필요는 없습니다. 명령 이름, 첫 줄 낱말, 본문 신호어, 판별할 수 없을 때 출력할 질문을 업무 언어로 적으면, 프론티어 모델이 그 문장을 규칙표 파일과 판별 스크립트로 옮깁니다. 아래 예시는 견적 작성, 영업기회 등록, 패치 적용 조회 세 명령을 가진 에이전트에서 쓴 문장이며, 위 해석 규칙 표와 같은 내용을 프론티어 모델에게 주는 형태로 적은 것입니다.

> [지시 ① 지시문 해석]
> 이 에이전트의 명령은 세 개다: 견적 작성, 영업기회 등록, 패치 적용 조회.
> - 요청 첫 줄이 「견적」이면 견적 작성, 「영업기회」면 영업기회 등록을 부른다.
> 첫 줄로 명령이 정해지면 본문에 다른 업무 낱말이 있어도 명령을 바꾸지 않는다.
> - 첫 줄이 명령 낱말이 아니면 본문 신호어로 고른다.
> 「취약점」「CVE」「패치」「조치」 → 패치 적용 조회
> - 신호어가 하나도 없거나 두 명령에 똑같이 맞으면 고르지 말고,
> 아래 질문을 글자 그대로 출력한 뒤 멈춘다.
> 「어떤 업무인가요? 1) 견적 2) 영업기회 등록 3) 패치 조회 중 번호로 답해 주세요.」
> - 고객사·사업명·금액은 「고객사:」「사업명:」「금액:」 라벨에서 읽고,
> 찾지 못하면 짐작해 채우지 말고 빠진 값을 되묻는다.
> - 이 판별 규칙은 규칙표 파일과 판별 스크립트로 만들고, 모델이 판단하게 두지 않는다.

5.1.2 모델이 추측하지 않고 코드가 정한 질문을 출력한 뒤 멈추는 규칙

추측 대신 되묻기를 코드에 정해 두는 이유

작은 모델은 모르는 값을 만나면 그럴듯한 값을 채워 넣는 경향이 있고, 업무 시스템에 기록하는 스킬에서는 그 한 번의 추측이 잘못된 등록이나 엉뚱한 서버 조회로 이어집니다. MCP 명세가 호스트에게 도구를 부르기 전에 사용자 동의를 받도록 요구하는 것도 도구 실행이 실제 부작용을 낳는 임의 코드 실행이기 때문입니다[S3]. 그래서 지시 ①은 판별에 필요한 값이 하나라도 비면 스크립트가 미리 정해 둔 질문을 출력하고 그 턴을 끝내도록 요구하며, 모델은 질문 문구를 새로 지어내지 않습니다. 질문 문구가 코드에 고정돼 있으면 같은 상황에서 언제나 같은 질문이 나오므로, 사용자가 답하는 방식도 함께 일정해집니다. 작은 언어 모델이 범위가 좁고 변화가 적은 작업에 알맞다는 주장[S4]도 되묻기처럼 입력과 출력의 형식이 정해진 작업일 때 성립합니다.

되묻기 문구에 담는 세 가지 정보

되묻기 문구는 무엇이 비었는지, 어떤 형식으로 답하면 되는지, 지금 고를 수 있는 후보가 무엇인지를 한 번에 알려야 합니다. 예를 들어 고객사 이름이 색인에서 두 곳과 맞으면 번호 선택지를 주고, 사업명이 빠졌으면 한 줄로 적어 달라고 형식을 지정합니다. 형식이 정해진 질문을 받으면 사용자의 다음 답이 짧고 형식이 정해져 있어 작은 모델이 그 답을 해석하다 틀릴 가능성이 줄고, 스크립트가 번호나 문자열을 그대로 받아 다음 단계를 이어서 실행할 수 있습니다. 아래 표는 업무 스킬에서 자주 쓰는 되묻기 질문의 예입니다.

상황	출력할 질문	기대하는 답 형식
명령 판별 불가	「어떤 업무인가요? 1) 견적 2) 영업기회 등록 3) 패치 조회 중 번호로 답해 주세요.」	번호
고객사 후보 여러 곳	「고객사가 두 곳과 일치합니다. 1) ○○공사 2) ○○공단 중 번호로 답해 주세요.」	번호
필수 값 누락	「사업명이 없습니다. 사업명을 한 줄로 적어 주세요.」	한 줄 문장
대상 서버 누락	「대상 서버 이름을 적어 주세요. 예: web-01」	서버 이름

멈춘 뒤 사용자 답으로 이어서 실행하는 방식

되묻기 뒤 멈추는 규칙에는 이어서 실행하는 규칙이 짝으로 따라야 합니다. 스크립트는 질문을 출력할 때 지금까지 읽은 값과 진행 단계를 상태 파일에 기록해 두고, 다음 사용자 답이 오면 그 상태에서 다시 시작합니다. 사용자가 번호 하나만 답해도 요청을 처음부터 다시 해석하지 않고, 앞서 읽은 고객사와 금액 같은 값이 그대로 유지됩니다. 대화가 길어지면 작은 모델이 앞 맥락을 놓치기 쉬운데, 진행 상태를 맥락 창 대신 파일이 기억하므로 이 문제도 함께 피할 수 있습니다.

되묻기 횟수를 줄이는 업무 정의서 작성법

되묻기가 잦으면 사용자는 스킬을 번거로운 양식으로 느끼고 쓰지 않게 됩니다. 되묻기는 값이 정말 없거나 후보가 둘 이상일 때로 한정하고, 라벨과 정규식, 색인으로 찾을 수 있는 값은 먼저 찾아 보여 준 뒤 확인만 받는 편이 좋습니다. 다만 라벨 없는 자유 문장에서 값을 뽑는 일은 아직 되묻기로 넘어가는 경우가 많으므로, 업무 전문가는 업무 정의서에 요청 문장의 권장 형식을 함께 적어 두어야 합니다. 예를 들어 「첫 줄에 업무 이름, 둘째 줄부터 고객사:, 사업명:, 금액: 라벨」 같은 형식을 사용자 안내에 넣으면, 해석 단계가 값을 찾지 못하는 경우가 눈에 띄게 줄어듭니다.

5.2 지시 ② 결정적 코드 — 값 추출부터 재조회 검증까지 스크립트로 옮기는 범위

결정적 코드(Deterministic Code)는 같은 입력을 받으면 언제나 같은 출력과 같은 부작용을 내는 코드입니다. Anthropic은 스킬에 넣은 스크립트가 코드만 줄 수 있는 결정적 신뢰성을 준다고 설명하며, PDF 양식 필드를 추출하는 스크립트가 스크립트 내용이나 PDF를 맥락에 싣지 않고도 늘 같은 결과를 낸다는 예를 듭니다[S2]. 지시 ②는 이 성질을 스킬 전체로 넓혀, 업무 판단 대부분을 스크립트로 옮기고 실행 모델의 몫을 명령 한 줄 실행과 결과 원문 전달로 줄이라는 요구입니다. 같은 요청에 같은 결과를 보장하는 설계의 중심이 이 지시이며, 진행기라는 구성 요소가 그 결과물입니다.

5.2.1 값 추출, 후보 선정, 확인 게이트, 쓰기, 재조회 검증의 스크립트 이관 목록

프론티어 모델이 만든 스킬에서 모델 판단을 찾아 목록으로 만드는 방법

이관 범위를 정하려면 먼저 프론티어 모델이 처음 만든 스킬에서 실행 모델이 판단하는 일을 모두 적어 내야 합니다. 대상은 SKILL.md 본문의 「~를 확인하고」, 「적절한 ~를 골라」, 「필요하면 ~한다」 같은 문장이며, 이런 문

장 하나하나가 작은 모델이 실행 중에 스스로 결정해야 하는 일입니다. 판단 목록은 SKILL.md에서 조건과 선택을 말하는 문장을 찾아 뽑으면 만들 수 있고, 코드를 읽지 못하는 업무 전문가도 문장 단위로 확인할 수 있습니다. 이 판단 목록이 지시 ②의 이관 범위이자, 5.4.2에서 스킬을 검토할 때 쓰는 기준표가 됩니다.

업무 에이전트 스킬에서 스크립트로 옮기는 여섯 항목

업무 시스템을 조회하고 기록하는 에이전트 스킬에서 판단 목록은 대개 여섯 항목으로 모입니다. 아래 표는 각 항목에서 모델이 하던 판단과 스크립트가 대신하는 일을 나란히 놓은 것으로, 프론티어 모델에게 지시 ②를 줄 때 이 여섯 항목을 이름으로 적어 이관 범위를 명시합니다.

이관 항목	모델이 하던 판단	스크립트가 하는 일
값 추출	본문을 읽고 값을 요약해 넘김	라벨·정규식·색인으로 원문에서 값을 읽음
후보 선정	비슷한 고객사·품목 중 하나를 고름	색인 대조 점수로 후보를 정하고, 동점이면 되묻기
순서	다음에 부를 도구를 매 턴 고름	정해진 단계 순서대로 실행
확인 게이트	쓰기 전에 사용자에게 물을지 판단	드라이런 결과를 보여 주고 승인 문구가 올 때만 진행
쓰기	도구 인자를 조립해 시스템에 기록	같은 키가 있으면 새로 만들지 않는 멱등 쓰기
재조회 검증	「등록했습니다」라고 답함	쓴 뒤 시스템을 다시 읽어 값이 맞는지 대조

확인 게이트와 쓰기를 코드가 쥐어야 하는 이유

여섯 항목 가운데 가장 먼저 옮겨야 할 것은 확인 게이트와 쓰기입니다. 확인 게이트가 모델 몫으로 남으면 작은 모델은 사용자의 승인을 기다리지 않고 쓰기를 실행하거나, 반대로 이미 승인된 작업을 다시 묻습니다. 또한 멱등 쓰기, 즉 같은 요청을 여러 번 실행해도 한 번 실행한 것과 결과가 같은 쓰기가 없으면 같은 요청을 두 번 받았을 때 같은 영업기회가 두 건 만들어집니다. 재조회 검증은 모델의 「완료했습니다」라는 답과 실제 시스템 상태를 구분하는 장치이므로, 진행 순서의 마지막 단계에 반드시 둡니다. 쓰기 결과를 코드가 다시 읽어 대조해야만 운영 담당자가 감사 추적에 쓸 수 있는 확인 기록이 남습니다.

업무 전문가가 지시 ②로 적는 문장 예시 — 영업기회 등록

지시 ②를 적을 때 업무 전문가는 여섯 이관 항목을 자기 업무의 값과 판정 기준으로 바꿔 적습니다. 스크립트가 주는 결정적 신뢰성은 스크립트로 옮긴 범위 안에서만 생기므로[S2], 「결정적 코드로 만들어라」 한 줄만 주면 무엇을 코드로 옮길지를 모델이 스스로 정하게 되고 5.4의 대조처럼 확인 게이트가 모델 몫으로 남을 수 있습니다. 따라서 항목마다 이름을 붙여 적는 것이 이 지시의 핵심이며, 아래는 메일로 받은 공고를 영업기회로 등록하는 업무에서 쓴 문장의 예입니다.

> [지시 ② 결정적 코드 — 영업기회 등록]
> 아래 여섯 가지는 모두 진행기 스크립트가 하고, 모델에게 판단을 맡기지 않는다.
> 1. 값 추출: 메일 원문에서 수요기관, 사업명, 공고번호, 마감일을 라벨과 정규식으로 읽는다.

> 2. 후보 선정: 수요기관을 CRM 고객사 색인과 대조한다. 두 곳 이상 맞으면 번호로 되묻는다.
> 3. 순서: 값 추출 → 후보 확인 → 드라이런 → 사용자 확인 → 등록 → 재조회 순서를 바꾸지 않는다.
> 4. 확인 게이트: 드라이런 결과를 보여 주고, 사용자가 「진행」이라고 답한 경우에만 등록한다.
> 5. 쓰기: 같은 기관·같은 사업명의 영업기회가 이미 있으면 새로 만들지 않고 기존 건을 알려 준다.
> 6. 재조회 검증: 등록 뒤 CRM을 다시 읽어 기관·사업명·마감일이 입력과 같은지 대조해 출력한다.
> 모델은 진행기 명령 한 줄만 실행하고, 진행기가 낸 문장을 고치지 않고 그대로 전한다.

스크립트 이관을 뒷받침하는 공개 근거

Anthropic은 정해진 코드 경로를 따르는 워크플로가 잘 정의된 작업에 예측 가능성과 일관성을 주고, 모델이 스스로 경로를 정하는 에이전트는 유연성을 얻는 대신 비용과 지연이 늘고 오류가 누적될 수 있다고 설명합니다[S1]. 실행 모델이 작을수록 단계마다 생기는 판단 오류가 다음 단계로 이어질 가능성이 크므로, 반복 업무 스킬에서는 판단을 코드로 옮기는 쪽이 정확도와 비용 모두에서 유리합니다. 게다가 스크립트는 그 내용을 맥락에 싣지 않고 실행 결과만 돌려주므로[S2], 매 턴 긴 업무 설명을 모델에게 읽히는 방식보다 입력 토큰 소비가 줄어듭니다. 결정적 코드는 같은 결과를 보장하는 장치이면서 보급형 GPU에서 응답 시간을 줄이는 장치이기도 합니다.

5.2.2 모델 몫을 명령 한 줄 실행과 원문 전달로 줄이는 진행기 설계

진행기의 정의와 처리 단계

진행기는 스킬 안에서 단계를 순서대로 실행하는 결정적 스크립트입니다. 해석, 값 추출, 되묻기, 드라이런, 확인, 쓰기, 재조회까지 5.1과 5.2.1에서 정한 일을 한 프로그램이 정해진 순서로 실행하고, 각 단계의 결과와 다음에 사용자에게 보여 줄 문장을 함께 출력합니다. 실행 모델은 단계마다 도구를 고를 필요가 없고, 진행기 명령 한 줄을 실행한 뒤 출력된 문장을 사용자에게 그대로 전달하면 됩니다. 진행기는 상태 파일에 진행 단계를 기록하므로 같은 명령을 다시 불러도 처음부터 반복하지 않고 멈춘 단계에서 이어서 실행합니다. 즉 진행기를 넣은 스킬은 Anthropic이 말하는 워크플로, 곧 미리 정한 코드 경로로 LLM과 도구를 엮은 구조에 해당합니다[S1].

진행기 도입 전후의 모델 몫과 코드 몫

진행기를 넣으면 실행 모델이 맡던 일 대부분이 코드로 넘어가고, 모델에게는 명령 실행과 원문 전달 두 가지만 남습니다. 아래 표는 같은 업무 스킬에서 진행기 도입 전후로 각 일을 누가 맡는지 대비한 것입니다.

일	진행기 도입 전	진행기 도입 후
부를 명령 선택	모델이 SKILL.md를 읽고 판단	진행기의 해석 단계 (첫 줄 규칙·규칙표)
값 찾기와 후보 고르기	모델	진행기
단계 순서	모델이 매 턴 도구를 고름	진행기의 고정 순서
확인·쓰기·재조회	모델	진행기
실행 방식	여러 도구를 차례로 호출	모델이 진행기 명령 한 줄 실행
사용자 응답	모델이 결과를 요약·재작성	모델이 진행기 출력 원문을 그대로 전달

원문 전달을 작은 모델에게 따로 지시하는 이유

원문 전달은 사소해 보이지만 작은 모델에게는 따로 지시해야 하는 규칙입니다. 작은 모델은 진행기가 낸 「확인하시면 '진행'이라고 답해 주세요」 같은 문장을 자기 말로 다듬다가 승인 문구를 바꾸거나 후보 번호를 빠뜨리는 경우가 있고, 그러면 다음 턴에서 진행기가 사용자 답을 인식하지 못합니다. SKILL.md에는 「출력을 고치지 말고 그대로 전하라」는 금지 문장을 명령 한 줄과 함께 적습니다. 다만 지시만으로 전달 충실도를 완전히 지키기는 어렵기 때문에, MSAP.ai AAS는 모델이 고쳐 쓴 답을 진행기 원문으로 바꿔 넣는 출력 직후 훅을 운영 프로파일에 함께 둡니다(6장).

진행기를 쓸 때 짧아지는 SKILL.md

진행기를 쓰면 SKILL.md에 적을 내용이 크게 줄어듭니다. Agent Skills의 점진적 공개 방식에서 에이전트는 이름과 설명만 먼저 보고, 관련이 있어 보일 때 SKILL.md 본문을 읽으며, 연결된 추가 파일은 필요할 때만 엽니다[S2]. Hermes Agent 스킬 문서도 같은 방식으로 스킬 목록을 먼저 보고 본문과 참고 파일은 나중에 읽는다고 설명합니다[S8]. 지시 ②에는 「SKILL.md는 60줄 이내로 명령 한 줄과 금지 사항만 적고, 업무 규칙 설명은 references 폴더의 참고 파일로 옮긴다」는 문장을 넣습니다. 작은 모델이 읽어야 할 글이 줄면 그 글을 다르게 해석할 여지도 함께 줄어듭니다.

5.3 지시 ③ 질문 예시 — 질문 패턴과 업무 시나리오를 회귀 시험으로 고정하는 예시 설계

질문 예시는 업무 전문가가 실제로 받는 질문의 여러 꼴과 업무 시나리오를 프론티어 모델에게 미리 건네는 지시입니다. 예시가 많을수록 프론티어 모델은 해석 규칙표와 진행기 명령을 더 잘게, 더 다양한 꼴의 질문에 맞게 만들고, 업무 전문가가 정확한 형식 없이 말해도 스킬이 요청을 알아듣습니다. 각 예시를 기대 결과와 묶어 회귀 시험(Regression Test)으로 고정하면, 운영을 시작한 뒤 모델이나 업무 규칙이 바뀌어도 결과가 달라졌는지 바로 확인할 수 있습니다.

5.3.1 많은 예시 입력으로 모델 실수를 미리 찾는 시험 설계와 실수 방지 인자

여러 예시 입력으로 시험하라는 공개 권고

Anthropic은 도구를 만들 때 여러 예시 입력으로 직접 시험해 보고, 모델이 어떤 실수를 하는지 찾아 고치라고 권합니다[S1]. 사람이 대표 요청 몇 개로 시험하면 스킬이 잘 동작하는 것처럼 보이지만, 작은 모델의 실수는 낱말 순서가 바뀌거나 줄임말이 섞이거나 다른 업무의 낱말이 끼어든 질문에서 주로 나타납니다. 지시 ③은 업무 전문가에게 질문 꼴을 가능한 한 많이 적게 하고, 프론티어 모델이 그 질문 전부를 해석 규칙표로 처리하도록 요구합니다. 예를 들어 서버 보안 패치 업무라면 「서버 1에 OS 취약점 조치됐어?」, 「지난달 공지된 CVE는 몇 대에 적용됐어?」, 「web-01 패치 내역 보여 줘」가 모두 같은 조회 명령으로 가야 합니다.

업무 정의서에 넣는 예시 입력 유형 목록

예시 입력은 표준 꼴만 모으면 시험 효과가 작으므로, 작은 모델이 틀리기 쉬운 꼴을 일부러 섞어야 합니다. 아래 표는 업무 정의서에 넣을 예시 입력 유형과 각 유형이 확인하는 규칙을 정리한 것입니다.

유형	예시	확인하는 규칙
표준 꼴	「견적 / ○○공사 모니터링 구축」	첫 줄 명령 규칙
줄임·구어 꼴	「서버 1 취약점 조치됐어?」	신호어 규칙표

유형	예시	확인하는 규칙
다른 업무 낱말이 섞인 꼴	영업기회 요청 본문에 지난 견적 번호가 있음	첫 줄 우선 규칙
값이 빠진 꼴	「견적 / 라이선스 20식」(고객사 없음)	되묻기 문구
후보가 여럿인 꼴	고객사 이름이 두 곳과 일치	번호 선택지
범위 밖 요청	「점심 메뉴 추천해 줘」	처리 범위 밖 안내 후 정지

업무 전문가가 지시 ③으로 적는 질문 예시 — 의도 하나를 여러 표현으로

지시 ③에서 효과가 가장 큰 부분은 같은 의도를 사람마다 다르게 말하는 표현을 한곳에 모아 주는 것입니다. 아래 예시는 서버 보안 패치 업무에서 「지정한 서버에 지정한 조치가 적용됐는지 조회」라는 의도 하나를 여러 꼴로 적은 것이며, 업무 전문가는 실제 요청 메일과 메신저 기록에서 문장을 그대로 옮겨 적으면 됩니다. 또한 값이 빠진 질문과 낱말은 비슷하지만 다른 명령으로 가야 하는 질문을 함께 적어 두면, 프론티어 모델이 되묻기 조건과 명령 사이의 경계까지 함께 정합니다. 이는 사용 예와 경계 사례를 담고 여러 예시 입력으로 시험하라는 Anthropic의 도구 설계 권고를 업무 전문가의 언어로 옮긴 형태입니다[S1].

> [지시 ③ 질문 예시 — 패치 적용 조회]
> 의도: 지정한 서버에 지정한 패치·취약점 조치가 적용됐는지 확인한다.
> 아래 질문은 모두 같은 명령(패치 적용 조회)으로 가야 한다.
> - 서버 1에 OS 취약점 조치됐어?
> - web-01 이번 달 보안 패치 적용 여부 알려 줘
> - CVE-2026-XXXX 우리 웹서버에 반영됐나요?
> - 웹 1호기 패치 했어?
> - 지난주 공지된 OpenSSL 취약점, db-02는 어떻게 됐어?
> - 서버1 커널 업데이트 끝났는지 확인 부탁
> 값이 빠진 질문 (되묻기 후 정지):
> - 패치 다 됐어? → 대상 서버 이름을 묻는다
> 다른 명령으로 가야 하는 질문:
> - 견적 / 패치 지원 서비스 1년 → 첫 줄 「견적」이므로 견적 작성
> - 서버 1에 패치 적용해 줘 → 조회 전용 스킬이므로 처리 범위 밖 안내 후 정지
> 업무 시나리오:
> 보안 공지가 나오면 서버 목록과 대조해 해당 서버를 찾고,
> 조치 계획을 만들어 담당자 확인을 받은 뒤 결과를 기록한다.
> 각 질문은 기대 명령·기대 값·되묻기 여부·쓰기 여부와 묶어 회귀 시험으로 저장한다.

실수 자체를 어렵게 만드는 실수 방지 인자 설계

예시 시험에서 찾은 실수는 SKILL.md 문구를 다듬어 고치기보다, 실수가 일어나기 어렵게 명령 인자를 바꿔 고칩니다. Anthropic은 이 방식을 실수 방지(poka-yoke) 설계라고 부르며, 에이전트가 루트 디렉터리를 벗어난 뒤 상대 경로를 잘못 쓰자 도구가 절대 경로만 받도록 바꿨더니 경로 오류가 사라진 사례를 듭니다[S1]. 업무 스킬에도 같은 원칙이 적용됩니다. 고객사를 자유 문자열 대신 색인 번호로 받고, 날짜를 한 가지 형식으로만 받고, 쓰기 명령은 드라이런 결과의 확인 번호가 있어야만 실행되게 하면 작은 모델이 인자를 잘못 조립할 수 있는 경우가 처음부터 줄어듭니다.

질문 예시가 결정적 코드의 범위를 넓히는 효과

질문 예시는 시험 자료이면서 동시에 설계 입력입니다. 프론티어 모델은 예시가 적으면 대표 질문 하나에 맞춘 명령 하나를 만들지만, 예시가 많으면 「서버 한 대의 패치 조회」, 「CVE별 적용 현황」, 「기간별 조치 내역」처럼 명령을 나누고 명령마다 인자를 정합니다. 업무 전문가가 적은 질문 예시의 폭이 곧 스킬이 처리할 수 있는 요청의 폭을 정합니다. 그래서 업무 정의서에는 시나리오마다 표준 꼴 하나가 아니라 여러 꼴의 질문을 함께 적고, 실제 업무 메일이나 요청 기록에서 가져온 문장을 그대로 쓰는 편이 효과가 큽니다.

5.3.2 예시마다 기대 결과를 묶어 모델 교체와 업무 변경 때 다시 실행하는 회귀 시험

질문 예시와 기대 결과를 한 묶음으로 두는 회귀 시험 형식

회귀 시험은 질문 예시 하나와 그 예시에서 기대하는 결과를 한 묶음으로 저장해 두고, 스킬이 바뀔 때마다 전부 다시 실행해 결과가 달라지지 않았는지 대조하는 시험입니다. 기대 결과에는 최종 답 문장만 적지 않고 선택된 명령, 추출된 값, 되묻기 여부, 쓰기 실행 여부를 함께 적습니다. 그래야 답 문장은 비슷해 보여도 내부에서 다른 명령이 실행된 경우를 찾아낼 수 있습니다. 아래 표는 패치 조회 예시 하나를 회귀 시험 묶음으로 적은 형식입니다.

필드	예시 값
입력	「서버 1에 OS 취약점 조치됐어?」
기대 명령	패치 적용 조회
기대 값	서버 = 서버 1, 범위 = OS
되묻기	없음
쓰기 실행	없음
기대 출력	진행기 출력 원문이 답에 그대로 포함됨

결정적 코드이기 때문에 가능한 기계 대조

진행기는 스크립트이므로 같은 입력에 늘 같은 결과를 내고[S2], 그래서 회귀 시험의 대조는 사람이 읽지 않고 스크립트가 할 수 있습니다. 같은 입력에 같은 출력이 나와야 하는 코드에서는 기대 결과와 실제 결과의 차이가 곧 회귀이며, 차이가 한 건도 없으면 통과입니다. 반면 모델이 답을 직접 만드는 스킬은 문장이 매번 조금씩 달라 사람이 다시 읽어야 하므로 회귀 시험을 자주 실행하기 어렵습니다. 따라서 회귀 시험은 진행기 수준의 필드(명령 선택, 값, 되묻기, 쓰기 여부)는 완전 일치로 판정하고, 모델이 전달한 최종 문장은 진행기 원문을 그대로 포함했는지로 판정합니다.

회귀 시험을 다시 실행하는 네 가지 시점

회귀 시험은 스킬 코드를 고칠 때뿐 아니라 스킬을 둘러싼 실행 조건이 바뀔 때마다 다시 실행해야 합니다. 예를 들어 LiteLLM 게이트웨이는 여러 LLM 제공자를 OpenAI 호환 접속 주소 하나로 묶어, 기존 클라이언트 코드를 바꾸지 않고도 뒤의 모델을 교체할 수 있게 합니다[S9]. 교체가 쉬운 만큼 회귀 시험 없이 모델을 바꾸면 결과 변화를 놓치기 쉽습니다. 서빙 설정도 같은 이유로 대상이 되는데, vLLM의 OpenAI 호환 서버는 `parallel_tool_calls=false` 일 때 요청당 도구 호출을 0~1회로 제한하고 여러 호출 여부는 모델에 맡기므로 설정 하나로 같은 스킬의 동작이 달라질 수 있습니다[S10]. 아래 표는 다시 실행해야 하는 시점을 정리한 것입니다.

시점	다시 실행하는 이유
운영 모델 교체	게이트웨이 뒤의 모델이 바뀌면 같은 명령 한 줄도 다르게 처리될 수 있음
서빙·추론 설정 변경	도구 호출 제한, 생각 모드 켜고 끄기 같은 설정이 응답 방식을 바꿈
업무 규칙 변경	규칙표·진행기·되묻기 문구를 고치면 다른 예시의 결과가 함께 바뀔 수 있음
스킬 파일 변경 감지	운영 중 에이전트가 스킬을 고치면 검증하지 않은 내용이 실행됨

운영 중 스킬 변경과 회귀 시험의 관계

넷째 시점은 오픈소스 Hermes Agent의 기능과 관련이 있습니다. Hermes Agent는 에이전트가 `skill_manage` 도구로 스스로 스킬을 만들고 고치고 지울 수 있게 하며, 선택 설정 `skills.write_approval`로 그 쓰기를 사람의 검토를 기다리게 할 수 있습니다[S8]. 공개 MoE 모델 Qwen3-30B-A3B가 생각 모드를 켜고 끄는 설정을 제공하는 것처럼[S7], 실행 조건은 스킬 바깥에서도 자주 바뀝니다. 그래서 회귀 시험 묶음은 스킬 디렉터리 안에 함께 넣어 스킬 업로드 때 운영 환경으로 같이 옮기고, 개발 환경에서 통과한 시험을 폐쇄망 운영 환경에서 다시 실행해 같은 결과가 나오는지 확인합니다. 최종 합격 판정은 MSAP.ai AAS 채팅에 실제 요청을 넣는 실측(8장)으로 하며, 회귀 시험은 그 실측 앞에서 먼저 통과해야 하는 기계 대조입니다.

5.4 「결정적 코드로」 한 줄 지시의 한계 — 확인 게이트가 모델 몫으로 남은 대조 결과

세 지시 가운데 결정적 코드는 「결정적 코드로 작성하라」는 한 줄만으로도 충분해 보이기 쉬운 지시입니다. 그러나 MSAP.ai AAS 구성으로 운영하는 한 업무 에이전트에서 그 한 줄만 주고 스킬을 다시 쓰게 했을 때, 스킬의 구조는 줄었지만 확인 게이트와 모호성 판단은 여전히 모델 몫으로 남았습니다. 지시를 주는 일과 지시가 실제로 반영됐는지 확인하는 일은 별개의 작업이며, 스킬을 받는 쪽에 그 확인을 위한 검토 항목이 필요합니다.

5.4.1 한 줄 지시만 받은 작은 모델의 재작성 결과 — 줄어든 SKILL.md와 그대로 남은 모델 판단

한 줄 지시 대조에 쓴 스킬과 실행 모델

대조는 한 업무 에이전트의 기존 스킬을 놓고, 운영 환경과 같은 활성 파라미터 3B 급 오픈소스 MoE 모델에게 「이 스킬을 결정적 코드 중심으로 다시 작성하라」는 한 줄만 주어 재작성하게 한 방식이었습니다. 지시문 해석 규칙, 5.2.1의 이관 항목 목록, 질문 예시는 주지 않았습니다. 다시 말해 지시 ②의 이름만 있고 내용은 없는 상태에서 모델이 스스로 이관 범위를 정하게 한 것입니다. 재작성한 스킬은 같은 요청으로 실행해 SKILL.md 길이, 실행 중 도구 호출 수, 모델에 남은 판단을 재작성 전과 비교했습니다.

재작성 전후의 SKILL.md 길이와 도구 호출, 남은 판단

재작성 결과는 수치만 보면 개선이었지만, 판단을 누가 맡는지로 다시 나누어 보면 핵심 판단 두 가지가 바뀌지 않았습니다. 아래 표는 그 결과를 재작성 전후로 나란히 놓은 것이며, 수치는 한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영했을 때 관찰된 동작입니다.

비교 항목	재작성 전	한 줄 지시로 재작성한 후
SKILL.md 길이	약 142줄	약 62줄
실행 중 도구 호출	15회	7회
조회 단계	모델이 도구를 하나씩 호출	스크립트 명령으로 일부 묶임
확인 게이트	모델	모델 (변화 없음)
모호성 판단 (후보 고르기)	모델	모델 (변화 없음)

줄어든 수치 뒤에 남은 모델 판단의 문제

표의 앞 두 줄만 보면 재작성은 성공한 것처럼 보입니다. 하지만 같은 요청에 같은 결과를 보장하는 데 가장 중요한 두 판단, 곧 쓰기 전에 사용자 승인을 받을지와 비슷한 후보 가운데 무엇을 고를지는 그대로 모델이 맡고 있었습니다. 작은 모델은 반복되는 조회를 스크립트로 묶어 구조를 짧게 만드는 일은 해냈지만, 어떤 판단이 결과를 바꾸는지는 스스로 찾아내지 못했습니다. 이 스킬을 그대로 운영에 올렸다면 도구 호출은 줄었어도 승인 없는 쓰기와 잘못된 후보 선택의 위험은 재작성 전과 같았을 것입니다. 모델이 스스로 경로를 정하는 구조에서는 단계마다 오류가 쌓일 수 있다고 Anthropic이 지적한 위험이 그대로 남은 것입니다[S1].

프론티어 모델에게 이관 항목을 이름으로 지정하는 이유

이 대조는 작은 모델이 재작성한 경우이지만, 프론티어 모델도 한 줄 지시만 받으면 비슷한 선택을 할 수 있습니다. 프론티어 모델에게는 확인 게이트를 모델이 판단해도 자기 수준에서는 잘 처리되는 일이므로, 그 판단을 코드로 옮겨야 할 이유가 스스로 드러나지 않을 수 있기 때문입니다. 따라서 지시 ②는 「결정적 코드로」 한 줄에서 끝내지 않고 5.2.1의 여섯 항목을 이름으로 적어 이관 범위를 명시해야 합니다. Anthropic이 가장 단순한 해법에서 시작해 결과가 실제로 좋아질 때만 복잡도를 더하라고 권한 것처럼[S1], 지시문도 결과를 재어 보고 빠진 항목만 늘려 가는 방식으로 다듬는 것이 맞습니다.

5.4.2 스킬 검토 항목 두 가지 — 판단 목록의 코드 이관 여부와 확인 게이트의 코드 소유

검토 항목 1: 판단 목록의 코드 이관 여부

첫 번째 검토 항목은 SKILL.md의 판단 목록이 모두 코드로 옮겨졌는지입니다. 검토자는 받은 SKILL.md에서 조건과 선택을 말하는 문장을 찾아 목록으로 만들고, 각 판단이 진행기 스크립트의 어느 단계에서 처리되는지 짝을 맞춥니다. 짝이 없는 판단이 남아 있으면 그 판단은 여전히 실행 모델이 맡고 있다는 뜻이므로, 프론티어 모델에게 해당 항목을 이름으로 지정해 다시 옮기게 합니다. 이 검토는 코드를 읽지 못하는 업무 전문가도 SKILL.md 문장 단위로 할 수 있고, 진행기 단계 이름만 IT 담당자가 알려 주면 됩니다.

검토 항목 2: 확인 게이트의 코드 소유

두 번째 검토 항목은 확인 게이트를 코드가 쥐고 있는지입니다. 판정 기준은 단순해서, 사용자 승인 문구가 오지 않은 상태에서 쓰기 명령을 직접 불렀을 때 진행기가 쓰기를 거부하면 코드가 게이트를 쥔 것이고 그대로 실행되면 모델이 쥔 것입니다. IT 담당자는 이 확인을 드라이런 환경에서 명령 한 번으로 할 수 있으며, 그 결과를 「승인 없는 쓰기 거부」라는 예시로 회귀 시험 묶음에 추가해 두면 이후의 모든 변경에서 같은 확인이 자동으로 반복됩니다. 이 방식은 MCP 명세가 도구 실행 전 사용자 동의와 통제를 요구하는 원칙을 스킬 안에서 코드로 지키는 방법이기도 합니다[S3].

업무 전문가와 IT 담당자가 함께 쓰는 검토 점검표

두 검토 항목은 확인 방법과 통과 기준이 분명하므로 점검표 한 장으로 운영할 수 있습니다. 아래 점검표를 통과한 스킬만 MCP를 거쳐 스킬 업로드로 운영 프로파일에 올리도록 정하면, 운영 환경의 작은 모델이 실행하는 스킬은 모두 같은 설계 기준을 따르게 됩니다.

검토 항목	확인 방법	통과 기준	주 확인자
판단 목록의 코드 이관	SKILL.md의 조건·선택 문장을 진행기 단계와 짝짓기	짝 없는 판단 0건	업무 전문가
확인 게이트의 코드 소유	승인 문구 없이 쓰기 명령을 직접 호출	진행기가 쓰기를 거부하고, 같은 예시가 회귀 시험에 들어 있음	IT 담당자

스킬 작성 지침에 넣는 지시문 문안

세 지시와 두 검토 항목은 조직의 스킬 작성 지침에 고정 문안으로 넣어 두고, 업무 전문가가 프론티어 모델에게 스킬을 요청할 때마다 업무 정의서와 함께 붙이는 방식이 가장 단순합니다. 문안 맨 앞에는 「이 스킬은 활성 파라미터 3B 급 로컬 모델이 실행한다」는 전제를 적어, 프론티어 모델이 자기 수준으로 스킬을 만들지 않게 합니다. 그 아래에는 지시문 해석, 여섯 이관 항목을 명시한 결정적 코드, 질문 예시와 회귀 시험, SKILL.md 길이 규칙을 차례로 적으며, 길이 규칙은 SKILL.md를 짧게 두고 긴 설명을 필요할 때만 여는 참고 파일로 옮기는 점진적 공개 방식을 따릅니다[S2]. 아래 문안은 그 형태를 보인 예이며, 질문 예시 부분은 업무마다 업무 전문가가 채웁니다.

> 이 스킬은 활성 파라미터 3B 급 로컬 LLM이 실행한다. 네 수준으로 만들지 마라.
> 1. 지시문 해석: 요청 첫 줄과 본문으로 부를 명령을 고르는 단계를 맨 앞에 둔다.
> 판별 규칙은 규칙표와 정규식으로 쓰고, 모르면 정해진 질문을 출력하고 멈춘다.
> 2. 결정적 코드: 값 추출, 후보 선정, 순서, 확인 게이트, 멱등 쓰기, 재조회 검증은
> 모두 진행기 스크립트가 한다. 모델은 명령 한 줄을 실행하고 출력을 고치지 않고 전한다.
> 3. 질문 예시: 아래 질문 예시와 업무 시나리오를 모두 처리하게 만들고,
> 예시마다 기대 결과를 묶어 회귀 시험으로 고정한다.
> [질문 예시와 업무 시나리오 — 업무 전문가가 5.3.1 형식으로 채운다]
> SKILL.md 는 60줄 이내로 명령 한 줄과 금지 사항만 적고, 긴 설명은 references/ 로 옮긴다.

이 문안은 한 번 정해 두면 업무가 바뀌어도 그대로 쓰고, 업무마다 달라지는 내용은 질문 예시와 업무 시나리오뿐입니다. 업무 전문가가 새로 준비할 것은 반복 업무 하나에 대한 질문 예시 목록과 요청 문장의 권장 형식이며, 나머지 설계 기준은 지침 문안과 두 검토 항목이 맡습니다. 결과적으로 프론티어 모델이 누가 요청하든 같은 구조의 스킬을 만들고, 그 스킬이 운영 환경의 작은 모델에서 같은 결과를 내는지를 회귀 시험과 점검표로 확인하는 절차가 조직 안에 고정됩니다. 이 세 가지 지시가 응답 시간과 정확도를 실제로 얼마나 바꾸는지는 운영 모델이 놓이는 보급형 GPU의 조건과 함께 6장에서 살펴봅니다.

6장: 보급형 GPU에서 에이전트가 쓸 만해지는 설계

MSAP.ai AAS가 운영 환경에서 쓰는 실행 모델은 활성 파라미터 3B 급 오픈소스 MoE(Mixture of Experts, 전문가 혼합) 모델이고, 이 모델은 보급형 GPU나 소형 AI 서버 한 대에서 vLLM으로 서빙합니다. 업무 에이전트를 사내에 두려는 조직이 먼저 따지는 것은 고가 GPU를 추가로 도입해야 하는가입니다. 토큰 생성 속도는 GPU 연산 능력보다 메모리에서 가중치를 옮기는 속도에 더 크게 좌우되므로[S5], 장비 등급보다 모델 구조와 스킬 설계가 응답 시간을 더 크게 바꿉니다. 게다가 작은 모델은 판단이 많이 섞인 지시를 받으면 도구를 수십 번 부르며 같은 일을 되풀이하기 때문에, 판단을 결정적 코드로 옮겨 모델 호출 횟수 자체를 줄여야 업무에 쓸 수 있는 시간 안에 답이 나옵니다. 이 장에서는 토큰 생성 속도의 원리와 모델 선택 기준, 코드·로컬 LLM·판단 모델·사람의 역할 분담, 코드로 옮긴 뒤에도 남는 공백 세 곳을 훅으로 막는 방법을 차례로 정리해 고가 GPU 증설 없이 업무 에이전트를 운영할 수 있는지 판단하는 근거를 제공합니다.

6.1 토큰 생성 속도를 정하는 메모리 대역폭과 활성 파라미터가 작은 MoE 모델의 이점

LLM 추론은 입력 프롬프트 전체를 한 번에 처리하는 프리필(prefill) 단계와 답을 토큰 하나씩 만들어 내는 디코드(decode) 단계로 나뉩니다[S5]. 업무 에이전트의 응답 시간에서 큰 몫을 차지하는 쪽은 디코드 단계이며, 이 단계의 속도는 모델 전체 크기보다 토큰 하나를 만들 때 메모리에서 읽어야 하는 가중치의 양으로 정해집니다. 보급형 장비에서 모델을 고를 때는 파라미터 총수보다 토큰마다 읽는 파라미터 수를 먼저 확인해야 하고, 이 값을 구조적으로 줄인 설계가 MoE 모델입니다. 아래 두 항은 디코드 단계의 원리와 공개 MoE 모델의 수치를 차례로 다룹니다.

6.1.1 디코드 단계의 메모리 대역폭 한계와 토큰마다 읽는 가중치 양

디코드 단계와 메모리 대역폭

디코드 단계는 직전까지 만든 토큰을 바탕으로 다음 토큰 하나를 계산하는 작업을 답이 끝날 때까지 반복합니다. 한 번 계산할 때 새로 들어오는 입력이 토큰 하나뿐이어서 연산은 행렬과 벡터의 곱이 되고, GPU 연산 장치는 처리할 일이 적어 대부분의 시간을 데이터가 도착하기를 기다리는 데 씁니다. NVIDIA는 이 단계를 메모리 한계(memory-bound) 작업으로 분류하고, 지연을 정하는 요인은 가중치·키·값·활성값을 메모리에서 연산 장치로 옮기는 속도라고 설명합니다[S5]. 토큰 하나를 만들 때마다 모델이 쓰는 가중치를 메모리에서 한 차례씩 읽어야 하므로, 초당 생성 토큰 수의 이론 상한은 메모리 대역폭을 토큰당 읽는 바이트 수로 나눈 값에 가깝습니다.

토큰당 읽는 가중치 양의 어림셈

이 원리를 숫자로 옮기면 장비와 모델의 궁합을 미리 가늠할 수 있습니다. 예를 들어 메모리 대역폭이 초당 273GB 급인 장비에서 dense 27B 모델을 실행하면, 파라미터 하나를 1바이트로 저장할 때 토큰마다 약 27GB를, 4비트로 줄여도 약 13.5GB를 읽어야 하므로 이론 상한은 초당 10~20토큰 사이에 머뭅니다. 반면 같은 장비에서 활성 파라미터가 3B 급인 MoE 모델은 토큰마다 읽는 양이 수 GB로 줄어 이론 상한이 초당 수십 토큰 이상으로 올라갑니다. 실제 속도는 키·값 캐시 읽기, 전문가 라우팅, 서빙 엔진의 부하 때문에 이 상한보다 낮게 나오지만, 두 구조 사이의 상대적 차이는 대체로 유지됩니다.

업무 에이전트에서 두 단계가 늘어나는 경우

두 단계는 속도를 정하는 요인이 다르기 때문에, 에이전트 응답이 느릴 때 어느 단계가 늘어났는지를 먼저 구분해야 대책이 맞습니다. 다음 표는 프리필과 디코드가 업무 에이전트에서 각각 어떤 경우에 길어지는지 정리한 것입니다.

단계	하는 일	속도를 정하는 요인	업무 에이전트에서 길어지는 경우
프리필	시스템 프롬프트·SKILL.md·대화 기록·도구 결과를 한 번에 처리	GPU 연산 능력, 입력 토큰 수	스킬 설명이 길거나 도구 결과가 대화에 계속 쌓일 때
디코드	답과 도구 호출 인자를 토큰 하나씩 생성	메모리 대역폭, 토큰당 읽는 가중치 양	모델이 긴 생각 과정이나 긴 답을 쓸 때
두 단계 반복	도구를 한 번 부를 때마다 프리필과 디코드를 한 차례 더 실행	한 턴의 도구 호출 횟수	모델이 순서를 스스로 정하며 도구를 여러 번 부를 때

출처: 단계 구분과 속도 요인은 NVIDIA 추론 최적화 설명[S5], 업무 에이전트 사례는 MSAP.ai AAS 운영 구성 기준

표의 세 번째 행이 보급형 장비에서 가장 큰 차이를 만듭니다. 도구 호출이 한 번 늘 때마다 늘어난 대화 전체를 다시 프리필하고 새 인자를 다시 디코드해야 하므로, 한 턴에 도구를 30번 부르는 스킬은 같은 모델에서도 한 번 부르는 스킬보다 수십 배 느려집니다. 모델 구조로 디코드 속도를 올리는 일과 스킬 설계로 호출 횟수를 줄이는 일은 함께 다뤄야 하며, 호출 횟수를 줄이는 스킬 설계는 6.3절과 6.4절에서 이어집니다.

모델 선택 기준으로 옮긴 원리

디코드 원리를 장비 선택 기준으로 옮기면 메모리 용량과 메모리 대역폭을 따로 봐야 한다는 결론이 나옵니다. 메모리 용량은 모델의 전체 파라미터와 키·값 캐시를 한 번에 올릴 수 있는지를 정하고, 메모리 대역폭은 토큰마다 읽어야 하는 활성 파라미터를 얼마나 빨리 옮기는지를 정합니다. MoE 모델은 전체 파라미터만큼의 용량이 필요하지만 대역폭 부담은 활성 파라미터만큼만 지므로, 용량은 넉넉하고 대역폭은 크지 않은 보급형 장비와 잘 맞습니다. 장비를 고를 때도 「몇 B 모델이 올라가는가」와 「토큰당 몇 B를 읽는가」를 서로 다른 질문으로 놓고 각각 답을 확인하는 것이 맞습니다.

6.1.2 Mixtral과 Qwen3 공개 사례로 본 전체 파라미터와 활성 파라미터의 차이

Mixtral 8×7B의 전문가 선택 구조

MoE 구조를 대중적으로 알린 공개 모델은 Mistral AI가 2024년 1월에 논문으로 발표한 Mixtral 8×7B입니다. Mixtral 8×7B는 층마다 전문가(expert) 역할을 하는 하위 신경망 8개를 두고, 라우터가 토큰마다 그중 2개를 골라 결과를 합칩니다[S6]. 토큰 하나는 약 47B 파라미터에 접근할 수 있지만 추론 때 실제로 계산에 쓰는 활성 파라미터는 약 13B입니다[S6]. 모델이 담고 있는 지식의 폭은 47B 모델에 가깝고, 토큰마다 메모리에서 읽는 양은 13B 모델에 가깝다는 것이 MoE 구조의 핵심입니다.

Qwen3-30B-A3B의 전문가 128개 구성

활성 비율을 더 낮춘 사례는 Qwen 팀이 공개한 Qwen3-30B-A3B 모델 카드에서 확인할 수 있습니다. 이 모델은 전체 파라미터가 30.5B이고 전문가 128개 가운데 토큰마다 8개만 켜서 활성 파라미터가 3.3B이며, 기본 맥락 길이는 32,768토큰이고 YaRN 방식을 적용하면 131,072토큰까지 늘어납니다[S7]. 이 모델은 생각 모드를 켜고 끄는 설정(`enable_thinking=False` , `/no_think`)을 제공하고 도구 호출 성능을 강점으로 내세웁니다[S7]. 생각 모드를 끄면 디코드해야 할 출력 토큰이 줄어 6.1.1의 원리상 응답이 빨라집니다. Apache-2.0 라이선스로 상업적 이용이 가능합니다. 다만 Qwen3-30B-A3B는 MSAP.ai AAS 운영 환경의 실행 모델과 같은 모델이 아니며, 이 백서에서는 활성 3B 급 MoE라는 범주를 설명하는 공개 사례로만 씁니다.

공개 모델의 활성 파라미터 비교

세 구조를 나란히 놓으면 활성 비율이 장비 부담을 어떻게 바꾸는지 드러납니다. 다음 표는 같은 dense 27B 급 모델을 기준으로 두 공개 MoE 모델과 MSAP.ai AAS 운영 모델 범주를 비교한 것입니다.

모델	구조	전체 파라미터	토큰당 활성 파라미터	활성 비율	근거
dense 27B 급 모델	모든 층이 매 토큰 계산에 참여	약 27B	약 27B	100%	구조상 정의
Mixtral 8×7B	전문가 8개 중 2개 선택	약 47B	약 13B	약 28%	[S6]
Qwen3-30B-A3B	전문가 128개 중 8개 선택	30.5B	3.3B	약 11%	[S7]
MSAP.ai AAS 운영 모델 범주	활성 파라미터 3B 급 오픈소스 MoE 모델	30B 대	3B 급	약 10%	제품 운영 구성

활성 비율이 보급형 장비에 주는 의미

표의 활성 비율이 10% 안팎이면, 토큰당 읽는 양은 3B 급 소형 모델과 비슷하면서도 모델이 담는 지식과 도구 호출 능력은 30B 급에 가깝게 유지됩니다. 30B 급 MoE 모델을 4비트로 양자화하면 가중치가 약 15GB여서 48GB 메모리를 가진 워크스테이션용 GPU 한 장에 키·값 캐시와 함께 올릴 수 있고, 토큰마다 읽는 양은 2GB 안팎으로 줄어듭니다. 이 어림셈이 보급형 GPU를 업무 에이전트 운영 장비로 검토할 수 있는 첫 번째 근거이며, 실제 장비에서 관찰한 속도 변화는 6.2절에서 다룹니다.

6.2 활성 파라미터 3B 급 오픈소스 MoE 모델로 바꾼 보급형 GPU의 응답 속도 변화

MSAP.ai AAS의 운영 환경은 Hermes Agent 기반 실행 환경, 사내 모델 게이트웨이인 LiteLLM, 로컬 LLM을 서빙하는 vLLM으로 구성합니다. 이 구성에서 같은 장비에 올린 모델을 dense 구조에서 활성 3B 급 MoE 구조로 바꾸면 토큰 생성 속도가 약 3배 빨라지는 동작이 관찰됩니다. 다만 이 수치는 한 업무 에이전트를 운영하며 관찰한 근사치이고 공개 벤치마크가 아니므로, 장비·양자화 방식·동시 요청 수가 다른 조직은 같은 방법으로 자기 환경에서 다시 재야 합니다. 속도 변화와 함께 서빙 설정이 작은 모델의 실행 안정성을 어떻게 바꾸는지도 이 절에서 다룹니다.

6.2.1 같은 소형 AI 서버에서 dense 모델과 활성 3B 급 MoE 모델의 토큰 생성 속도 비교

같은 장비에서 바꾼 모델 구조

한 업무 에이전트를 MSAP.ai AAS 구성으로 운영하면서 메모리 대역폭 273GB/s 급 소형 AI 서버에 올린 실행 모델을 dense 27B 급에서 활성 3B 급 MoE로 바꿨을 때, 생성 속도는 아래 표와 같이 달라졌습니다. 장비 외 조건은 그대로 두고 모델 구조만 바꾼 비교이므로, 차이는 대부분 6.1절에서 설명한 토큰당 읽는 가중치 양의 차이에서 나옵니다. 표의 생성 시간은 초당 토큰 수에서 거꾸로 계산한 어림값입니다.

항목	dense 27B 급 모델	활성 3B 급 MoE 모델
토큰당 읽는 파라미터	약 27B	약 3B
초당 생성 토큰 (근사치)	약 12토큰	약 37~44토큰
100토큰 도구 호출 인자 생성 시간 (어림)	약 8초	약 2~3초
300토큰 답변 생성 시간 (어림)	약 25초	약 7~8초

주: 한 업무 에이전트를 MSAP.ai AAS 구성으로 운영했을 때 관찰된 동작의 근사치이며 외부 벤치마크가 아닙니다. 장비·양자화·동시 요청 수에 따라 달라집니다.

속도 차이와 업무 응답 시간의 관계

초당 12토큰과 40토큰의 차이는 답 하나만 보면 몇 초 정도지만, 도구 호출 횟수를 곱하면 업무 응답 시간에서 분 단위 차이가 됩니다. 한 턴에 도구를 15번 부르고 호출마다 100토큰의 인자를 쓰는 스킬은 dense 모델에서 디코드에만 2분 안팎이 들고, MoE 모델로 바꿔도 40초 안팎이 남습니다. 여기에 호출마다 늘어난 대화를 다시 프리필하는 시간이 더해지므로, 모델 구조를 바꾸는 것만으로는 수 분 걸리던 업무가 몇 초 안에 끝나지 않습니다. 따라서 MoE 전환은 필요조건이고, 호출 횟수를 1회 안팎으로 줄이는 스킬 설계가 함께 있어야 사람이 기다릴 수 있는 응답 시간이 나옵니다.

속도와 함께 확인할 정확도

활성 파라미터가 작은 모델은 빠른 대신 여러 단계를 스스로 추론하는 능력이 큰 모델보다 약하므로, 속도만 보고 모델을 고르면 정확도에서 손해를 봅니다. NVIDIA 연구진은 에이전트 안의 언어 모델이 적은 수의 전문 작업을 거의 변화 없이 반복하므로 작은 언어 모델이 대부분의 에이전트 호출에 충분하고 더 경제적이라는 입장을 입장 논문으로 밝혔습니다[S4]. 이 주장은 실험으로 증명된 결론이 아니라 설계 방향에 대한 주장이지만, 모델에게 맡기는 일을 좁고 반복적인 작업으로 줄이면 작은 모델로도 충분하다는 방향은 MSAP.ai AAS의 스킬 설계와 같습니다. 작은 모델의 정확도를 정하는 것은 모델 자체보다 모델에게 무엇을 맡기느냐이며, 그 기준은 6.3절의 역할 분담에서 구체화합니다.

6.2.2 vLLM OpenAI 호환 서빙과 요청당 도구 호출을 한 번 이하로 묶는 구성

vLLM OpenAI 호환 서버의 역할

vLLM은 `/v1/completions`, `/v1/chat/completions` 같은 OpenAI 호환 경로를 제공하는 서빙 엔진이고, Chat Completions 형식으로 도구 호출을 지원하며 이 형식을 쓰려면 모델별 채팅 템플릿이 필요합니다[S10]. Hermes Agent와 LiteLLM은 모두 OpenAI 형식으로 모델을 부르므로, vLLM으로 로컬 모델을 서빙하면 실행 환경 쪽 코드를 바꾸지 않고 모델만 교체할 수 있습니다. LiteLLM은 100개 넘는 LLM 제공자와 MCP 도구

를 OpenAI 호환 주소 하나로 묶는 자체 구축형 게이트웨이이며, 팀별 가상 키·예산·속도 제한, 배포 간 부하 분산과 장애 시 우회, 호출 로그 전송을 제공합니다[S9]. MSAP.ai AAS 운영 환경의 모든 모델 호출은 LiteLLM 게이트웨이 하나를 거쳐 vLLM의 로컬 LLM에 닿고, 고객 정보나 금액 자료가 조직 밖 모델로 나가지 않습니다.

병렬 도구 호출을 끄는 설정

vLLM에서 요청에 `parallel_tool_calls=false`를 지정하면 요청 하나당 도구 호출이 0회나 1회로 제한되며, 여러 호출을 한 번에 내보낼지는 원래 모델이 정합니다[S10]. 큰 모델에서는 병렬 호출이 시간을 줄여 주지만, 작은 모델은 여러 호출을 한꺼번에 내보낼 때 순서를 잘못 잡거나 앞 결과를 보지 않고 다음 인자를 짐작하는 실수를 자주 합니다. 반면 요청당 호출을 1회 이하로 묶으면 모델은 한 번에 하나의 명령만 결정하면 되고, 스킬 안의 진행기(스킬의 처리 순서를 코드로 진행하는 스크립트)가 그 명령을 받아 다음 단계를 정합니다. 이 설정은 6.3절의 「모델은 한 줄 실행」 원칙을 서빙 계층에서 강제하는 장치로 쓰입니다.

작은 모델 운영을 위한 서빙 구성 요소

다음 표는 MSAP.ai AAS 운영 환경에서 작은 모델을 안정적으로 쓰기 위해 함께 맞추는 구성 요소를 정리한 것입니다.

구성 요소	맡는 일	작은 모델 운영에서 얻는 효과	근거
vLLM OpenAI 호환 서버	로컬 LLM 서빙, 채팅 템플릿 적용, 도구 호출 형식 처리	실행 환경 코드 수정 없이 모델 교체	[S10]
`parallel_tool_calls=false`	요청당 도구 호출 0~1회	순서 오류와 인자 짐작 감소	[S10]
생각 모드 끄기	실행 단계에서 생각 과정 출력 생략	디코드 토큰 감소로 응답 단축	[S7]
LiteLLM 게이트웨이	모든 모델 호출의 단일 주소, 가상 키·예산·우회·로그	모델 교체와 비용 집계를 한 곳에서 관리	[S9]
Hermes Agent 프로파일	스킬·도구·플러그인 훅 실행	업무별로 스킬과 훅을 분리해 운영	[S8]

> 🎯 *결정 포인트 보급형 GPU 도입 여부는 장비 단가보다 「활성 3B 급 MoE 모델이 초당 수십 토큰을 내는가」와 「스킬이 턴당 도구 호출을 1회 안팎으로 유지하는가」 두 가지로 판단합니다. 앞의 조건은 모델 선택과 서빙 설정으로, 뒤의 조건은 6.3절과 6.4절의 스킬 설계로 충족합니다.*

6.3 판단은 코드, 모델은 한 줄 실행 — 코드와 로컬 LLM, 판단 모델, 사람의 역할 분담

Anthropic은 LLM 시스템을 미리 정한 코드 경로로 모델과 도구를 엮는 워크플로와 모델이 스스로 순서와 도구 사용을 정하는 에이전트로 나누고, 가장 단순한 해법부터 찾아 결과가 실제로 좋아질 때만 복잡도를 더하라고 권합니다(Building effective agents)[S1]. MSAP.ai AAS의 스킬 설계는 이 구분에서 워크플로 쪽으로 무게를 옮긴 형태이며, 판단·검증·쓰기는 결정적 코드가 맡고 모델은 사람의 말을 받아 명령 한 줄을 실행하는 입구 역할

만 맡습니다. 이렇게 나누면 「AI가 필요 없는 일은 그냥 코드로 처리하면 되지 않는가」라는 반론이 자연스럽게 나오는데, 실제로 업무 처리의 대부분은 코드로 옮기는 것이 맞고 남는 것은 코드만으로 해석할 수 없는 입력입니다. 두 항은 네 주체의 역할표와 판단 모델을 어디에 둘지 정한 비교 결과를 다룹니다.

6.3.1 사람 말을 받는 입구는 모델, 판단과 쓰기는 코드, 최종 확인은 사람인 역할표

네 주체의 역할 분담표

MSAP.ai AAS 스킬에서 일을 나눠 맡는 주체는 결정적 코드, 로컬 LLM, 판단 모델, 사람의 넷입니다. 여기서 판단 모델은 실행 모델과 별도로 두는 모델로, 코드가 만든 결과가 사용자의 요청을 충족했는지와 실패했다면 어떤 유형인지를 판정하는 일만 맡습니다. 다음 표는 각 주체가 맡는 일과 맡지 않는 일, 그리고 책임 경계를 정리한 것입니다.

주체	맡는 일	맡지 않는 일	책임 경계
결정적 코드	첫 줄 라우팅 규칙, 값 추출, 조회, 후보 선정, 순서 진행, 확인 게이트, 멱등 쓰기, 재조회 검증	자유 문장 해석	같은 입력에 같은 결과를 내고 시험으로 고정
로컬 LLM (활성 3B 급 MoE)	요청 받기, 진행기 명령 한 줄 실행, 결과 원문 전달	업무 판단, 결과 고쳐 쓰기, 스킬 수정	명령을 정확히 부르고 원문을 그대로 돌려줌
판단 모델	사용자의 자유 답 해석, 결과가 요청을 충족했는지와 실패 유형 판정	업무 시스템 쓰기	판정만 내고 재시도 여부는 코드가 정함
사람 (업무 전문가·담당자)	업무 규칙 정의, 쓰기 전 최종 확인, 예외 처리	반복 조회와 입력	확인 게이트에서 한 번 승인

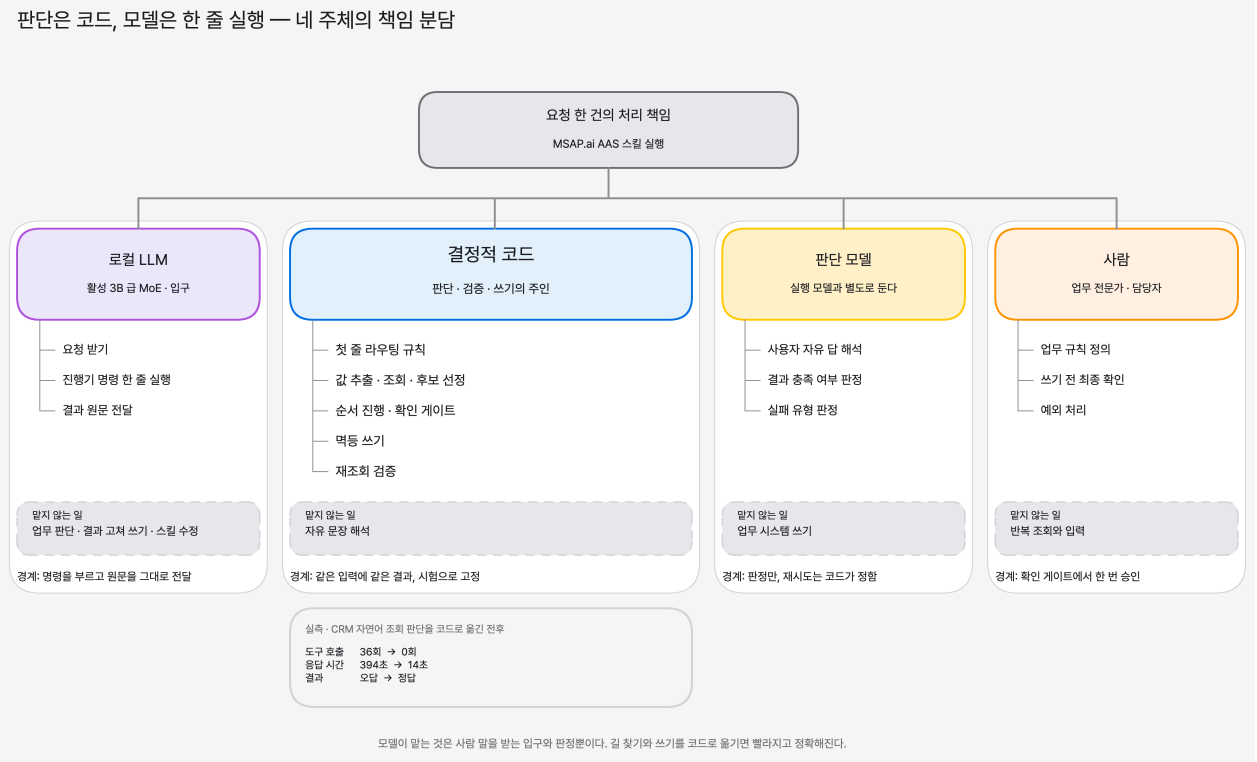


그림 6-1. 판단은 코드, 모델은 한 줄 실행 — 코드·로컬 LLM·판단 모델·사람이 맡는 일

코드만으로 처리하지 못하는 입력

반론에 답하려면 코드만으로 처리하지 못하는 입력이 무엇인지를 먼저 좁혀야 합니다. 첫째는 「이번 달 ○○ 기관 건 상태 알려 줘」처럼 형식 없이 들어오는 사람의 요청이고, 둘째는 진행기가 「후보가 둘입니다, 어느 쪽입니까」라고 물었을 때 사용자가 「두 번째 거, 근데 금액은 지난번 기준으로」처럼 돌려주는 자유 답이며, 셋째는 만들어진 초안이 원래 요청을 충족했는지에 대한 판정입니다. 이 세 가지는 규칙으로 모두 적기 어려워 모델이 필요하지만, 나머지 처리인 값 추출·조회·순서 진행·쓰기는 규칙이 분명해 코드가 더 빠르고 정확합니다. 한 업무 에이전트에서 CRM 자연어 조회의 판단을 코드로 옮겼을 때 도구 호출이 36회에서 0회로, 응답이 394초에서 14초로 줄고 오답이 정답으로 바뀌었으며, 이 사례의 전후 과정은 7.1절에서 다룹니다.

코드를 쓰는 주체와 실행하는 주체의 분업

반론과 MSAP.ai AAS 설계가 갈라지는 부분은 그 코드를 누가 쓰느냐입니다. 업무 전문가가 말로 정의한 업무를 프론티어 AI가 MCP로 MSAP.ai AAS에 붙어 지시문 해석 단계, 결정적 코드, 질문 예시를 담은 스킬로 만들고, 운영 환경의 작은 모델은 완성된 스킬을 실행하기만 합니다. NVIDIA 연구진도 범용 대화가 꼭 필요한 곳에만 큰 모델을 쓰고 반복 작업은 작은 모델이 맡는 이종 구성을 자연스러운 형태로 봅니다[S4]. 큰 모델은 코드를 만드는 개발 환경에서 한 번 쓰고, 작은 모델은 그 코드를 매일 실행하는 운영 환경에서 쓰는 배치가 비용과 정확도를 함께 맞추는 방법입니다.

6.3.2 스킬 고르기보다 실행이 병목인 모델 배치 — 판단 모델을 더해도 이득이 작았던 비교

스킬 고르기에 판단 모델을 더한 비교

판단 모델을 어디에 둘지 정하려면, 모델을 더했을 때 정확도가 얼마나 오르고 시간이 얼마나 늘어나는지를 단계별로 재야 합니다. 한 업무 에이전트에서 요청에 맞는 스킬을 고르는 단계에 판단 모델을 더해 비교한 결과는 다

음 표와 같습니다.

구성	스킬 고르기 정확도	고르기 응답 시간
로컬 LLM이 선택만 수행	약 86%	1초 미만
로컬 LLM 선택 + 판단 모델 검토	약 89%	약 20초

주: 한 업무 에이전트를 MSAP.ai AAS 구성으로 운영했을 때 관찰된 동작의 근사치입니다.

정확도가 2%p 남짓 올랐지만 응답 시간은 20배 넘게 늘었으므로, 스킬 고르기 단계에는 판단 모델을 넣지 않는 것이 맞습니다. 게다가 업무 요청의 대부분은 첫 줄 규칙만으로 결정적 코드가 스킬을 고를 수 있어, 모델이 고르기를 맡는 경우 자체가 많지 않습니다. 반면 같은 에이전트에서 응답이 수 분씩 걸리고 틀린 결과가 나온 원인을 추적하면 대부분 스킬을 고른 뒤의 실행 단계, 즉 모델이 도구를 여러 번 부르며 순서를 스스로 정하는 과정에 있었습니다. 따라서 병목은 고르기가 아니라 실행이며, 정확도를 올리는 투자는 실행 단계의 판단을 코드로 옮기는 쪽에 먼저 가야 합니다.

판단 모델을 두는 단계와 두지 않는 단계

판단 모델은 시간을 들인 만큼 이득이 큰 단계에만 둡니다. 판단 모델은 결과 글만 보고 실제로 업무 시스템에 무엇이 만들어졌는지는 볼 수 없으므로, 생성 여부는 코드가 먼저 재조회로 확인하고 그다음에 판단 모델이 요청 충족 여부와 실패 유형을 판정합니다. 판정이 실패면 코드가 재시도를 한 번 하거나 사람에게 넘기며, 이 검증은 마지막 확인 직전뿐 아니라 첫 초안 단계에서도 실행합니다(견적 초안 사례는 7.3절). 다음 표는 단계별로 판단 모델 투입 여부를 정리한 모델 배치표입니다.

단계	맡는 주체	판단 모델 투입	이유
첫 줄 판별·스킬 고르기	결정적 코드 규칙, 규칙에 없으면 로컬 LLM	넣지 않음	정확도 이득에 비해 응답 시간이 20배 이상 늘어남
값 추출·조회·멱등 쓰기	결정적 코드	넣지 않음	규칙이 분명해 코드가 같은 결과를 냄
사용자 자유 답 해석	판단 모델	넣음	규칙으로 적기 어려운 문장 해석
결과 충족 판정	코드가 생성 여부 확인 뒤 판단 모델	넣음	초안이 요청과 어긋난 경우를 쓰기 전에 걸러냄
쓰기 전 최종 확인	사람	해당 없음	업무 책임은 사람이 짐

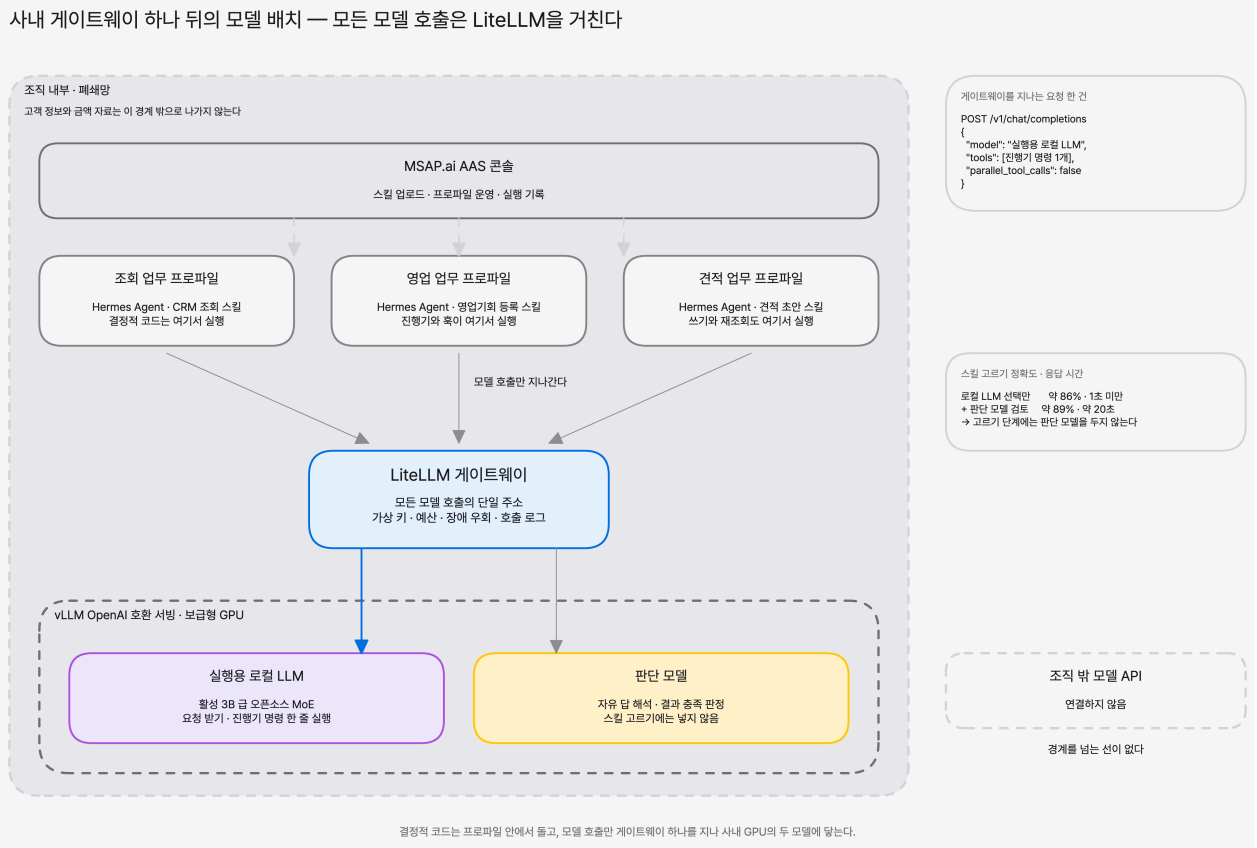


그림 6-2. 사내 게이트웨이 하나 뒤의 모델 배치 — 모든 모델 호출은 LiteLLM을 거친다

6.4 코드 이관 뒤에도 남는 공백 세 곳과 모델 호출 전·도구 호출 전·출력 직후 훅의 차단

5.2절의 방식대로 판단을 결정적 코드로 옮긴 스킬도 실행 모델이 작으면 세 군데에서 의도와 다르게 움직입니다. 첫째는 요청에 맞지 않는 스킬로 들어가는 라우팅 문제, 둘째는 진행기가 만든 문구를 모델이 고쳐 써서 사용자에게 전하는 전달 충실도 문제, 셋째는 대화가 끝난 뒤 배경 검토 과정에서 모델이 스킬 파일 자체를 고치는 문제입니다. 세 번째 문제는 Hermes Agent의 설계에서 나오는데, Hermes Agent는 에이전트가 `skill_manage` 도구로 스킬을 스스로 만들고 고치고 지울 수 있게 하고 이를 절차 기억이라고 부릅니다(Hermes Agent 스킬 문서)[S8]. MSAP.ai AAS는 이 세 공백을 Hermes Agent 프로파일의 플러그인 훅으로 막으며, 훅이 실행되는 시점은 모델 호출 전, 도구 호출 전, 모델 출력 직후의 세 가지입니다.

공백 세 곳과 훅의 대응

세 공백은 원인과 증상이 서로 다르므로 막는 훅도 각각 다릅니다. 다음 표는 공백마다 어떤 증상이 나타나고 어느 시점의 훅이 무엇을 하는지 정리한 것입니다.

공백	훅이 없을 때 나타나는 증상	막는 훅	훅이 하는 일
라우팅	본문에 나온 다른 업무 낱말 때문에 엉뚱한 스킬을 고름	모델 호출 전 훅	요청 첫 줄을 등록된 명령과 대조해 맞으면 진행기를 바로 실행
전달 충실도	진행기의 질문·수치·확인 문구를 모델이 줄이거나 바	출력 직후 훅	모델 답을 진행기 원문으로 교체

공백	훅이 없을 때 나타나는 증상	막는 훅	훅이 하는 일
	꿔 전함		
배경 검토의 스킬 수정	대화 뒤 모델이 SKILL.md 나 스크립트를 고쳐 확인 게이트가 사라짐	도구 호출 전 훅	스킬 디렉터리 쓰기를 차단하고 원본 매니페스트로 복원



그림 6-3. 요청 한 턴에서 모델 호출 전·도구 호출 전·출력 직후 훅이 하는 일

6.4.1 첫 줄 우선 라우팅과 모델이 고쳐 쓴 답을 진행기 원문으로 바꾸는 출력 훅

본문 낱말 때문에 잘못 고른 스킬

Agent Skills 방식에서는 스킬의 이름과 설명만 먼저 시스템 프롬프트에 올리고, 모델이 관련 있어 보이는 스킬을 골라 본문을 읽습니다[S2]. 이 방식은 큰 모델에서 맥락을 아끼는 장점이 있지만, 스킬이 수십 개로 늘어난 프로파일에서 작은 모델은 요청 본문에 나온 낱말에 이끌려 다른 스킬을 고르는 실수를 자주 합니다. 예를 들어 첫 줄이 「견적」인 요청 본문에 영업기회 이름과 기관명이 함께 들어 있으면, 작은 모델은 견적 스킬 대신 영업기회 등록 스킬을 열 수 있습니다. 그래서 MSAP.ai AAS는 업무 전문가가 정한 첫 줄 명령어를 등록부에 두고, 모델 호출 전 훅이 요청 첫 줄을 등록부와 대조해 맞는 명령이 있으면 모델에게 고르게 하지 않고 해당 진행기를 바로 실행합니다.

진행기 원문으로 바꾸는 출력 훅

라우팅이 맞아도 작은 모델은 진행기가 만든 결과를 사용자에게 전하는 과정에서 문구를 고쳐 쓰는 경향이 있습니다. 진행기의 「후보 두 건 중 어느 쪽입니까」라는 질문을 자기 말로 바꾸면서 후보 번호를 빼거나, 금액과 날짜를 요약하면서 자릿수를 틀리거나, 진행기가 묻지 않은 확인을 덧붙이는 식입니다. 이때 출력 직후 훅은 이번 턴에 진행기가 낸 원문이 있으면 모델이 쓴 답을 버리고 원문으로 교체해 사용자에게 보냅니다. 사용자가 받는 질문과 결과는 언제나 시험으로 고정한 진행기의 문구이며, 모델이 바꿀 수 있는 내용이 남지 않습니다.

라우팅과 전달 충실도의 전후 대응

두 훅은 모델이 판단할 여지를 줄여 결과를 고정할 뿐 아니라, 모델 호출 수도 함께 줄입니다. 다음 표는 두 공백에 대해 훅이 없을 때와 적용한 뒤의 동작을 비교한 것입니다.

항목	훅이 없을 때	훅 적용 후
스킬 선택	모델이 스킬 목록과 본문을 읽고 고름	첫 줄이 등록된 명령이면 코드가 바로 진행기를 실행
턴당 도구 호출	모델이 순서를 정하며 여러 번 호출	진행기 실행 1회 또는 0회
사용자에게 가는 문구	모델이 요약·변형한 문장	진행기 원문 그대로
결과 재현성	같은 요청에도 답이 달라짐	질문 예시 회귀 시험과 같은 결과

한 업무 에이전트의 진행기에 두 훅을 적용하기 전에는 한 세션에서 도구가 41번 호출되고 잘못된 등록이 생겼지만, 적용한 뒤에는 모델의 도구 호출이 0회가 되고 턴당 응답이 4~29초 안에 끝났습니다. 라우팅과 전달을 코드가 쥐면 6.2절에서 본 「턴당 도구 호출 1회 안팎」 조건이 스킬 작성자의 주의에 기대지 않고 구조로 지켜집니다.

6.4.2 배경 검토가 스킬을 고치지 못하게 막는 도구 호출 전 훅과 원본 복원

배경 검토가 스킬을 고치는 문제

Hermes Agent는 대화가 끝난 뒤 그 대화에서 배운 것을 스킬에 반영하는 배경 검토를 수행하며, 이때 모델은 `skill_manage` 도구로 스킬 디렉터리의 파일을 고칠 수 있습니다[S8]. 큰 모델에게 이 기능은 경험을 쌓는 장치지만, 작은 모델은 한 번의 예외적인 대화에서 잘못된 교훈을 끌어내 SKILL.md의 확인 게이트를 지우거나 진행기 대신 직접 도구를 부르라는 문장을 적어 넣기도 합니다. 이렇게 바뀐 스킬은 다음 요청부터 시험을 거치지 않은 상태로 실행되므로, 업무 전문가가 검증해 스킬 업로드로 들여온 스킬이 운영 중에 조용히 다른 스킬이 됩니다. Hermes Agent는 선택 설정 `skills.write_approval`로 스킬 쓰기를 사람 검토 대기로 전환하는 방법을 제공하지만[S8], 이 설정만으로는 이미 바뀐 파일을 되돌리지 못합니다.

도구 호출 전 훅의 차단과 매니페스트 복원

그래서 MSAP.ai AAS는 차단과 복원을 함께 둡니다. 도구 호출 전 훅은 모델이 부르려는 도구와 대상 경로를 검사해 스킬 디렉터리에 대한 쓰기·수정·삭제 시도이면 실행을 막고, 모델에게 스킬은 운영 중에 고칠 수 없다는 결과를 돌려줍니다. 또한 스킬을 올릴 때 파일 목록과 해시를 담은 원본 매니페스트를 함께 저장하고, 실행 전 대조에서 파일이 매니페스트와 다르면 원본으로 복원합니다. 한 업무 에이전트에서는 운영 중 배경 검토가 스킬 수정을 세 번 시도했고 세 번 모두 도구 호출 전 훅에서 막혔습니다.

단계	하는 일	주체
1. 스킬 업로드	검증을 마친 스킬과 파일별 해시를 담은 매니페스트를 함께 등록	업무 전문가, MSAP.ai AAS
2. 도구 호출 검사	모델이 부르려는 도구의 대상 경로가 스킬 디렉터리인지 확인	도구 호출 전 훅

단계	하는 일	주체
3. 차단	스킬 쓰기·수정·삭제 시도를 막고 막은 이유를 모델에 반환	도구 호출 전 훅
4. 변조 확인	스킬 파일을 매니페스트 해시와 대조	결정적 코드
5. 원본 복원	다른 파일이 있으면 매니페스트 기준 원본으로 되돌리고 기록	결정적 코드
6. 개선 요청 반영	고칠 필요가 있으면 개발 환경에서 다시 만들고 시험한 뒤 재업로드	업무 전문가, 프론티어 AI

스킬 변경 경로를 개발 환경 하나로 고정

차단과 복원은 스킬을 고치지 말자는 장치가 아니라, 고치는 경로를 개발 환경 하나로 모으는 장치입니다. 운영에서 드러난 개선점은 업무 전문가가 개발 환경으로 가져가 프론티어 AI와 함께 스킬을 고치고, 질문 예시 회귀 시험을 다시 통과한 뒤 스킬 업로드로 운영 환경에 들여옵니다. 이렇게 하면 운영 환경의 스킬은 언제나 시험을 통과한 판본 하나로 유지되고, 활성 3B 급 오픈소스 MoE 모델은 그 판본을 실행하는 일만 맡습니다. 따라서 이 장의 판단 질문에 대한 답은 조건부 긍정입니다. 보급형 GPU와 활성 3B 급 MoE 모델로도 업무 에이전트를 운영할 수 있지만, 그 전제는 판단을 결정적 코드로 옮긴 스킬과 세 시점의 훅이 함께 갖춰져 있는 것입니다. 운영 중 스킬 변조를 위험 관리 관점에서 다루는 내용은 9.2절에 있습니다.

통제 수단으로 세지 않는 아웃바운드 웹훅

훅을 고를 때는 이름이 비슷한 아웃바운드 웹훅과 구분해야 합니다. Hermes Agent의 아웃바운드 웹훅은 실행 수명주기의 사건을 바깥 시스템으로 알리는 장치로, 도구 호출을 막지 못하고 모델에 맥락을 넣지도 못하며 받는 쪽의 응답 본문도 반영하지 않습니다. 따라서 스킬 수정 차단이나 원문 교체처럼 실행을 바꿔야 하는 통제는 플러그인 훅이나 셸 훅으로 걸고, 웹훅은 실행 사실을 기록·알림 시스템에 남기는 관찰 수단으로만 셉니다. 보안 담당자가 통제 장치 목록을 검토할 때도 웹훅을 차단 항목에 넣지 않아야 점검표가 실제 통제 범위와 일치합니다.

7장: 업무 전문가의 스킬 작성 예시

5장의 세 가지 지시와 6장의 역할 분담에 따라 MSAP.ai AAS로 만든 업무 스킬이 실제로 어떻게 동작하는지는 업무 하나를 정의부터 운영까지 따라가 보면 가장 분명하게 보입니다. 이 장은 영업 부서의 세 업무인 CRM 자연어 조회, 영업기회 등록, 견적 초안 검증을 예로 들어, 업무 전문가가 업무를 문장으로 정의하고 프론티어 모델이 그 문장을 결정적 코드 중심의 스킬로 옮긴 뒤 활성 파라미터 3B 급 오픈소스 MoE 모델이 운영하는 흐름을 보여 줍니다. 세 업무를 고른 이유는 에이전트 안의 언어 모델이 맡는 일이 대부분 적은 수의 전문 작업을 거의 같은 모양으로 반복하는 일이라는 관찰과 맞기 때문입니다[S4]. 세 업무 모두 처음에는 작은 모델에게 스킬 문서만 주었을 때 응답이 몇 분씩 걸리고 결과가 틀렸으며, 판단을 코드로 옮긴 뒤에야 속도와 정확도가 함께 좋아졌습니다. 사례의 수치는 한 업무 에이전트를 MSAP.ai AAS 구성으로 운영했을 때 관찰된 제품 동작이므로, 다른 조직의 업무에 그대로 옮겨 쓸 외부 벤치마크 값으로 보아서는 안 됩니다. 마지막 7.4는 같은 방법을 서버 업그레이드와 보안 공지 반영 같은 IT 운영 업무로 넓히는 설계안이며 실측 수치가 없습니다. 각 사례를 우리 부서의 반복 업무와 견주어 보면 어떤 업무가 같은 모양인지, 파일럿 대상으로 무엇을 고를지 판단할 근거를 얻을 수 있습니다.

7.1 CRM 자연어 조회 스킬 — 오답 394초에서 정답 14초로 바뀐 업무 정의와 결정적 코드 이관

CRM 자연어 조회는 영업 담당자가 평소 말투로 고객 기관의 영업 현황을 묻고, 에이전트가 CRM 기록과 영업 문서에서 답을 찾아 돌려주는 업무입니다. 조회는 데이터를 바꾸지 않으므로 위험이 낮아 보이지만, 작은 모델에게 어떤 도구를 어떤 순서로 쓸지 맡기면 가장 많은 시간과 입력 토큰을 쓰는 업무이기도 합니다. Anthropic의 에이전트 설계 연구는 LLM과 도구를 미리 정한 코드 경로로 엮은 시스템을 워크플로, 모델이 스스로 순서와 도구를 정하는 시스템을 에이전트로 나누고, 가장 단순한 해법부터 쓰라고 권합니다[S1]. 이 사례는 조회 업무의 무게를 에이전트 쪽에서 워크플로 쪽으로 옮겼을 때 무엇이 달라지는지를 보여 줍니다.

7.1.1 업무 전문가가 쓴 조회 업무 문장과 결정적 조회 코드로 옮긴 판단

영업 담당자의 질문 꼴을 적은 조회 업무 정의

업무 전문가가 정의한 내용은 질문의 꼴과 답의 조건이었고, 코드는 한 줄도 쓰지 않았습니다. 질문은 「○○기관 영업기회 지금 어느 단계야」, 「○○ 사업 견적서 어디 있어」처럼 기관 이름과 알고 싶은 내용이 한 문장에 함께 들어오고, 기관 이름은 약칭이나 연도가 붙은 형태로 바뀌어 들어올 수 있다고 적었습니다. 답에는 CRM에 기록된 값을 그대로 쓰고 문서 링크를 붙이며, 이름이 맞는 기관이 여럿이면 후보를 보여 주고 다시 묻는다는 조건을 달았습니다. 업무 전문가는 「무엇을 묻는가」와 「무엇이 정답인가」를 문장으로 정했고, 어떤 도구를 부를지는 정하지 않았습니다. 이 정의서가 프론티어 모델에게 넘어가 스킬과 코드로 바뀌었습니다.

조회 판단을 코드로 옮긴 의도 규칙과 이름 색인

프론티어 모델은 이 정의서를 받아 판단 다섯 가지를 코드로 옮겼습니다. 질문 의도는 영업기회 진행 상태, 견적 문서 위치, 계약 상대 같은 일곱 가지 규칙으로 나눠 코드가 판별하고, 기관 이름은 CRM 기록을 미리 읽어 만든 이름 색인에서 연도·날짜·말머리를 뗀 유사도로 찾습니다. 이름 색인은 MSAP.ai AAS의 예약 작업이 매시간 새로 만들기 때문에, 질문이 들어올 때마다 모델이 CRM 화면을 뒤질 필요가 없습니다. 모델 호출 전 훅은 이 조회 코드를 먼저 실행해 결과를 대화에 넣고, 출력 직후 훅은 모델이 빠뜨린 값을 코드 결과로 되돌립니다(6.4). 이렇

게 하면 코드가 같은 입력에 늘 같은 결과를 내므로, 스크립트가 주는 결정적 신뢰성을 조회 업무가 그대로 얻습니다[S2].

업무 전문가가 정한 문장	코드로 옮긴 판단	모델에게 남은 일
기관 이름이 조금 달라도 같은 곳을 찾아야 한다	이름 색인 조회, 연도·날짜·말머리를 뗀 유사도 비교	없음
무엇을 묻는지는 질문 문장으로 안다	의도 규칙 7종 판별	없음
여러 곳이 맞으면 후보를 보여 주고 묻는다	후보 수 판정, 정해진 질문 문장 출력	질문 문장을 그대로 전달
답에는 CRM 기록과 문서 링크를 붙인다	결과 양식 생성, 링크 조회	결과를 그대로 전달
새 표현을 알려 주면 다음부터 알아듣는다	별칭 목록 추가, 회귀 문항 53건으로 고정	없음

회귀 문항 53건으로 고정한 질문 예시

마지막 행의 회귀 문항은 업무 전문가가 준 질문 예시를 시험으로 바꾼 것입니다. 새 표현이 나오면 별칭 목록과 회귀 문항에 한 줄씩 더하고, 스킬을 고칠 때마다 53건을 다시 실행해 이전에 맞던 질문이 틀리지 않았는지 확인합니다. 같은 연구도 도구 설명에 사용 예와 경계 사례를 담고 여러 예시 입력으로 시험해 모델의 실수를 찾아 고치라고 권합니다[S1]. 이렇게 쌓인 질문 예시는 스킬을 만든 뒤에도 남아 운영 중 변경을 승인하는 기준 묶음이 됩니다. 조회 업무는 쓰기가 없어 회귀 시험을 자주 실행해도 부담이 적으므로, 질문 예시를 쌓는 첫 업무로 고르기에 알맞습니다.

7.1.2 도구 호출과 입력 토큰이 함께 줄어든 전후 비교표

스킬 문서만 준 구성과 코드로 옮긴 구성의 비교

같은 질문 하나를 두 구성으로 처리한 결과는 아래 표와 같습니다. 왼쪽은 스킬 문서만 넣고 모델이 도구를 골라 엮게 한 구성이고, 오른쪽은 조회 판단을 코드로 옮기고 모델 호출 전 훅이 결과를 먼저 넣은 구성입니다. 두 값 모두 한 업무 에이전트에서 MSAP.ai AAS 채팅으로 실제 질문을 넣어 관찰한 제품 동작입니다.

항목	스킬 문서만 준 구성	판단을 코드로 옮긴 구성
응답 시간	394초	14초
도구 호출	36회(그중 브라우저 조작 26회)	0회
입력 토큰 소비	약 127만 토큰	도구 결과가 쌓이지 않아 한 턴 분량만 남음
결과	오답	정답

도구 호출 36회가 입력 토큰 127만으로 불어난 원인

입력 토큰이 이만큼 불어난 원인은 도구 호출이 반복될 때마다 앞선 도구 결과가 다음 모델 호출의 입력으로 다시 들어가기 때문입니다. 브라우저 조작 26회는 CRM 화면을 열고 찾고 다시 여는 과정이었고, 화면 내용이 매

번 대화에 쌓였습니다. 토큰을 하나씩 만드는 디코드 단계는 계산보다 가중치와 키·값을 메모리에서 옮기는 속도가 지연을 정하는 메모리 한계 작업이므로, 입력이 길어지면 같은 장비에서도 응답이 눈에 띄게 늦어집니다[S5]. 즉 394초 가운데 대부분은 모델에게 길 찾기를 맡겨서 생긴 시간입니다. 길 찾기를 코드로 옮기자 도구 호출이 0회가 되었고, 모델은 코드가 넣은 결과를 한 번 읽고 전달하는 일만 맡았습니다.

속도와 정답률이 함께 좋아진 이유와 해석의 범위

이 사례에서 속도와 정확도는 서로 맞바꾼 값이 아니었습니다. 모델이 고르던 도구 순서와 이름 비교를 코드가 맡자 틀릴 수 있는 단계 자체가 사라졌고, 그 결과 시간이 줄면서 오답도 정답으로 바뀌었습니다. 작은 언어 모델이 에이전트 호출 대부분에 충분하고 더 경제적이라는 주장은 이런 반복 작업을 전제로 합니다[S4]. 다만 이 비교는 한 에이전트의 한 질문에서 본 값이며, 질문 꼴이 정의서 범위를 벗어나면 코드는 정해진 질문을 출력하고 멈춥니다. 따라서 조회 업무를 파일럿으로 고를 때는 응답 시간보다 먼저, 업무 전문가가 질문 예시를 몇 가지 꼴로 적을 수 있는지를 확인해야 합니다.

7.2 영업기회 등록 스킬 — SKILL.md 281줄에서 52줄로, 턴당 도구 호출 1회로 줄인 진행기

영업기회 등록은 고객 문의 메일을 받아 CRM, 공유 드라이브, 영업 문서 페이지 세 곳에 같은 이름의 영업기회를 만드는 쓰기 업무입니다. 쓰기 업무는 한 번 잘못 실행되면 세 시스템을 모두 되돌려야 하므로, 조회 업무보다 결과의 일관성이 훨씬 중요합니다. 이 업무는 처음에 281줄짜리 SKILL.md를 모델이 읽고 단계를 엮는 방식이었고, 턴마다 도구를 11~15회 불렀습니다. 이 문제를 풀려고 판단 전체를 진행기라는 결정적 코드로 옮기고 모델에게는 명령 한 줄만 남겼습니다. LLM과 도구를 미리 정한 코드 경로로 엮는 워크플로 방식을 쓰기 업무에 적용한 것입니다[S1]. 여기서 진행기는 업무의 단계와 현재 상태를 쥐고 다음에 할 일을 코드로 정하는 프로그램을 말합니다.

7.2.1 되묻기부터 재조회까지 이어지는 진행기 처리 순서와 사람 확인 1회

업무 전문가가 정한 영업기회 등록 규칙

업무 전문가가 적은 규칙은 다섯 줄 남짓이었습니다. 요청 첫 줄에 「영업기회」라고 쓰고 아래에 메일 원문을 붙이면 등록 요청으로 보고, 영업기회 이름은 세 시스템 모두 `YYYYMMDD[기관명]사업명` 꼴로 같게 만듭니다. 수요기관과 사업명은 메일 본문에 라벨이 붙은 값만 읽고, 메일 제목의 대괄호 말머리는 수요기관으로 쓰지 않으며, 모르는 값은 짐작하지 않고 묻습니다. 세 곳에 무엇이 만들어질지 미리 보여 준 뒤 사람이 한 번 확인해야 씁니다. 이 문장들이 그대로 진행기의 단계와 정지 조건이 되었습니다.

결정적 진행기가 쥐는 여덟 단계 처리 순서

진행기는 요청 한 건을 다음 순서로 처리합니다. 각 단계의 판단은 코드가 하고, 모델이 개입하는 단계는 사람의 말을 받아 넘기는 일뿐입니다.

1. 첫 줄 판별 — 요청 첫 줄의 업무 이름으로 진행기를 고른다
2. 값 추출 — 메일 원문에서 라벨과 정규식으로 수요기관·사업명·문의자를 읽는다
3. 되묻기 — 읽지 못한 값이 있으면 정해진 질문을 출력하고 멈춘다
4. 드라이런 — 세 시스템에 만들 내용을 쓰기 없이 미리 보여 준다
5. 판단 모델 검증 — 초안이 요청 원문과 맞는지 판정해 어긋나면 경고를 붙인다
6. 사람 확인 1회 — 담당자가 「네」라고 답해야 다음으로 넘어간다

7. 멱등 쓰기 — 이미 있는 항목은 다시 만들지 않고 이어서 쓴다
8. 재조회 검증 — 쓴 뒤 세 시스템을 다시 읽어 결과가 맞는지 확인한다

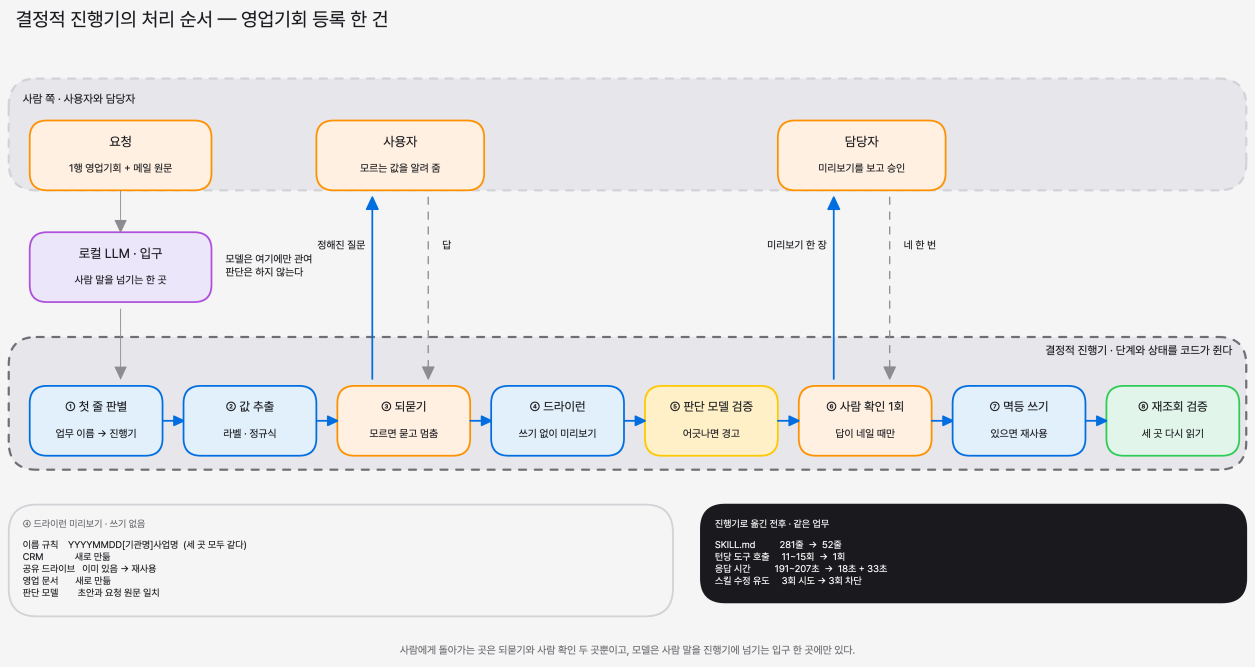


그림 7-1. 결정적 진행기의 처리 순서 — 첫 줄 판별부터 재조회 검증까지

그림에서 사람에게 돌아가는 단계는 되묻기와 사람 확인 두 곳뿐입니다. MCP 명세도 호스트가 도구를 부르기 전에 사용자의 명시적 동의를 받아야 한다는 원칙을 두는데[S3], 진행기는 이 동의를 모델의 해석에 맡기지 않고 코드가 받은 「네」 한 번으로 정합니다. 따라서 담당자는 매 단계를 지켜볼 필요 없이, 드라이런 미리보기 한 장을 보고 한 번만 판단하면 됩니다.

모델에게서 거둬들인 판단 — 지어낸 답과 직접 쓰기

진행기를 만들기 전 구성에서는 모델이 판단을 쥐면서 생긴 오류가 실제로 관찰되었습니다. 새 요청이 진행 중이던 이전 건의 답으로 처리되어 같은 영업기회가 다시 등록되었고, 모델이 사용자의 답을 기다리지 않고 답을 지어 넣어 수요기관 칸에 협력사 이름이 들어갔습니다. 모델이 CRM 쓰기 도구를 직접 부른 일도 있었고, 대화가 끝난 뒤에는 스킬 문서까지 고쳤습니다. 그래서 MSAP.ai AAS 구성에서는 사용자의 답을 훅만 기록하게 하고 서명이 없는 답은 진행기가 받지 않게 했으며, 진행 중인 턴의 도구 호출과 CRM 쓰기 도구는 도구 호출 전 훅이 막습니다. 스킬 파일은 지문을 대조해 바뀌면 원본으로 되돌리는데, Hermes Agent가 에이전트 스스로 스킬을 만들고 고치는 기능을 기본으로 두기 때문에 이 장치가 필요합니다[S8]. 이렇게 고친 뒤 같은 업무는 턴마다 도구 호출 0회, 4~29초로 처리되었고, 대화 중 스킬 문서를 고치도록 유도한 시도 3회는 모두 차단되었습니다.

7.2.2 프론티어 모델이 같은 명령을 실행한 결과와 일치한 결과 JSON

스킬 문서를 엮던 구성과 진행기 구성의 전후 비교

같은 문의 메일로 영업기회 등록을 다시 실행해 세 구성을 비교했습니다. 가운데 열은 활성 파라미터 3B 급 오픈소스 MoE 모델이 진행기를 실행한 결과이고, 오른쪽 열은 프론티어 모델이 같은 진행기 명령을 직접 실행한 결과로, 두 열을 견주면 실행 모델이 달라져도 결과가 같은지 알 수 있습니다.

항목	스킬 문서를 읽고 엮던 구성	진행기 구성(작은 모델)	같은 명령을 프론티어 모델이 실행
SKILL.md 길이	281줄	52줄	52줄(같은 스킬)
턴당 도구 호출	11~15회	1회	1회
응답 시간	191~207초	18초 + 33초(두 턴)	11초 + 21초(두 턴)
입력 토큰 소비	36만~48.5만 토큰	약 8.3만 토큰	해당 없음
결과	두 번 중 한 번은 인증 경로를 잘못 골라 실패	기존 항목 재사용, 결과 JSON 일치	기준

결과 JSON이 프론티어 모델 실행과 같았던 이유

가운데 열과 오른쪽 열의 결과 JSON과 추출값 JSON은 한 글자도 다르지 않았습니다. 값 추출, 후보 고르기, 순서, 확인 게이트, 멱등 쓰기를 모두 코드가 하므로, 어느 모델이 명령을 실행하든 같은 입력에서 같은 결과가 나오기 때문입니다. SKILL.md가 52줄이 된 것은 명령 한 줄과 금지 사항만 남기고 이전 절차 설명은 필요할 때만 여는 참고 파일로 옮긴 결과입니다. 스킬의 이름과 설명만 먼저 올리고 본문과 추가 파일은 필요할 때 읽는 점진적 공개 방식이 이 구성을 뒷받침합니다[S2]. vLLM의 OpenAI 호환 서버는 병렬 도구 호출을 끄면 요청당 도구 호출을 0~1회로 묶을 수 있어, 턴당 1회라는 진행기의 설계와 잘 맞습니다[S10].

신규 등록에서 재조회 검증이 맡는 역할

기존 항목을 재사용하는 경우와 별도로, 신규 영업기회 한 건을 만드는 경우에는 여섯 턴에 도구 호출 7회, 턴마다 5~40초가 걸려 생성을 마쳤습니다. 신규 등록은 기존 데이터에 없던 입력 꼴을 처음 만나는 경우가 많습니다. 예를 들어 사업명이 괄호로 시작하면 문서 페이지의 기록 표시가 링크 문법으로 해석되어 글자가 사라질 수 있는데, 진행기의 재조회 검증은 이런 경우 기록이 맞지 않는다며 멈춰 잘못된 결과가 그대로 남지 않게 합니다. 즉 재조회 단계는 쓰기 업무에서 결과의 어긋남을 실제로 잡는 장치이므로, 쓰기 업무의 파일럿에서는 「쓴 뒤 다시 읽어 맞는지 본다」를 업무 정의서에 처음부터 넣어야 합니다.

「결정적 코드로」 지시 한 줄만으로는 부족했던 대조 결과

같은 요구사항으로 작은 모델에게 직접 설계를 맡기면서 「결정적 코드로 작성하라」는 지시만 더한 대조에서는 SKILL.md가 142줄에서 62줄로, 실행 중 도구 호출이 15회에서 7회로 줄었습니다. 그러나 사용자 확인과 모호한 값 판단은 여전히 모델에게 남아 있었습니다(5장). 영업기회 등록 스킬은 프론티어 모델이 만들었고, 설계 검토 항목은 「판단 목록을 코드로 옮겼는가, 확인 게이트를 코드가 쥐는가」였습니다.

7.3 견적 초안 검증 — 첫 초안 단계부터 실행되는 판단 모델의 요청 충족 판정

견적 작성은 「견적」으로 시작하는 요청 문장에서 고객사, 수신처, 사업명, 금액, 기간을 읽어 CRM 견적 초안을 만들고, 사람이 확인하면 견적서와 PDF를 만드는 업무입니다. 견적은 값 하나가 틀려도 고객에게 잘못된 문서가 나가므로, 초안이 요청을 그대로 반영했는지 확인하는 단계가 꼭 필요합니다. MSAP.ai AAS 구성은 이 확인에 판단 모델, 즉 결과가 요청을 충족했는지 판정하는 별도 모델(6.3)을 씁니다. 판단 모델은 결과 글만 읽으므로, 무엇을 코드가 먼저 확인하고 무엇을 판단 모델에 맡길지 순서를 정하는 일이 이 사례의 핵심입니다.

7.3.1 실제 생성 여부를 코드가 먼저 확인한 뒤 판단 모델이 판정하는 순서

판단 모델이 볼 수 없는 것을 코드가 먼저 확인하는 이유

판단 모델은 요청 원문과 결과 글을 받아 「요청을 빠짐없이 정확히 충족했는가」에 점수로 답합니다. 결과 글에 「견적을 만들었습니다」라고 적혀 있어도 실제로 CRM에 견적이 생겼는지, PDF가 저장됐는지는 판단 모델이 알 수 없습니다. 그래서 견적번호 재조회, PDF 파일 존재, 링크 열림 같은 사실 확인은 코드가 먼저 하고, 판단 모델은 그 확인을 통과한 결과의 내용이 요청과 맞는지만 판정합니다. 정가와 할인율, VAT 표기, 영구 라이선스와 구독의 구분 같은 금액 규칙도 판단 모델은 알지 못하므로 기존 규칙 점검 코드가 맡습니다. 정리하면 값의 뜻은 판단 모델이, 사실과 규칙은 코드가 확인하는 분담이며, 같은 입력에 늘 같은 결과가 나와야 하는 확인을 코드에 두면 스크립트가 주는 결정적 신뢰성을 검증 단계에서도 얻을 수 있습니다[S2].

견적 업무의 검증 순서표

견적 한 건이 지나는 검증 순서는 다음 표와 같습니다. 판단 모델의 판정은 두 번, 첫 초안과 최종 확인 직전에 실행됩니다.

순서	담당	확인하는 것	어긋나면
1	코드	요청 원문에서 읽은 값, 거래처 정확 일치, 금액 규칙 점검	경고 또는 차단, 사람에게 질문
2	판단 모델	첫 초안이 요청 원문·사용자 답과 맞는가	초안에 경고 한 줄을 붙임
3	사람	초안 검토와 수정 지시	값을 고쳐 다시 1번부터
4	판단 모델	최종 확인 직전 초안이 요청과 맞는가	확인 메시지에 경고 한 줄을 붙임
5	사람	생성 승인 1회	승인 전에는 쓰지 않음
6	코드	견적번호 재조회, PDF 저장, 링크 기록	재시도 1회 또는 사람에게 넘김

검증을 마지막 확인에만 둘 때의 한계

판단 모델의 판정을 4번, 즉 최종 확인 직전에만 두면 그보다 앞 단계에서 생긴 어긋남이 먼저 업무에 영향을 줄 수 있습니다. 예를 들어 작은 모델이 수신처 값으로 요청 첫 줄의 「견적」이라는 낱말을 넘기고, 거래처 찾기 코드가 그 낱말을 이름이 일부만 겹치는 거래처에 붙이면 엉뚱한 연락처가 담당자 후보로 나옵니다. 최종 확인 단계의 판단 모델은 수신처 오류를 짚을 수 있지만, 그 전에 엉뚱한 거래처 밑에 연락처가 만들어질 수 있습니다. 따라서 검증은 사람에게 처음 보여 주는 초안 단계에서도 실행해야 하며, 위 표의 2번 단계가 이 역할을 합니다. 이와 함께 거래처는 이름이 정확히 일치할 때만 붙이고, 모델이 넘긴 값보다 요청 원문 해석을 앞세우는 규칙을 코드에 두면, 같은 요청이 도구 호출 20회를 넘기며 헤매던 구성이 턴마다 도구 호출 1회로 처리됩니다.

판단 서비스가 멈췄을 때의 진행 원칙

판단 모델 서비스가 꺼져 있거나 40초 안에 답하지 않으면 진행기는 판정을 건너뛰고 견적 작성을 계속합니다. 판단 모델은 경고를 붙이는 보조 장치이고, 생성 여부는 언제나 5번 단계에서 사람이 정하기 때문입니다. 1번과 6번의 코드 확인은 건너뛰지 않으며, 실패하면 견적을 만들지 않거나 사람에게 넘깁니다. 이 구분 덕분에 판단

모델을 붙여도 업무 전체가 그 서비스 하나의 가용성에 묶이지 않습니다. 생성 모델 쪽의 장애 우회와 재시도는 사내 게이트웨이인 LiteLLM이 배포 간 부하 분산과 함께 맡으므로[S9], 진행기는 판단 모델의 응답 한도만 따로 관리하면 됩니다.

7.3.2 시범 문항으로 본 판단 모델의 판정 정확도와 판정 시간

견적 초안 시범 문항 12건의 구성

판단 모델의 판정 정확도는 견적 초안 시범 문항 12건으로 확인했습니다. 12건 가운데 5건은 요청과 맞는 초안이고, 7건은 고객사, 수신처, 금액, 기간, VAT, 담당자, 품목 가운데 한 곳씩만 어긋나게 만든 초안입니다. 판단 모델에는 요청 원문, 사용자 답, 초안을 주고 충족 점수 하나를 받았으며, 코드가 그 점수로 통과와 경고를 나눕니다.

항목	결과
맞음·틀림 판정	12건 중 12건 정확
어긋난 곳 짚기	어긋난 초안 7건 중 7건 정확
충족 점수 분포	어긋난 초안 0.14 이하, 맞는 초안 0.83 이상
판정 시간(중앙값)	약 3초(첫 기동 때 약 9초)
같은 문항, 4B 급 소형 판단 모델	12건 중 7건, 어긋난 곳 7건 중 1건, 약 0.9초

판단 모델 정확도를 좌우하는 판정 지시문의 조건

판정 지시문에는 세 가지 조건이 필요합니다. 첫째, 사용자가 「담당자 2」처럼 번호로 답하면 판단 모델은 후보 목록을 보지 못하므로, 코드가 번호를 후보 이름으로 풀어서 넘깁니다. 둘째, 요청에 없는 값을 문제 삼아 맞는 초안을 틀렸다고 판정하지 않도록 「말하지 않은 값은 따지지 않는다」를 넣습니다. 셋째, 이 조건만 두면 금액이나 품목 어긋남을 놓칠 수 있으므로 「사업명, 품목, 금액은 반드시 본다」를 함께 넣습니다. 즉 판단 모델의 정확도는 모델만으로 정해지지 않고, 코드가 판단에 필요한 사실을 얼마나 풀어서 넘기는지에 크게 좌우됩니다. Anthropic이 권하는 실수 방지 설계, 즉 인자를 바꿔 실수 자체가 어렵게 만드는 방식도 같은 방향입니다[S1].

시범 문항 결과를 읽을 때 지켜야 할 범위

판정 정확도는 문항 구성과 지시문 조건에 따라 달라지므로, 시범 문항 12건의 결과가 실제 업무에서 그대로 나온다고 보아서는 안 됩니다. 실제 요청 문장으로 시험했을 때는 첫 초안 판정이 충족 0.89를 약 3초에 내 통과했고, 영구 라이선스 견적을 구독 금액으로 늘려 붙인 초안은 판단 모델이 아니라 코드의 규칙 점검이 잡았습니다. 판단 모델이 잘하는 일과 못 하는 일이 이렇게 실제 요청에서 나뉘어 드러나므로, 판정 정확도는 운영 중 쌓이는 실제 요청과 결과 쌍으로 다시 재야 합니다(8장). 3초 안팎의 판정을 초안마다 붙일 수 있다는 점은, 검증을 마지막에만 두지 않고 초안 단계부터 실행해도 응답 시간이 크게 늘지 않는다는 근거가 됩니다.

7.4 서버 업그레이드와 보안 공지를 반영하는 운영 에이전트 설계안 — 영업 밖 IT 운영 업무로의 확장

서버 업그레이드와 보안 공지 반영 운영 에이전트는 아직 만들어 운영해 본 적이 없는 설계안이며, 아래에는 실측 수치가 없습니다. 이 설계안을 싣는 이유는 앞의 세 사례가 모두 영업 업무여서, IT 운영이나 보안 부서가 같

은 방법을 쓸 수 있는지 판단할 재료가 필요하기 때문입니다. 한국인터넷진흥원(KISA) 보안 공지를 읽고 우리 서버에 해당하는 항목을 찾아 조치하는 업무는 반복 주기가 분명하고 판정 기준이 문장으로 정리되므로, 앞의 사례와 모양이 같습니다. 성과를 말할 수 있는 단계가 아니므로 업무 정의서와 운영 흐름만 제시합니다.

7.4.1 보안 공지 수집과 서버 업그레이드 판단을 나눈 업무 정의서 작성 예

업무 전문가가 말로 적는 운영 업무 문장

운영 담당자가 프론티어 모델에게 줄 업무 문장은 예를 들어 「KISA 보안 공지를 주기적으로 읽고, 우리 서버 목록에 해당되는 것이 있으면 조치 계획을 만들어 내 확인을 받은 뒤 반영해 줘. 서버 목록은 이 파일이야」와 같습니다. 이 한 문장에는 시작 신호(주기), 대상(서버 목록), 판정 기준(해당 여부), 사람 확인(내 확인), 실행(반영)이 모두 들어 있습니다. 여기에 지시문 해석, 결정적 코드, 질문 예시라는 스킬 작성 지시 세 가지를 붙이면 업무 정의서가 됩니다(5장). 질문 예시로는 「서버 1에 이번 달 OS 취약점 조치됐어?」, 「이번 공지 중에 우리 서버에 해당하는 것 있어?」, 「지난주 반영한 패치 목록 보여 줘」처럼 운영 담당자가 실제로 묻는 꼴을 여러 가지 적습니다. 여러 예시 입력으로 시험해 모델의 실수를 찾아 고치는 방식은 Anthropic이 도구 설계에 권하는 절차와 같기 때문에, 질문 예시는 많을수록 회귀 시험이 촘촘해집니다[S1].

코드·모델·사람으로 나눈 업무 정의서 작성 예(설계안)

설계안은 판단을 코드, 요약을 모델, 적용 결정을 사람에게 나눕니다. 아래 표는 업무 정의서에 적을 단계와 담당을 보여 줍니다.

단계	담당	업무 정의서에 적을 내용(설계안)
공지 수집	코드	공지 목록을 읽고 이미 처리한 공지는 건너뛴다
대상 정보 추출	코드	공지에서 CVE 번호, 제품명, 영향받는 버전을 읽는다. 못 읽으면 사람에게 넘긴다
서버 대조	코드	서버 목록의 제품·버전과 비교해 해당 서버를 고른다
조치 계획 문안	모델	해당 공지와 서버를 담당자가 읽기 쉬운 문장으로 요약한다
적용 결정	사람	서버별로 적용, 보류, 예외 중 하나를 고른다
적용과 재조회	코드	승인된 항목만 기존 변경 절차로 적용하고 버전을 다시 읽어 확인한다

폐쇄망 운영 환경에서 공지 수집 단계를 바꾸는 설계

이 설계안은 폐쇄망 운영 환경을 전제로 하므로 공지 수집 단계가 영업 사례와 다릅니다. 운영 환경은 인터넷에 닿지 않으므로, 공지 수집 코드는 망 연계로 반입한 공지 파일을 읽도록 만들고 인터넷에서 직접 받는 코드는 개발 환경의 시험용으로만 둡니다. CVE 번호와 버전 비교처럼 같은 입력에 같은 답이 나와야 하는 판단은 전부 코드에 두어야, 업그레이드가 될 때와 안 될 때가 실행마다 달라지는 일을 막을 수 있습니다. 모델이 맡는 일은 요

약 문장뿐이고, 범용 대화가 필요한 일만 큰 모델에 맡기는 이종 구성이 이 분담과 같은 방향입니다[S4]. 따라서 운영 부서가 먼저 확인할 것은 모델 성능보다, 서버 목록과 버전 정보를 코드가 읽을 수 있는 형태로 갖고 있는가입니다.

7.4.2 예약 작업과 사람 승인 지점을 이은 운영 흐름 설계

MSAP.ai AAS 예약 작업으로 실행하는 정기 점검 흐름(설계안)

설계안의 운영 흐름은 MSAP.ai AAS의 예약 작업(크론)이 정해진 주기에 진행기를 실행하는 것으로 시작합니다. MSAP.ai AAS는 MCP 도구로 예약 작업 만들기와 실행, 채팅 시작과 메시지 보내기를 제공하므로, 업무 전문가는 프론티어 모델에게 「매주 월요일 아침에 이 점검을 실행해」라고 말해 예약 작업까지 만들 수 있습니다. MCP는 LLM 애플리케이션과 외부 도구를 잇는 개방형 프로토콜이며, 호스트가 서버의 도구를 부르는 구조입니다[S3]. 흐름은 다음 순서로 설계합니다.

1. 예약 작업이 진행기를 실행한다
2. 코드가 공지를 수집하고 서버 목록과 대조한다
3. 해당 항목이 없으면 점검 기록만 남기고 끝낸다
4. 해당 항목이 있으면 코드가 적용 대상·명령·재기동 필요 여부를 드라이런으로 만들고, 모델이 조치 계획 문안을 쓴다
5. 운영 채널로 담당자에게 승인 요청을 보낸다
6. 담당자가 승인한 항목만 코드가 적용한다
7. 코드가 버전을 다시 읽어 적용 결과를 확인하고 기록한다

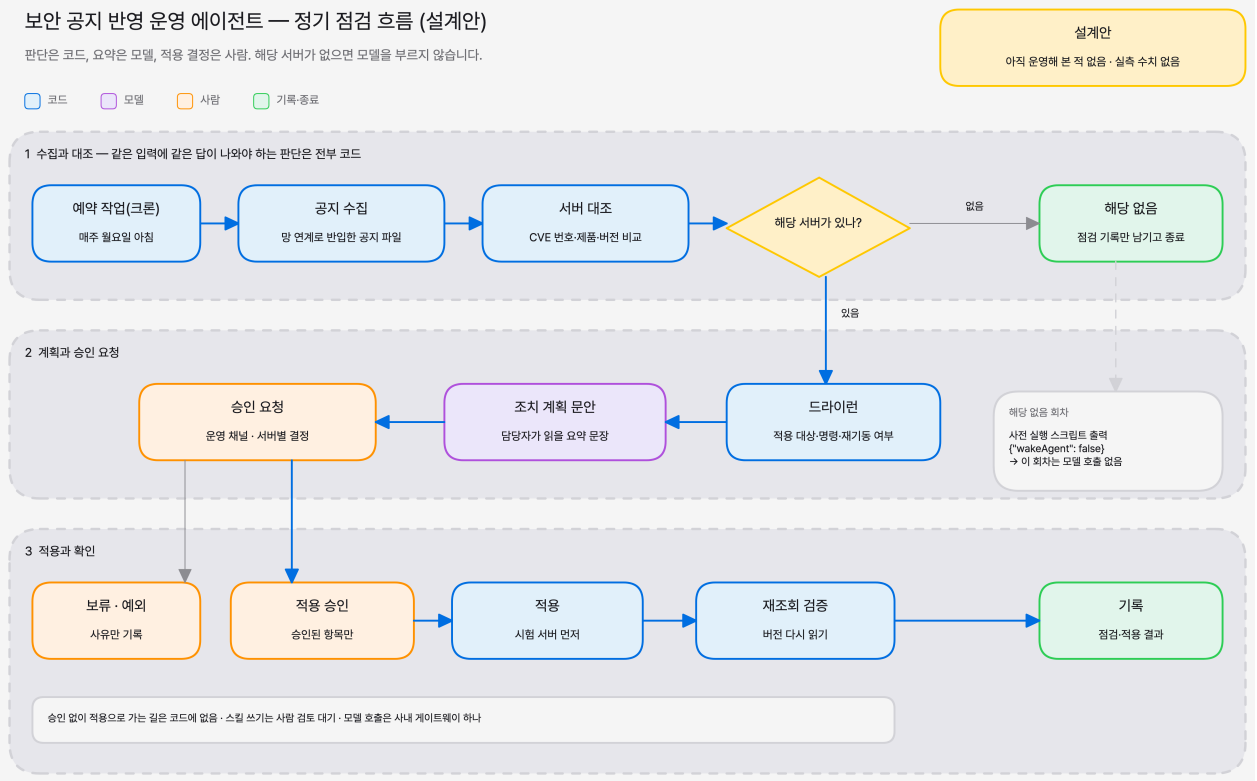


그림 7-2. 보안 공지 반영 운영 에이전트의 정기 점검 흐름(설계안)

사람 승인과 스킬 변경 승인을 코드로 고정하는 이유

이 흐름에서 사람의 승인은 적용 직전 한 번입니다. MCP 명세는 사용자가 모든 작업에 명시적으로 동의하고 통제해야 하며 도구는 임의 코드 실행이므로 신중히 다루라고 하므로[S3], 승인 없이 적용으로 넘어가는 길은 코드에 두지 않습니다. 운영 중 에이전트가 스킬을 스스로 고치는 일을 막기 위해, Hermes Agent의 스킬 쓰기를 사람 검토 대기로 전환하는 설정과 MSAP.ai AAS의 도구 호출 전 차단을 함께 씁니다[S8]. 모델 호출은 사내 게이트웨이 하나를 거치게 하고, LiteLLM의 팀별 가상 키로 운영 부서의 모델과 예산을 영업 부서와 따로 관리합니다[S9]. 이렇게 하면 영업 업무에서 검증한 구성 요소를 그대로 쓰면서, 운영 부서에서 새로 만들 것은 공지 수집과 서버 대조 코드뿐입니다.

해당 항목이 없는 회차에 모델을 부르지 않는 예약 작업 구성

Hermes Agent의 예약 작업에는 회차마다 모델 호출 여부를 다시 정하는 장치가 있습니다. 예약 작업에 사전 실행 스크립트를 붙여 두고 그 스크립트가 에이전트를 깨우지 않겠다는 값(`{"wakeAgent": false}`)을 돌려주면 그 회차는 모델을 부르지 않고 끝납니다. 이 설계안에 대입하면, 수집한 공지가 서버 목록과 겹치지 않는 회차는 코드가 점검 기록만 남기고 끝내고, 해당 항목이 있는 회차에만 모델이 조치 계획 문안을 씁니다. 실행 주기는 그대로 두면서 실제 추론 호출 수만 줄어들므로, 보급형 GPU에 올린 운영 모델의 부하도 해당 항목이 있는 회차에만 생깁니다.

판단이 전혀 필요 없는 정기 업무라면 예약 작업 전체를 모델 밖에 둘 수도 있습니다. 스크립트를 일정에 걸고 그 출력을 그대로 담당자에게 배달하는 방식은 추론 호출이 한 건도 생기지 않으므로, 지표 수집이나 점검 기록처럼 정해진 형식의 결과만 전하는 업무에서 비용과 실패 요인을 함께 줄입니다. 그래서 예약 실행 건수를 그대로 모델 호출 건수로 환산해 GPU 규모를 정하면 실제보다 큰 값이 나오며, 업무 목록에서 모델 없이 끝나는 작업을 먼저 가려내야 운영 모델 용량을 바르게 잡을 수 있습니다.

설계안을 파일럿으로 옮길 때 먼저 정할 합격선

이 설계안은 수치가 없으므로, 파일럿으로 옮기기 전에 무엇을 재면 합격인지부터 정해야 합니다. 앞의 사례에 비추면 MSAP.ai AAS 채팅에 실제 질문 예시를 넣었을 때 도구 호출 수, 응답 시간, 정답 여부를 재고, 공지와 서버 대조 결과는 운영 담당자가 미리 손으로 맞춘 정답 묶음과 견줍니다(8장). 적용 단계는 시험 서버에서만 먼저 실행하고, 재조회 검증이 버전 변화를 정확히 읽는지 확인한 뒤에 운영 서버로 넓힙니다. 따라서 IT 운영 부서는 이 설계안을 그대로 도입 계획으로 삼기보다, 질문 예시와 정답 묶음을 먼저 만드는 데서 시작하는 편이 안전합니다.

8장: 만든 스킬이 쓸 만한지 재는 기준과 운영

MSAP.ai AAS에서 업무 전문가가 만든 스킬은 프론티어 모델이 작성한 규칙 파일, 결정적 스크립트, 시험, 짧은 SKILL.md로 이루어지고, 운영 환경에서는 활성 파라미터 3B 급 오픈소스 MoE(Mixture of Experts, 전문가 혼합) 모델이 그 스킬을 실행합니다. 스킬을 만드는 모델과 실행하는 모델이 서로 다르기 때문에, 프론티어 모델이 개발 환경에서 정답을 냈다는 사실만으로는 운영 모델도 같은 결과를 낸다고 판단할 수 없습니다. 그래서 운영 반입 여부는 실제 운영 모델에 실제 요청 문장을 넣어 얻은 결과로 판정해야 합니다. Anthropic 역시 Building effective agents에서 에이전트가 쓰는 도구를 설계할 때 여러 예시 입력을 직접 넣어 모델이 어떤 실수를 하는지 확인하고 고치는 과정을 반복하라고 권합니다[S1].

스킬은 한 번 만들고 끝나는 산출물로 남지 않습니다. 업무 규칙이 바뀌면 스크립트를 고쳐야 하고, 더 빠르거나 운영 비용이 낮은 모델이 나오면 운영 모델을 교체하고, 운영 중인 에이전트가 스스로 스킬 파일을 고치는 일도 생깁니다. Hermes Agent는 에이전트가 `skill_manage` 도구로 스킬을 만들고 고치고 지우는 기능을 「절차 기억」이라는 이름으로 제공합니다(Hermes Agent 스킬 문서)[S8]. 따라서 변경이 생길 때마다 무엇을 근거로 승인할지를 미리 정해 두지 않으면 스킬 품질은 담당자의 감각과 기억에 기대고, 담당자가 바뀌는 순간 그 근거도 함께 사라집니다.

이 장에서는 업무 전문가가 코드를 읽지 않고도 합격 여부를 판정할 수 있는 세 지표와 기록 양식을 먼저 정리했습니다. 운영 중 오류를 재시도하거나 사람에게 이관하는 결과 검증 순환, 질문 예시 회귀 시험으로 변경을 승인하는 절차, 업무 정의서에서 시작해 개정으로 돌아오는 스킬 생애 주기, 담당자별 책임 분담도 차례로 다룹니다. 의사결정권자가 스킬의 운영 반입과 변경을 승인할 때 조건으로 삼을 것은 합격선 충족, 회귀 시험 통과라는 두 가지입니다.

8.1 실제 요청 기준의 합격선 — 도구 호출 3회 이하, 수 분 이내 응답, 정답

MSAP.ai AAS가 권하는 합격선은 AAS 채팅에 실제 요청을 넣었을 때 도구 호출 3회 이하, 수 분 이내 응답, 정답이라는 세 조건을 모두 만족하는 것입니다. 세 값은 모두 채팅 화면에서 바로 읽을 수 있으므로 업무 전문가가 스크립트를 열어 보지 않고도 합격 여부를 판정할 수 있습니다. 기준값은 예시이며, 조직은 업무의 긴급도와 보유한 GPU 규모에 맞춰 시간 상한이나 호출 수 상한을 조정할 수 있습니다. 다만 세 조건 가운데 하나를 빼고 운영하면 그 조건이 걸러 내던 설계 결함이 그대로 운영 환경에 반입되므로, 기준값을 조정하더라도 세 지표는 늘 함께 기록합니다.

세 지표는 각각 스킬 설계의 서로 다른 상태를 보여 줍니다. 도구 호출 수는 판단이 코드로 옮겨졌는지를 나타냅니다. 판단을 코드로 옮긴 스킬에서는 모델이 명령 한 줄을 실행하고 결과를 그대로 전달하므로 호출 수가 1회 안팎에 머물고, 판단이 모델에 남아 있으면 모델이 파일을 읽고 조회를 되풀이하면서 호출 수가 수십 회로 늘어납니다. 응답 시간은 호출 수와 함께 움직이는데, LLM이 토큰을 하나씩 만드는 생성 단계는 계산량보다 메모리에서 가중치를 읽어 오는 속도가 지연을 정하는 작업이기 때문입니다[S5]. 도구 결과가 맥락에 쌓일수록 매 호출에서 처리할 입력이 커지므로, 호출 수가 늘면 응답 시간도 함께 늘어납니다.

지표	기준값(MSAP.ai AAS 권장 예시)	채팅 화면에서 읽는 값	기준을 넘을 때 먼저 의심할 설계 문제
도구 호출 수	요청 한 건당 3회 이하	응답과 함께 표시되는 도구 호출 목록의 개수	판단 목록이 코드로 옮겨지지 않고 모델에 남아 있음
응답 시간	수 분 이내	요청을 보낸 시각부터 최종 응답이 끝난 시각까지	모델이 파일 내용을 읽으며 절차를 엮고 있음
정답 여부	기대 결과 일치	응답 내용과 업무 시스템의 실제 반영 결과	지시문 해석 규칙이나 질문 예시가 부족함

정답 여부는 업무 전문가만 판정할 수 있는 지표이며, 앞의 두 지표가 기준을 만족해도 이 값이 틀리면 불합격입니다. NVIDIA 연구진은 에이전트 안의 언어 모델이 적은 수의 전문 작업을 거의 변화 없이 반복하므로 작은 모델로 충분하다고 주장합니다[S4]. 하지만 이 주장은 입장 논문의 논지이고 실험으로 증명된 결론은 아니므로, 각 조직은 자기 업무의 실제 요청으로 합격선을 재서 작은 모델이 충분한지를 직접 확인해야 합니다.

8.1.1 업무 전문가가 코드 없이 판정하는 세 지표와 측정 절차

측정 요청 문장의 출처인 질문 예시

측정에 쓰는 요청 문장은 업무 전문가가 업무 정의서에 적어 둔 질문 예시에서 가져옵니다. 질문 예시는 스킬 작성 지시 세 가지 가운데 세 번째 항목으로 프론티어 모델에게 전달한 문장 묶음이며, 같은 업무를 여러 꼴로 묻는 문장과 업무 시나리오를 함께 담고 있습니다. 예를 들어 서버 패치 업무라면 「서버 1에 OS 취약점 조치됐어?」와 「CVE 몇 번 적용됐어?」처럼 표현은 다르지만 같은 조회로 이어져야 하는 질문을 함께 넣습니다. 측정 묶음에는 정상 요청뿐 아니라 필요한 값이 빠진 요청이나 두 업무가 섞인 요청처럼 스킬이 되물어야 하는 문장도 포함합니다. 이때 스킬이 정해진 질문을 출력하고 멈췄다면 그 응답이 정답이며, 값을 추측해 업무 시스템에 써 넣었다면 결과가 그럴듯해 보여도 오답입니다.

AAS 채팅 화면에서 세 지표를 읽는 측정 순서

측정은 운영 모델이 연결된 에이전트 프로파일에서 진행합니다. 업무 전문가는 AAS 콘솔에서 해당 프로파일을 고르고 새 대화를 시작한 뒤 질문 예시를 원문 그대로 입력합니다. 화면에는 응답과 함께 모델이 부른 도구 호출이 표시되므로 그 개수를 세고, 요청을 보낸 시각부터 최종 응답이 끝난 시각까지를 소요 시간으로 적습니다. 쓰기가 일어나는 업무라면 응답 문장만 보지 않고 업무 시스템 화면에서 레코드가 실제로 생겼는지도 함께 확인해야 하는데, 모델이 「등록했습니다」라고 답했어도 실제로는 쓰기가 실패한 경우가 있기 때문입니다. vLLM은 `parallel_tool_calls=false` 로 설정하면 요청 하나에서 도구 호출을 0~1회로 제한하므로[S10], 이렇게 운영하는 환경에서는 화면의 호출 수가 모델이 다음 동작을 다시 고른 횟수와 거의 같아집니다.

단계	하는 일	확인 위치	기록 값
1. 준비	질문 예시마다 기대 결과를 미리 적음	업무 정의서	기대 결과
2. 입력	운영 모델 프로파일의 새 대화에 요청 원문 입력	AAS 채팅 화면	요청 원문, 시작 시각

단계	하는 일	확인 위치	기록 값
3. 관찰	도구 호출 표시 개수와 응답 종료 시각 확인	AAS 채팅 화면	도구 호출 수, 소요 시간
4. 대조	응답·업무 시스템 반영 결과를 기대 결과에 대조	채팅 응답, 업무 시스템 화면	정답 여부
5. 반복	같은 요청을 두세 번 다시 넣어 결과가 같은지 확인	AAS 채팅 화면	반복 일치 여부

마지막 반복 단계는 운영 모델의 출력에 무작위성이 있다는 점 때문에 둡니다. 판단을 코드로 옮긴 스킬은 같은 입력에 같은 결과를 내야 하므로, 반복할 때마다 결과가 달라진다면 아직 모델에게 맡긴 판단이 남아 있다는 신호로 읽습니다.

질문 예시 묶음의 MCP 반복 실행

질문 예시가 수십 건으로 늘면 사람이 한 건씩 입력해 시간을 재기 어렵습니다. MSAP.ai AAS는 MCP 서버로 채팅 시작·보내기·기다리기 도구를 제공하므로, 업무 전문가는 개발 환경의 프론티어 모델에게 「이 예시 묶음을 운영 모델 프로파일에 차례로 보내고 도구 호출 수와 소요 시간, 응답을 표로 정리하라」고 말로 지시할 수 있습니다. MCP 명세는 호스트 애플리케이션이 서버의 도구를 부르기 전에 사용자의 명시적 동의를 받도록 정하고 있어[S3], 측정 실행도 업무 전문가가 승인한 범위 안에서만 진행됩니다. 이 방식에서 프론티어 모델은 요청을 대신 보내고 결과를 모을 뿐이며, 요청에 답하는 모델은 운영 모델 그대로입니다. 결과표에는 운영 모델의 실제 동작이 그대로 남고, 업무 전문가는 표를 보며 정답 열만 채웁니다.

8.1.2 사례별 전후 수치를 한 표로 모은 합격 판정 기록 양식

합격 판정 기록 양식에 남기는 항목

합격 판정 기록 양식은 요청 원문, 스킬 판 번호, 운영 모델 이름, 세 지표, 판정 결과, 판정자를 한 줄에 남기는 표입니다. 같은 양식을 계속 쓰면 개선 전후를 나란히 놓고 비교할 수 있고, 나중에 업무 규칙을 고치거나 운영 모델을 바꿀 때 이 기록을 비교 기준값으로 다시 씁니다. 입력 토큰 수는 합격선에 들어가지 않지만 GPU 사용량과 운영 비용을 가늠하는 값이므로 참고 열로 함께 남깁니다. 아래 표는 한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영했을 때 관찰된 동작(7장 사례)을 이 양식에 옮겨 적은 예입니다.

업무	측정 시점	도구 호출	응답 시간	입력 토큰(참고)	결과	판정
CRM 자연어 조회	개선 전	36회	394초	약 127만	오답	불합격(오답, 호출 수 초과)
CRM 자연어 조회	결정적 조회 코드 이관 후	0회	14초	기록 생략	정답	합격
영업기회 등록	개선 전	턴당 11~15회	191~207초	36만~48.5만	기록 생략	불합격(호출 수 초과)
영업기회 등록	진행기 적용 후	턴당 1회	18초, 33초(두 턴)	약 8.3만	프론티어 모델이 같은 명	합격

업무	측정 시점	도구 호출	응답 시간	입력 토큰(참고)	결과	판정
					령을 실행한 결과와 일치	
「결정적 코드로」 지시 한 줄만 준 대조 설계	지시 반영 후	15회에서 7회로 감소	기록 생략	기록 생략	확인 게이트가 모델에 남음	불합격(호출 수 초과)

> 📊 *데이터 — 표의 수치는 외부 벤치마크가 아니며, 한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영했을 때 관찰된 제품 동작입니다. 같은 수치를 다른 조직의 업무에 그대로 기대할 수는 없으므로 각 조직은 같은 양식으로 자기 업무를 다시 잽니다.*

표의 불합격 행이 가리키는 설계 결함

기록 양식의 가치는 합격 행보다 불합격 행에서 드러납니다. CRM 자연어 조회는 개선 전에 도구를 36회 부르고도 오답을 냈는데, 이는 조회 조건을 고르는 판단이 모델에 남아 있어 모델이 매번 다른 조회를 시도했다는 뜻입니다. 반면 판단을 결정적 조회 코드로 옮긴 뒤에는 도구 호출 없이 14초 만에 정답을 냈으므로, 같은 운영 모델에서도 설계에 따라 결과가 달라진다는 점이 수치로 남았습니다. 마지막 대조 설계는 도구 호출이 15회에서 7회로 줄어 개선된 것처럼 보이지만 3회 상한을 넘었고, 확인 게이트와 모호성 판단이 여전히 모델 몫으로 남아 있었습니다. 즉 합격선은 「좋아졌다」와 「운영에 넣어도 된다」를 구분해 주며, 업무 전문가는 이 차이를 코드 없이 표 한 장으로 판정할 수 있습니다.

기록 양식이 쌓여서 생기는 감사 추적

판정 기록이 스킬 판마다 쌓이면 어떤 변경이 언제 어떤 근거로 운영에 들어갔는지를 나중에 되짚을 수 있습니다. 영업기회 등록 스킬의 SKILL.md를 281줄에서 52줄로 줄인 개정도 승인 근거는 줄 수보다 도구 호출 수, 응답 시간, 결과 일치 기록으로 남깁니다. 판정 기록은 감사 추적과 변경 추적의 자료가 되고, 다음 절의 결과 검증 순환에서 실패 유형을 분류할 때 비교 자료로도 쓰입니다.

8.2 코드 실행과 판단 모델 판정, 재시도 1회 또는 사람 이관으로 이어지는 결과 검증 순환

합격선을 통과한 스킬도 운영 중에는 실패합니다. 업무 시스템이 응답하지 않거나, 요청 문장이 예시에 없던 꼴로 들어오거나, 작은 모델이 결과 문구를 고쳐 쓰는 경우가 생기기 때문입니다. MSAP.ai AAS는 이런 실패를 다루기 위해 코드가 먼저 실행하고, 판단 모델이 결과가 요청을 충족했는지와 실패 유형을 판정하고, 코드가 그 판정에 따라 재시도 1회 또는 사람 이관을 고르는 결과 검증 순환을 씁니다. Anthropic은 한 모델이 결과를 만들고 다른 모델이 평가와 피드백을 주는 평가자-개선자(evaluator-optimizer) 구성을 워크플로 유형의 하나로 소개하면서, 에이전트가 단계마다 도구 실행 결과 같은 실제 환경의 근거를 확인해야 하고 반복 횟수 상한 같은 중단 조건을 두어야 한다고 설명합니다[S1].

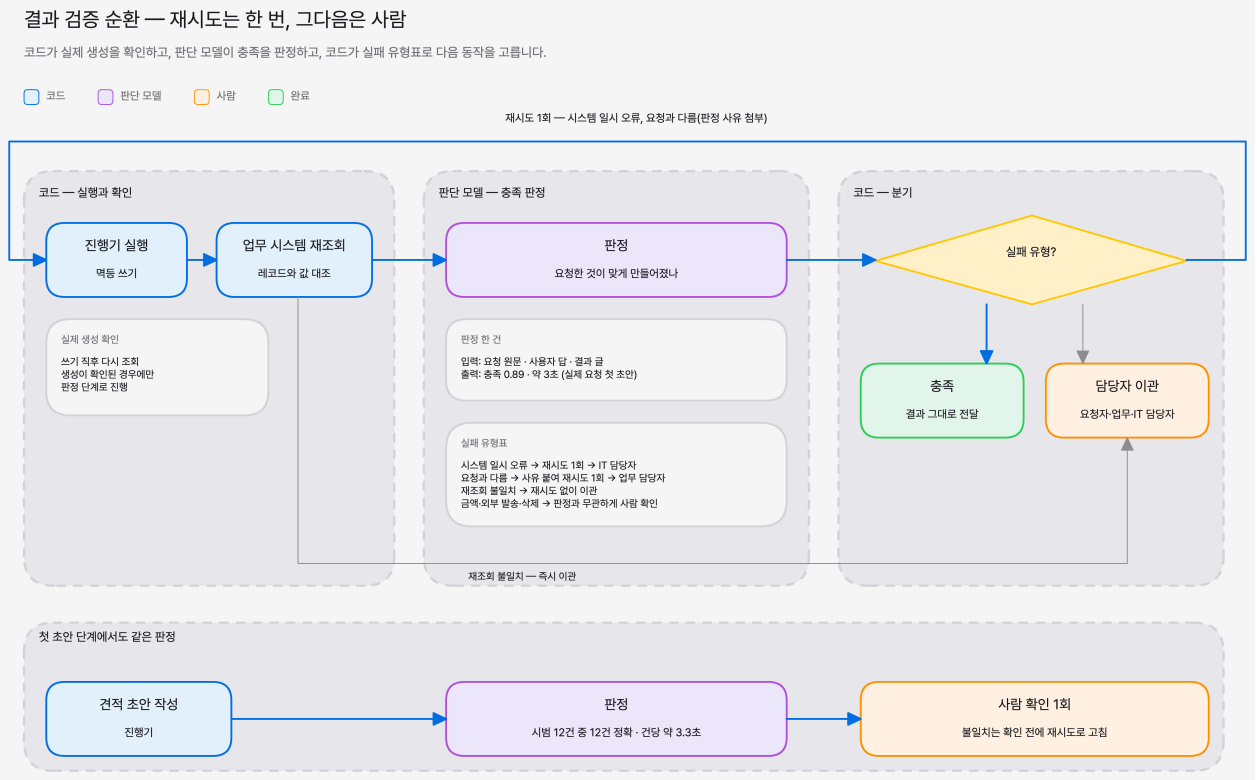


그림 8-1. 검증 순환도 — 코드 실행·실제 생성 확인, 판단 모델 판정, 재시도 1회 또는 사람 이관

결과 검증 순환에서 판정 결과별 이관 경로가 미리 정해져 있으면, 운영 중 오류가 누구에게 이관됐고 누가 처리했는지가 기록으로 남아 책임이 분명해집니다. Anthropic은 에이전트가 진행 중 막히거나 확인이 필요한 단계에서 사람의 피드백을 받도록 멈출 수 있어야 한다고 설명하는데[S1], 결과 검증 순환의 사람 이관은 그 멈춤을 코드가 정한 규칙으로 실행하는 장치입니다.

8.2.1 결과 글만 보는 판단 모델의 한계와 코드가 먼저 확인하는 실제 생성 여부

판단 모델이 볼 수 있는 정보의 범위

판단 모델은 요청 원문, 결과 글을 함께 받아 요청이 충족됐는지를 판정하는 모델입니다. 그런데 판단 모델이 받는 입력은 글뿐이므로, 업무 시스템에 레코드가 실제로 생겼는지, 금액이 실제로 그 값으로 저장됐는지는 볼 수 없습니다. 따라서 결과 글이 「영업기회를 등록했습니다」라고 적혀 있으면 판단 모델은 충족으로 판정할 가능성이 높고, 쓰기가 실패한 경우를 놓칩니다. 이 한계의 원인은 판단 모델의 성능보다 입력에 사실 확인 결과가 들어 있지 않은 구조에 있습니다.

판단 모델이 놓치는 경우와 앞에 두는 코드 확인

판단 모델이 놓치는 경우는 크게 네 가지이며, 각각 판정 전에 코드가 먼저 확인할 항목이 정해져 있습니다.

판단 모델의 한계	운영 중 실제로 생기는 일	판정 전에 코드가 먼저 하는 확인
실제 생성 여부를 볼 수 없음	쓰기가 실패했는데 결과 글은 성공으로 보임	쓰기 직후 업무 시스템을 재조회해 레코드와 값 대조
결과 글이 바뀌면 판정 대상도 바뀜	작은 모델이 진행기 출력 문구를 고쳐 써서 원래 결과가 사라짐	출력 직후 훅이 모델의 답을 진행기 원문으로 교체

판단 모델의 한계	운영 중 실제로 생기는 일	판정 전에 코드가 먼저 하는 확인
요청 해석이 판단 모델마다 다를 수 있음	모호한 요청을 판단 모델이 임의로 해석해 충족으로 판정	지시문 해석 규칙이 모르는 요청을 먼저 되물음
판정에 시간이 더 듦	모든 단계에 판정을 넣으면 응답 시간이 늘어남	판정이 정확도를 올리는 단계에만 판단 모델 배치

표의 첫 번째 항목인 실제 생성 여부 확인이 네 가지 가운데 가장 중요합니다. MSAP.ai AAS의 진행기는 쓰기를 멱등 쓰기(같은 요청을 여러 번 실행해도 한 번 실행한 것과 결과가 같은 쓰기)로 처리하고 쓰기 직후 업무 시스템을 다시 조회해 결과를 대조하며, 재조회로 실제 생성이 확인된 경우에만 판단 모델의 판정 단계로 진행합니다. Anthropic도 에이전트가 단계마다 도구 호출 결과나 코드 실행 결과 같은 실제 환경의 근거로 진행 상태를 확인해야 한다고 설명합니다[S1]. 즉 「만들어졌는가」는 코드가 사실로 확인하고, 판단 모델은 「요청한 것이 맞게 만들어졌는가」만 판정하도록 역할을 나눕니다.

첫 초안 단계부터 실행하는 판정과 판정 시간

판정은 마지막 확인 단계에만 두지 않고 첫 초안 단계부터 실행합니다. 견적 초안 검증용 시범 문항 12건에서 판단 모델은 12건 모두 충족 여부를 정확히 판정했고, 한 건 판정에 약 3.3초가 걸렸습니다. 초안 단계에서 요청 불일치를 잡으면 사람이 확인하기 전에 재시도로 고칠 수 있어, 사람 확인 1회가 실제로 1회로 끝나는 비율이 높아집니다. 반면 스킬 고르기 단계에 판단 모델을 더했을 때는 정확도가 86.4%에서 88.6%로 조금 올랐지만 시간이 0.7초에서 20.5초로 늘어 채택하지 않았습니다. 다만 이 결과는 시범 문항 12건에 대한 것이어서 판정 정확도를 일반화하기에는 표본이 작으므로, 각 조직은 자기 업무의 실패 사례를 문항에 더해 판정 정확도를 다시 재야 합니다.

8.2.2 실패 유형별 재시도와 사람 이관을 가르는 기준

재시도 1회 상한을 두는 이유

재시도는 1회로 제한합니다. 결정적 코드는 같은 입력에 같은 결과를 내므로, 코드가 낸 오류는 같은 입력으로 몇 번을 다시 실행해도 같은 오류로 끝납니다. 따라서 재시도가 의미 있는 경우는 업무 시스템의 일시적인 응답 지연이나 작은 모델의 전달 실수처럼 실행할 때마다 결과가 달라질 수 있는 실패뿐입니다. Anthropic도 에이전트에 반복 횟수 상한 같은 중단 조건을 두어 통제를 유지하라고 권합니다[S1]. 재시도 상한이 없으면 같은 실패가 도구 호출과 시간만 소비하며 반복되고, 그동안 사람은 실패 사실을 알지 못합니다.

실패 유형별 처리 기준표

판단 모델이 내는 실패 유형마다 처리 경로를 미리 정해 두면 코드는 판정 결과만 보고 다음 동작을 고를 수 있습니다.

실패 유형	판정 근거	처리	이관 대상
필요한 값 누락, 요청이 모호함	지시문 해석 단계에서 판별 불가	재시도 없이 정해진 질문을 출력하고 멈춤	요청자
업무 시스템 일시 오류(시간 초과, 연결 실패)	코드의 실행 결과	재시도 1회, 다시 실패하면 이관	IT 담당자

실패 유형	판정 근거	처리	이관 대상
결과가 요청 내용과 다름	판단 모델의 미충족 판정·사유	판정 사유를 붙여 재시도 1회, 다시 실패하면 이관	업무 담당자
쓰기 후 재조회 결과가 기대값과 다름	코드의 재조회 대조	재시도 없이 즉시 이관	업무 담당자와 IT 담당자
확인 게이트 대상(금액, 외부 발송, 삭제)	규칙 파일의 게이트 목록	판정과 무관하게 사람 확인 후 실행	업무 담당자

쓰기 후 재조회가 어긋난 경우는 재시도하지 않습니다. 이미 일부가 업무 시스템에 반영됐을 수 있으므로 다시 실행하면 중복 등록이나 값 덮어쓰기가 생길 수 있기 때문입니다. 확인 게이트 대상은 판단 모델이 충족이라고 판정해도 사람이 확인한 뒤에 실행하며, 이 규칙은 모델의 판단에 상관없이 진행기 코드가 규칙 파일대로 적용합니다. 이관 경로와 수신자가 실패 유형마다 정해져 있으면, 운영 중 오류가 발생했을 때 의사결정권자는 그 오류가 어느 담당자에게 이관됐고 몇 번 재시도했는지를 기록에서 바로 확인할 수 있습니다.

8.3 질문 예시 회귀 시험으로 관리하는 스킬 변경 — 모델 교체와 업무 규칙 변경 때의 재확인

회귀 시험은 스킬을 고친 뒤 이전에 통과하던 질문 예시를 그대로 다시 실행해 결과가 여전히 같은지를 확인하는 시험입니다. 스킬 작성 지시 세 번째 항목이 프론티어 모델에게 질문 예시 하나하나를 회귀 시험으로 고정하라고 요구하는 이유도 여기에 있습니다. Agent Skills 구조에서는 스킬 디렉터리에 스크립트를 넣으면 에이전트가 이를 도구처럼 실행하고, 코드만 줄 수 있는 결정적 신뢰성을 얻을 수 있습니다[S2]. 결정적 코드는 같은 입력에 같은 출력을 내므로 회귀 시험 결과가 달라졌다면 그 원인은 변경 내용에 있다고 좁혀 볼 수 있고, 변경 승인의 근거를 시험 결과에 둘 수 있습니다.

변경이 생기는 계기는 크게 셋입니다. 업무 규칙이 바뀌어 업무 정의서와 스크립트를 고치는 경우, 운영 모델을 다른 모델이나 다른 양자화 판으로 교체하는 경우, 그리고 에이전트 실행 환경을 새 버전으로 올리는 경우입니다. 세 경우 모두 같은 질문 예시 묶음을 같은 기록 양식으로 다시 실행하고 비교하면 되므로, 변경 종류마다 다른 승인 절차를 만들 필요가 없습니다.

8.3.1 회귀 시험 통과를 운영 반영 조건으로 두는 변경 관리

스킬 변경 승인 절차

스킬 변경은 업무 정의서 개정에서 시작해 회귀 시험 통과를 거쳐 스킬 업로드로 끝납니다. 이 순서를 지키면 개정 이유, 시험 결과, 반입 기록이 한 묶음으로 남습니다.

순서	하는 일	담당	산출물	다음 단계로 가는 조건
1	업무 정의서 개정과 변경 사유 기록	업무 전문가	개정된 업무 정의서	바뀐 규칙이 문장으로 적혀 있음
2	바뀐 규칙을 반영한 질문 예시 추가, 기존 예시 유지	업무 전문가	확장된 질문 예시 묶음	새 예시마다 기대 결과가 있음

순서	하는 일	담당	산출물	다음 단계로 가는 조건
3	프론티어 모델이 MCP로 스크립트·시험·SKILL.md 수정	업무 전문가(지시), 프론티어 모델(작성)	새 스킬 판	스크립트 수준 시험 전부 통과
4	운영 모델로 전체 예시 묶음 재실행	업무 전문가	합격 판정 기록	모든 예시가 합격선 충족
5	스킬 업로드로 운영 환경 반입 승인	IT 담당자	반입 기록	4단계 기록 첨부
6	반입 후 운영 환경에서 대표 예시 재실행	업무 전문가, IT 담당자	반입 확인 기록	개발 환경 결과와 일치

핵심 규칙은 4단계에서 예시 하나라도 불합격하면 반입하지 않는다는 것입니다. 새 규칙을 반영한 예시가 통과해도 기존 예시가 하나 깨졌다면, 그 변경은 다른 업무의 동작을 바꾼 것이므로 3단계로 돌아가 원인을 고칩니다. 예를 들어 한 업무 에이전트의 CRM 자연어 조회 스킬은 의도 규칙과 이름 색인을 바꿀 때마다 질문 예시 53건을 회귀 시험으로 다시 실행합니다. MCP는 호스트가 도구를 실행하기 전에 사용자 동의를 받도록 정하고 있으므로[S3], 3단계에서 프론티어 모델이 스킬을 수정하고 올리는 동작도 업무 전문가가 승인한 범위 안에서 기록과 함께 진행됩니다.

측정 결과로 변경을 거절한 사례

회귀 시험과 합격 기록은 변경을 채택할 때뿐 아니라 거절할 때도 근거입니다. 앞서 본 스킬 고르기 단계에 판단 모델을 더하는 변경은 정확도를 2.2%p 올렸지만 응답 시간이 약 30배 늘었고, 병목이 스킬 고르기보다 실행 단계에 있다는 점이 함께 드러나 반입하지 않았습니다. Anthropic이 가장 단순한 해법에서 시작해 결과가 실제로 좋아질 때만 복잡도를 더하라고 권하는 것도 같은 판단 기준입니다[S1]. 즉 변경이 운영에 들어가려면 기록 양식의 숫자가 실제로 좋아졌다는 증거가 있어야 하며, 「좋아 보인다」는 의견은 승인 근거로 쓰지 않습니다.

> 🎯 *결정 포인트 — 스킬의 운영 반입과 변경 승인 조건을 「전체 질문 예시 묶음의 합격선 충족」과 「기존 예시 회귀 시험 무결 통과」 두 가지로 정해 두면, 승인자는 스킬 코드를 읽지 않고 기록 양식만으로 결재할 수 있습니다.*

8.3.2 모델 교체 전후 같은 예시 묶음으로 비교하는 결과 일치 확인

운영 모델 교체가 결과를 바꾸는 경로

운영 모델을 바꿀 때도 기존 질문 예시 묶음을 그대로 다시 실행해 결과 일치를 확인합니다. 판단을 결정적 코드로 옮긴 스킬이라면 모델이 바뀌어도 업무 결과는 같아야 하는데, 모델이 맡는 일이 요청 받기, 명령 한 줄 실행, 결과 전달로 좁혀져 있기 때문입니다. 다만 그 좁은 일에서도 모델에 따라 차이가 생깁니다. vLLM에서 도구 호출을 여러 번 묶어 낼지 여부는 모델에 달려 있고, Chat Completions 요청에는 모델별 채팅 템플릿이 필요합니다(vLLM 온라인 서빙 문서)[S10]. Qwen3 계열 모델은 생각 모드를 켜고 끄는 설정(`enable_thinking=False` , `/no_think`)을 제공하므로[S7], 같은 모델이라도 이 설정에 따라 응답 시간과 출력 형태가 달라집니다.

게이트웨이 뒤에서 두 모델을 나란히 비교하는 방법

MSAP.ai AAS 운영 환경에서는 모든 모델 호출이 사내 게이트웨이인 LiteLLM 하나를 거치므로, 운영 모델 교체는 에이전트 설정을 고치지 않고 게이트웨이의 모델 연결을 바꾸는 작업이 됩니다[S9]. LiteLLM은 여러 LLM 제공자를 OpenAI 호환 호출 하나로 묶고, 배포 간 부하 분산과 장애 시 우회, 재시도, 관측 도구로의 로그 전송을 제공합니다(LiteLLM 게이트웨이 문서)[S9]. IT 담당자는 이 기능을 이용해 기존 모델과 새 모델을 게이트웨이에 함께 등록하고 시험용 프로파일만 새 모델로 연결해 같은 예시 묶음을 실행할 수 있으며, 비교가 끝나기 전까지 운영 프로파일은 기존 모델을 그대로 씁니다. 생성 단계의 처리 시간은 토큰마다 읽어야 하는 가중치 양에 좌우되므로[S5], 활성 파라미터 수가 다른 모델로 바꾸면 소요 시간 열이 가장 먼저 달라집니다.

점검 항목	확인 방법	합격 조건
업무 결과 일치	전체 예시 묶음의 결과 JSON을 교체 전 기록에 대조	모든 예시가 교체 전 결과 그대로
첫 줄 라우팅	업무가 섞인 예시에서 고른 스킬 비교	교체 전과 같은 스킬 선택
전달 충실도	진행기 출력과 최종 응답 문구 비교	원문 교체 훅 적용 후 진행기 원문 그대로
도구 호출 형식	도구 호출이 실패 없이 실행됐는지 확인	형식 오류 0건
생각 모드·양자화 등 모델 설정	게이트웨이와 서빙 설정 기록 대조	설정값이 기록 양식에 남아 있음
합격선 세 지표	기록 양식으로 전후 비교	모든 예시가 합격선 충족

모델 교체 비교가 남기는 운영 판단 근거

이 점검표를 통과한 교체만 운영 프로파일에 반영합니다. 영업기회 등록 사례에서 결과 JSON이 프론티어 모델이 같은 명령을 실행한 결과와 같았다는 점은, 판단을 코드로 옮긴 스킬은 모델 규모가 크게 달라도 같은 결과를 낼 수 있다는 것을 보여 줍니다. 즉 결과 일치 확인은 운영 모델을 자기 업무의 시험 결과로 고르게 해 주며, 더 작은 모델이나 다른 공급처 모델로 옮길 때의 위험도 같은 방식으로 잴 수 있습니다.

프로파일 단위의 모델 선택과 시험용 프로파일

모델은 프로파일마다 따로 정할 수 있으므로, 교체 비교도 운영 프로파일을 직접 바꾸지 않고 같은 스킬을 올린 시험용 프로파일에서 먼저 진행하는 편이 안전합니다. 같은 성격과 같은 스킬을 써도 모델이 바뀌면 결과 문구나 도구 호출 형식이 달라질 수 있기 때문입니다. 고객·금액 자료를 다루는 업무 프로파일에는 검증을 마친 로컬 모델만 지정해 두면, 다른 프로파일에서 진행하는 교체 시험이 그 업무의 결과에 영향을 주지 않습니다.

예약 작업도 같은 원칙을 따릅니다. Hermes Agent는 예약 작업이 쓸 모델을 작업별 고정값, 공통 기본값, 전역 설정의 순서로 정하므로, 결과 형식이 중요한 정기 업무는 작업 단위로 모델을 고정해 두면 전역 설정을 바꾸는 교체 시험 중에도 그 작업의 실행 조건이 유지됩니다. 반대로 고정하지 않은 예약 작업은 상위 설정 변경을 그대로 따르므로, 모델 교체 비교 대상 목록에 예약 작업도 함께 올려야 합니다.

8.4 업무가 바뀌면 업무 정의서부터 고치는 스킬의 생애 주기와 업무 전문가의 소유

스킬은 SKILL.md 파일과 스크립트, 참고 파일을 담은 디렉터리입니다[S2]. Hermes Agent는 `~/.hermes/skills/` 디렉터리를 스킬의 기본 디렉터리이자 정본으로 삼으며[S8], 실행 환경에서는 이 파일들이 곧 에이전트의

동작을 정합니다. 그러나 스킬 파일은 업무 정의서를 프론티어 모델이 코드로 옮긴 결과이므로, 업무가 무엇이고 무엇이 정답인지에 대한 정본은 업무 전문가가 쓴 업무 정의서와 질문 예시 묶음입니다. 업무가 바뀌면 스킬 파일을 직접 고치지 않고 업무 정의서부터 고친 뒤 제작과 시험을 다시 거칩니다.

이 원칙이 없으면 운영 중 스킬 파일만 여러 번 고쳐지고 업무 정의서는 처음 판에 머물러, 스킬이 왜 그렇게 동작하는지 아는 사람이 담당자 한 명뿐인 상태에 이릅니다. 그 담당자가 부서를 옮기거나 퇴사하면 스킬은 고칠 수 없는 코드가 되어 기술 부채로 남습니다. 반대로 업무 정의서와 질문 예시가 최신 상태로 남아 있으면, 후임자는 같은 정의서로 프론티어 모델에게 스킬을 다시 만들게 하고 같은 예시 묶음으로 결과를 확인할 수 있습니다.

8.4.1 정의, 제작, 시험, 반입, 운영, 개정으로 이어지는 스킬 생애 주기

스킬 생애 주기 여섯 단계와 되돌아가는 경로

스킬의 생애는 정의, 제작, 시험, 반입, 운영, 개정 여섯 단계로 이어지며, 시험 불합격이나 운영 중 실패가 생기면 앞 단계로 되돌아갑니다.

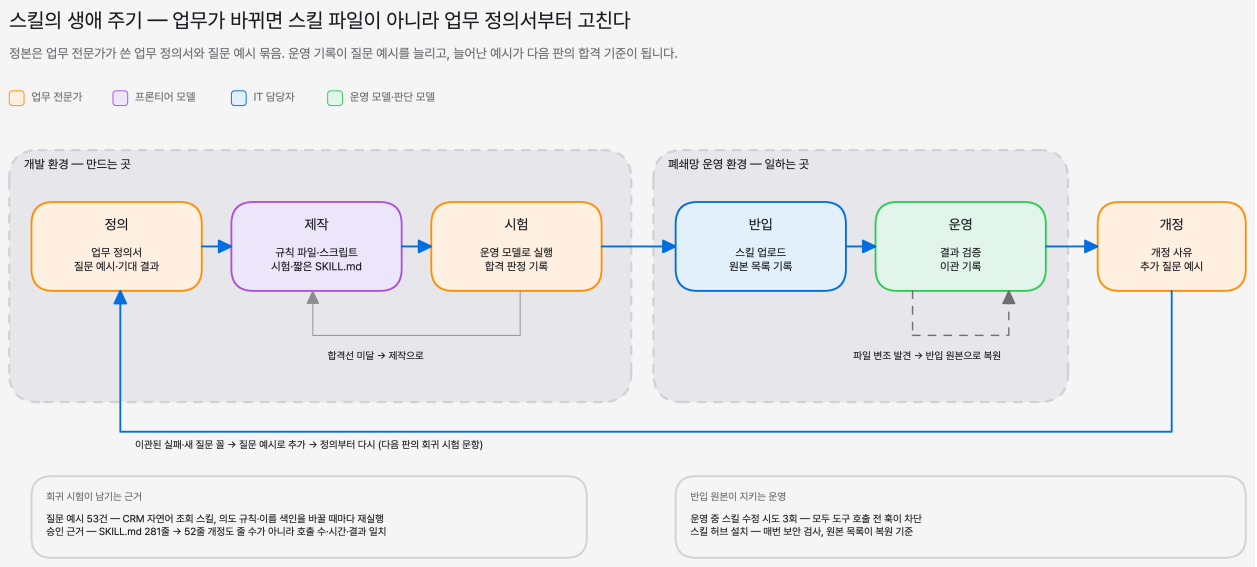


그림 8-2. 업무 전문가 스킬의 생애 주기 — 업무 정의에서 운영까지, 그리고 실패·피드백·변조 복구로 되돌아가는 경로

단계	산출물	주 담당	다음 단계로 가는 조건
정의	업무 정의서(지시문 해석 규칙, 결정적 코드로 옮길 판단 목록, 질문 예시·기대 결과)	업무 전문가	질문 예시마다 기대 결과가 있음
제작	규칙 파일, 결정적 스크립트, 시험, 짧은 SKILL.md, 플러그인 훅	프론티어 모델(MCP로 작성), 업무 전문가(지시)	스크립트 수준 시험 통과
시험	합격 판정 기록	업무 전문가	전체 예시가 합격선 충족

단계	산출물	주 담당	다음 단계로 가는 조건
반입	스킬 업로드 기록, 원본 목록	IT 담당자	반입 승인
운영	결과 검증 기록, 이관 기록	운영 모델, 판단 모델, 담당자	상시
개정	개정 사유, 추가 질문 예시	업무 전문가	정의 단계로 복귀

반입 단계와 변조 복구에서 쓰는 원본 목록

반입 단계에서는 개발 환경에서 합격한 스킬을 스킬 업로드나 스킬 허브로 폐쇄망 운영 환경에 옮기고, 반입한 파일의 원본 목록을 함께 남깁니다. Hermes Agent는 스킬 허브에서 스킬을 설치할 때마다 보안 검사를 거치도록 되어 있어[S8], 외부에서 받은 스킬이 검사 없이 운영 환경에 들어가는 것을 막습니다. 원본 목록은 운영 중 변조 복구의 근거입니다. 에이전트가 대화 뒤 배경 검토 과정에서 스킬 파일을 고치는 경우가 있는데, MSAP.ai AAS는 도구 호출 전 훅으로 스킬 수정 시도를 막고 원본 목록으로 파일을 복원합니다. 한 업무 에이전트에서는 운영 중 스킬 수정 시도 3회가 모두 이 훅에 막혔습니다. 다만 쓰기 승인 대기 같은 실행 환경 설정과 함께 써야 남는 위험을 줄일 수 있으며, 그 설정은 운영 점검 항목으로 따로 관리합니다(9.2절).

개정 단계에서 운영 기록을 질문 예시로 바꾸는 순서

개정 단계의 입력은 운영 단계가 남긴 기록입니다. 사람에게 이관된 실패 가운데 같은 유형이 반복되면, 업무 전문가는 그 요청 문장을 질문 예시로 추가하고 기대 결과를 적은 뒤 업무 정의서에 처리 규칙을 보탭니다. 그러면 운영 중 발견한 실패가 다음 판의 회귀 시험 문항이 되어, 한 번 고친 실패가 다시 나타나지 않는지를 회귀 시험이 매번 확인합니다. 스킬의 생애 주기는 운영 기록이 질문 예시를 늘리고, 늘어난 질문 예시가 다음 판의 합격 기준을 높이는 방식으로 순환합니다.

운영 단계에서 상태를 먼저 드러내는 관측 화면

운영 단계의 업무 에이전트는 대부분 사람이 보지 않는 시각에 예약 작업으로 실행되므로, 실패가 조용히 쌓이지 않게 하는 관측이 개정 단계의 입력을 만듭니다. MSAP.ai AAS의 관측 화면은 예약 작업의 실패와 멈춤, 실패해도 알림이 가지 않는 작업, 만료가 다가온 자격증명, 스킬이 실제로 부르는 외부 시스템을 숫자와 목록으로 보여 줍니다. 스킬 하나가 어느 예약 작업에서 시작해 어떤 스크립트를 거쳐 어느 MCP 서버와 외부 호스트에 닿는지도 흐름 그림으로 드러나므로, 결과 확인자는 에이전트가 무엇을 했는지 따라갈 수 있습니다.

자격증명 쪽에서는 토큰 하나를 쓰는 곳을 거꾸로 찾는 기능이 운영 부담을 줄입니다. 업무별 에이전트가 늘어날수록 같은 토큰을 공유하는 스킬도 늘어나므로, 쓰는 곳 목록 없이 토큰을 교체하면 어느 업무가 멈출지 예측하기 어렵습니다. 자격증명 상세 화면이 같은 값이 저장된 위치와 그 값을 쓰는 스크립트를 함께 보여 주면, IT 담당자는 값을 한 번도 열어 보지 않고도 교체 작업의 영향 범위를 미리 정할 수 있습니다.

8.4.2 업무 정의서 소유자로서 업무 전문가와 IT 담당자의 책임 분담

업무 정의서와 정답 판정을 맡는 업무 전문가

업무 전문가는 업무 정의서의 소유자이며, 업무가 무엇이고 무엇이 정답인지를 정하는 유일한 담당자입니다. 업무 정의서 작성과 개정, 질문 예시·기대 결과 작성, 합격 판정 기록의 정답 열 판정이 업무 전문가의 일입니다. 이 일들은 모두 문장과 표로 이루어지므로 코드를 몰라도 수행할 수 있고, 스크립트 작성과 수정은 프론티어 모델이

MCP를 통해 맡습니다. 따라서 업무 전문가를 스킬의 소유자로 두는 것은 업무 지식을 가진 사람이 스킬 품질을 최종 판정한다는 뜻이며, 개발자가 업무를 추측해 코드를 고치는 상황을 막습니다.

반입과 운영 통제를 맡는 IT 담당자

IT 담당자는 스킬이 만들어지는 환경과 실행되는 환경을 통제합니다. 개발 환경의 비식별 데이터베이스와 목업 인터페이스 구성, 프론티어 모델과 MSAP.ai AAS MCP 서버 사이의 연결과 권한, 스킬 업로드 반입 승인, 운영 모델 교체, 변조 감시와 원본 복원이 IT 담당자의 일입니다. LiteLLM 게이트웨이는 사용자·팀별로 모델과 예산, 속도 제한을 거는 가상 키와 비용 집계 기능을 제공하므로[S9], IT 담당자는 이 기능으로 부서별 사용량을 확인하고 이상 호출을 찾아낼 수 있습니다.

책임 영역	업무 전문가	IT 담당자	의사결정권자
업무 정의서·질문 예시	작성·개정·소유	보관 위치 제공	대상 업무 선정
스킬 제작	프론티어 모델에게 말로 지시	개발 환경과 MCP 연결 제공	—
합격 판정	정답 판정과 기록	측정 환경 제공	합격선 기준값 승인
반입	반입 요청	스킬 업로드 승인과 원본 목록 관리	반입 조건 승인
운영	업무 관련 이관 처리	시스템 관련 이관 처리, 변조 복원, 게이트웨이 통제	이관 기록 검토
개정·모델 교체	개정 사유 작성, 예시 추가	모델 교체 비교 실행	변경 반영 승인

담당자 교체 뒤에도 스킬을 유지하는 기록 네 가지

이 분담에서는 스킬을 이해하는 데 필요한 정보가 업무 정의서, 질문 예시 묶음, 합격 판정 기록, 원본 목록이라는 네 가지 문서에 남고, 특정 담당자의 기억에 기대지 않습니다. 업무 전문가가 바뀌어도 후임자는 정의서와 예시 묶음을 이어받아 같은 기준으로 판정할 수 있고, IT 담당자가 바뀌어도 반입 기록과 원본 목록으로 운영 상태를 재현할 수 있습니다. 의사결정권자는 합격선 기준값, 반입 조건, 변경 반영을 승인하는 세 가지 결정만 맡고, 각 결정의 근거는 기록 양식에 이미 숫자로 남아 있습니다. 즉 MSAP.ai AAS로 운영하는 스킬은 남긴 기록에 기대어 유지하는 구조이고, 그 덕분에 업무 전문가가 말로 만든 스킬이 담당자가 바뀐 뒤에도 조직의 자산으로 남습니다. 다만 이 기준과 운영 체계를 갖춰도 작은 모델에 맡기기 어려운 업무와 운영 중에 새로 생기는 위험은 남으며, 그 한계와 한계를 감안한 도입 순서는 9장에서 정리합니다.

작업공간에 남는 스킬과 사람이 읽는 규칙 페이지

MSAP.ai AAS에서는 사용자마다 AI 에이전트 작업공간이 하나씩 만들어지고 프로파일·스킬·예약 작업·기억과 파일이 그 안에 저장됩니다. 업무 전문가가 말로 정한 규칙은 이 작업공간의 규칙 파일과 스킬로 남고, 프론티어 AI는 같은 규칙 파일에서 사람이 읽는 규칙 페이지를 함께 만들 수 있습니다. 담당자가 부서를 옮기거나 회사를 떠나도 후임자는 규칙 페이지와 업무 정의서를 읽고 같은 스킬을 이어 쓰므로, 개인의 노하우가 조직의 기록으로 바뀝니다. 아래 그림은 사용자 한 사람의 작업공간이 담는 구성 요소와 그 업무상 의미를 보여 줍니다.



그림 8-3. 사용자별 AI 에이전트 작업공간의 구성 요소와 업무에서의 의미

한편 IT 담당자는 관리자·팀 관리자·작업공간 주인·일반 사용자로 나뉜 권한으로 누가 어떤 에이전트와 스킬에 접근하는지를 정하므로, 업무 정의서의 소유와 실행 환경의 통제가 서로 다른 사람에게 놓입니다. 작업공간을 복제하면 설정은 따라오지만 비밀값은 따라오지 않아, 후임자는 자기 자격증명을 새로 입력한 뒤에만 같은 스킬로 업무 시스템에 쓸 수 있습니다.

9장: 남은 한계와 도입 순서

MSAP.ai AAS는 업무 전문가가 말로 정의한 업무를 프론티어 AI가 결정적 코드 중심의 스킬로 만들고, 조직 안의 활성 파라미터 3B 급 오픈소스 MoE 모델이 그 스킬을 같은 결과로 실행하게 하는 구성입니다. 이 구성은 규칙을 문장으로 쓸 수 있는 반복 업무에서 응답 시간과 정답률을 함께 개선하지만, 모든 업무에 같은 효과를 내지는 않으며 운영 중에 새로 생기는 위험도 있습니다. 게다가 작은 모델이 에이전트 호출 대부분에 충분하다는 판단의 대표 근거는 NVIDIA 연구진의 입장 논문이고, 같은 정확도를 실험으로 증명한 결과는 아닙니다[S4]. 도입 범위를 정하기 전에 한계를 먼저 확인해야 과대 기대로 인한 파일럿 실패를 막을 수 있습니다. 이 장은 작은 모델에 아직 어려운 업무 유형과 운영 중 스킬 자가 수정 위험을 먼저 밝히고, 파일럿 대상 업무의 선정 기준과 첫 업무 정의서 작성부터 부서 확산까지의 단계별 결정 항목을 제시합니다.

9.1 작은 모델에 아직 어려운 영역 — 모호한 요청, 범용 대화, 코드로 옮기지 못한 판단

결정적 코드 중심 스킬의 효과는 업무 규칙을 문장으로 쓸 수 있고 결과의 정답 여부를 판정할 수 있는 반복 업무에서 가장 크게 나타납니다. 반면 요청 문장이 매번 다르게 해석되는 업무, 판단 근거를 규칙으로 적을 수 없는 업무, 대화 자체가 목적인 업무에서는 코드로 옮길 판단이 적어 작은 모델의 약점이 그대로 드러납니다. Anthropic은 에이전트 시스템을 설계할 때 가장 단순한 해법부터 찾고 결과가 실제로 좋아질 때만 복잡도를 더하라고 권하며, 이 원칙은 어떤 업무를 작은 모델에 맡길지 고르는 판단에도 그대로 적용됩니다(Building effective agents [S1]).

9.1.1 결정적 코드로 옮기기 어려운 업무와 큰 모델이 여전히 필요한 자리

매번 판단이 달라지는 업무의 코드 이관 한계

결정적 코드가 맡는 일은 값 추출, 후보 선정, 확인 게이트, 쓰기, 재조회 검증처럼 같은 입력에 같은 결과가 나와야 하는 처리입니다. 그런데 고객사와의 거래 이력과 그 분기의 영업 상황을 함께 고려해 할인 폭을 정하는 일처럼 판단 기준이 사례마다 달라지는 업무는 규칙 문장으로 고정하기 어렵습니다. 이런 업무를 그대로 스킬로 만들면 규칙 파일이 예외 목록으로 채워지고, 예외 목록에 없는 요청은 결국 모델의 추측으로 처리됩니다. MSAP.ai AAS의 진행기는 모르는 값이 있으면 정해진 질문을 출력하고 멈추므로, 판단 근거가 문장으로 정리되지 않은 업무에서는 되묻기가 반복될 뿐 처리가 끝나지 않습니다. 판단이 매번 달라지는 업무는 사람이 결론을 내고, 스킬은 자료 수집과 초안 작성까지만 맡도록 범위를 줄이는 편이 안전합니다.

라벨 없는 자유 문장에서 값을 뽑는 요청의 되묻기

진행기는 요청 첫 줄과 본문에서 「고객사:」·「사업명:」 같은 라벨이나 정해진 형식을 기준으로 값을 뽑습니다. 라벨이 없는 자유 문장에서는 값의 위치를 코드가 확정하지 못하고, 이때 MSAP.ai AAS는 값을 추측해 기록하는 대신 정해진 질문으로 되묻습니다. 정확성 쪽에서는 올바른 동작이지만, 사용자가 받은 메일 본문을 그대로 붙여넣는 업무에서는 되묻기 횟수가 늘어 처리 시간이 길어집니다. 이 한계는 요청 양식을 정해 주거나, 자주 들어오는 문장 형태를 질문 예시로 모아 추출 규칙을 늘리는 방식으로 줄일 수 있습니다. 다만 완전히 없앨 수는 없으므로, 파일럿 대상을 고를 때 요청이 어떤 형태로 들어오는지를 반드시 함께 확인해야 합니다.

긴 추론과 범용 상담에 남는 큰 모델의 역할

여러 로그를 읽고 장애 원인을 추론하거나 계약 조항의 위험을 해석하는 업무는 처리 단계가 길고 매번 경로가 달라집니다. Anthropic의 구분으로 보면 이런 업무는 미리 정한 코드 경로를 따르는 워크플로보다 모델이 스스로 순서와 도구를 정하는 에이전트에 가깝습니다[S1]. 이 유형을 활성 3B 급 모델에 맡기면 도구 호출 횟수가 늘고 같은 요청에도 결과가 달라지며, 범용 상담 역시 코드로 옮길 판단이 거의 없어 같은 문제가 생깁니다. NVIDIA 연구진도 범용 대화가 꼭 필요한 업무에는 여러 모델을 섞는 이종 구성이 자연스럽다고 보았습니다(Small Language Models are the Future of Agentic AI [S4]). 폐쇄망 조직은 이런 업무를 큰 모델을 사내에 두는 별도 투자로 다루거나 첫 파일럿 범위에서 빼고, 그 업무 안에서 반복되는 후속 처리만 골라 스킬로 만드는 방식을 택해야 합니다.

결과 글만 보는 판단 모델의 검증 한계

MSAP.ai AAS의 결과 검증 순서는 코드가 실행하고, 판단 모델이 요청 충족 여부와 실패 유형을 판정하고, 코드가 재시도 1회 또는 사람 이관을 정하는 구조입니다. 판단 모델은 결과로 나온 글만 보므로, 「등록했습니다」라는 답이 실제 등록으로 이어졌는지는 판단 모델이 확인하지 못합니다. 그래서 실제 생성 여부는 코드가 재조회로 먼저 확인하고, 판단 모델은 그다음 단계에서 내용의 충족 여부만 판정하도록 순서를 고정합니다. 결과적으로 판단 모델은 사람의 최종 확인을 대신하지 못하며, 쓰기가 있는 업무에서는 사람 확인 1회를 진행기 안에 남겨 두어야 합니다. 작은 모델에 아직 어려운 업무 유형과 권하는 처리를 정리하면 다음과 같습니다.

업무 유형	작은 모델에서 생기는 현상	권하는 처리
매번 판단이 달라지는 업무	규칙 밖 요청이 모델 추측으로 넘어감	사람이 판단, 스킬은 자료 수집·초안
라벨 없는 자유 문장 입력	값 추출 실패로 되묻기 반복	요청 양식 지정, 질문 예시로 추출 규칙 보강
긴 추론이 필요한 업무	도구 호출 증가, 결과 불일치	큰 모델 또는 사람, 후속 반복 처리만 스킬화
범용 상담	코드로 옮길 판단이 거의 없음	파일럿 대상에서 제외
결과 글에만 의존하는 검증	실제 생성 여부를 확인하지 못함	코드 재조회를 판단 모델보다 앞에 배치

9.1.2 입장 논문 근거의 한계와 조직마다 자기 업무로 측정해야 하는 이유

NVIDIA 입장 논문이 주장하는 범위

NVIDIA 연구진은 에이전트 안의 언어 모델이 적은 수의 전문 작업을 거의 변화 없이 반복한다는 관찰에서 출발해, 작은 언어 모델이 에이전트 호출 대부분에 충분히 강력하고 더 알맞으며 더 경제적이라고 주장합니다. 연구진은 큰 모델 기반 에이전트를 작은 모델로 옮기는 일반 절차도 함께 제안했으며, 이 논문은 2025년 6월 첫 판 이후 2026년 9월 세 번째 판까지 개정되었습니다[S4]. 다만 이 논문은 방향을 제시하는 입장 논문이고, 특정 업무에서 작은 모델이 큰 모델과 같은 정확도를 낸다는 것을 비교 실험으로 보인 연구는 아닙니다. S4를 인용할 때는 「작은 모델이 반복 업무에 맞을 수 있다」는 설계 가설의 근거로만 쓰고, 조직의 도입 효과를 보장하는 근거로 써서는 안 됩니다.

입장 논문과 제품 관찰, 조직 측정의 근거 수준 차이

이 백서에 나오는 근거는 성격이 서로 달라서, 의사결정권자는 각 근거가 어디까지를 말해 주는지 구분해서 읽어야 합니다. 모델 구조 자료는 활성 파라미터가 작은 MoE 모델이 토큰마다 읽는 가중치가 적다는 사실을 보여 주지만, 업무 정답률에 대해서는 말하지 않습니다[S6][S7]. 한 업무 에이전트에서 MSAP.ai AAS 구성으로 운영했을 때 관찰된 전후 동작은 설계 방향이 맞다는 예시이지만, 요청 형태·데이터·장비가 다른 조직에 그대로 옮겨지지는 않습니다. 근거 수준을 정리하면 다음과 같습니다.

근거	성격	의사결정에서 쓰는 방식
NVIDIA 연구진 입장 논문[S4]	방향 주장, 비교 실험 아님	작은 모델 구성이 맞을 수 있다는 가설
Mixtral·Qwen3 공개 자료[S6][S7]	모델 구조 사양	활성 파라미터와 생성 속도의 관계 설명
MSAP.ai AAS 구성의 전후 관찰	특정 업무·특정 모델의 동작	설계 원칙의 예시, 다른 업무에 그대로 적용 불가
조직이 자기 업무로 잰 합격선 결과	자기 업무·자기 모델·자기 장비	운영 반영 판단의 직접 근거

자기 업무와 합격선으로 다시 재야 하는 이유

한 CRM 자연어 조회 업무에서는 결정적 조회 코드로 판단을 옮긴 뒤 394초·도구 36회·오답이던 처리가 14초·도구 0회·정답으로 바뀌었지만, 이 결과는 그 업무의 질문 형태와 그 운영 모델에서 나온 것입니다. 그 조회 업무를 고정한 회귀 시험이 53건 규모였던 것처럼, 업무 하나에 모이는 질문 예시는 아직 수십 건 수준이라 통계적으로 넓은 결론을 내기에는 표본이 작습니다. 조직은 MSAP.ai AAS 채팅에 실제 요청을 넣어 도구 호출 3회 이하, 수 분 이내 응답, 정답이라는 합격선(8.1)을 자기 업무로 직접 재야 합니다. 운영 모델을 바꾸거나 업무 규칙이 바뀔 때에도 같은 질문 예시 묶음으로 다시 재서 결과 일치를 확인해야 합니다(8.3). 즉 외부 근거는 파일럿을 시작할 이유를 주고, 운영 반영 여부는 조직 스스로 잰 숫자가 정합니다.

9.2 운영 중 에이전트의 스킬 자가 수정 위험과 쓰기 승인 대기 설정

MSAP.ai AAS가 실행 환경으로 쓰는 오픈소스 Hermes Agent는 에이전트가 운영 중에 스스로 스킬을 만들고 고치고 지울 수 있는 기능을 기본으로 갖추고 있습니다[S8]. 이 기능은 개인 비서형 에이전트에서는 경험을 쌓는 장치로 쓰이지만, 프론티어 AI가 만들고 회귀 시험으로 고정한 업무 스킬에서는 검증받지 않은 변경이 운영 스킬에 들어가는 경로가 됩니다. 운영 승인권자는 이 위험이 설정과 훅으로 줄어들 뿐 완전히 사라지지는 않는다는 점을 알고 운영을 승인해야 합니다.

9.2.1 Hermes Agent의 절차 기억 기능이 운영 스킬을 바꾸는 경로

Hermes Agent 절차 기억의 동작 방식

Hermes Agent는 스킬을 `~/.hermes/skills/` 디렉터리에 YAML 앞머리를 단 SKILL.md 형식으로 두며, 공식 문서는 이 디렉터리를 스킬의 정본이라고 설명합니다. 에이전트는 `skill_manage` 도구로 이 디렉터리의 스킬을 직접 만들고 고치고 지울 수 있고, 공식 문서는 이 기능을 에이전트의 절차 기억이라고 부릅니다(Hermes Agent 스킬 문서 [S8]). 운영 스킬의 정본이 놓인 디렉터리에 에이전트 자신이 쓰기 권한을 가진 구조입니다. 이 구조에서는 사람이 아무것도 바꾸지 않아도 운영 중인 스킬 파일의 내용이 달라질 수 있습니다.

운영 스킬이 바뀌는 네 가지 경로

운영 스킬이 바뀌는 경로는 대화 중 도구 호출, 대화 뒤 배경 검토, 스킬 허브 설치, 사람의 직접 수정 네 가지입니다. 이 가운데 배경 검토는 대화가 끝난 뒤 모델이 그 대화를 돌아보고 더 나은 절차를 스킬에 반영하려는 동작이라, 사용자가 변경 사실을 알기 어렵습니다. 스킬 허브 설치는 설치마다 보안 검사를 거치지만[S8], 외부에서 들어온 스킬이 기존 업무 스킬과 이름이나 역할이 겹치면 라우팅 결과가 달라질 수 있습니다. 네 경로를 정리하면 다음과 같습니다.

변경 경로	바뀌는 대상	운영상 위험
대화 중 `skill_manage` 호출	운영 스킬 본문·스크립트	회귀 시험을 거치지 않은 변경
대화 뒤 배경 검토	모델이 고쳐 쓴 스킬 파일	사용자가 모르는 사이의 변경
스킬 허브 설치	새 스킬 추가	기존 업무 스킬과 라우팅 충돌
운영 환경에서 사람이 직접 수정	개별 파일	개발 환경 정본과 어긋남

작은 모델이 고친 스킬에서 사라지는 확인 게이트

이 경로가 업무 스킬에 특히 위험한 이유는 고치는 주체가 활성 3B 급 운영 모델이라는 점에 있습니다. 결정적 코드로 확인 게이트를 쥐도록 설계한 스킬을 작은 모델이 「더 간단하게」 고치면, 사람 확인 단계나 재조회 검증이 빠진 절차로 바뀔 수 있습니다. 작은 모델에게 「결정적 코드로」라는 한 줄 지시만 주고 스킬을 다시 쓰게 했을 때, SKILL.md는 142줄에서 62줄로 줄었지만 확인 게이트는 여전히 모델 판단으로 남았습니다(5.4). 한 업무 에이전트를 MSAP.ai AAS 구성으로 운영하는 동안에는 모델의 스킬 수정 시도가 세 차례 나왔고, 도구 호출 전 훅이 세 번 모두 차단했습니다. 자가 수정 기능은 켜 두면 언젠가 쓰이는 기능이므로, 막는 장치를 운영 설정으로 미리 갖춰야 합니다.

9.2.2 쓰기 승인 대기와 원본 복원으로 남는 위험을 줄이는 운영 설정

`skills.write_approval` 설정의 효과와 한계

Hermes Agent는 선택 설정 `skills.write_approval`로 에이전트의 스킬 쓰기를 바로 반영하지 않고 사람 검토 대기로 돌릴 수 있습니다[S8]. 이 설정을 켜면 에이전트의 스킬 쓰기가 승인 대기 목록에 쌓이고, 승인 전에는 그 변경이 운영 스킬에 반영되지 않습니다. 다만 승인 대기는 변경을 멈춰 세울 뿐이고, 승인하는 사람이 내용을 확인하지 않고 통과시키면 위험은 그대로 남습니다. 승인 대기 건은 IT 담당자가 혼자 처리하지 않고, 업무 정의서의 소유자인 업무 전문가가 정의서와 대조해 판단하도록 분담해야 합니다.

도구 호출 전 훅과 원본 매니페스트 복원

MSAP.ai AAS는 승인 대기 설정에 더해 두 가지 장치를 함께 씁니다. 하나는 운영 프로파일에서 스킬 파일을 고치려는 도구 호출을 실행 전에 차단하는 Hermes Agent 플러그인의 도구 호출 전 훅이고(6.4.2), 다른 하나는 스킬을 반입할 때 남긴 원본 매니페스트와 비교해 달라진 파일을 원본으로 되돌리는 복원 장치입니다. 두 장치를 함께 쓰면 운영 스킬을 바꾸는 정식 경로는 개발 환경에서 고치고, 질문 예시 회귀 시험을 통과시키고, 스킬 업로드로 다시 반입하는 하나로 좁혀집니다. 훅은 Hermes Agent의 플러그인 기능에 의존하므로, Hermes Agent 버전을 올릴 때마다 훅이 같은 시점에 호출되는지 다시 확인해야 합니다.

화면 편집에도 남는 변경 내역과 이전 버전

사람이 화면에서 직접 고치는 경로도 같은 원칙으로 관리됩니다. MSAP.ai AAS 화면에서 고칠 수 있는 대상은 성격, 설정, 스킬 문서, 예약 작업 지시문 같은 글이며, 저장하기 전에 바뀐 줄을 먼저 보여 주고 글마다 이전 버전을 40개까지 남깁니다. 승인 대기 건을 검토하던 업무 전문가가 잘못 반영된 문구를 발견하면 이전 버전으로 되돌릴 수 있고, 무엇이 언제 바뀌었는지를 업무 정의서와 대조할 수 있습니다. 다만 규칙 점검 코드나 시험 코드까지 바뀌어야 하는 변경은 화면 편집 범위를 넘으므로, 개발 환경에서 프론티어 AI가 다시 만들고 회귀 시험을 통과한 뒤 스킬 업로드로 반입하는 경로를 따릅니다.

운영 승인 전 운영 설정 점검표

운영 승인권자가 반입 승인 전에 확인할 항목은 다음과 같습니다. 각 항목은 설정 하나로 끝나지 않고 담당자가 정해져 있어야 실제로 유지됩니다.

점검 항목	설정·장치	담당
스킬 쓰기 승인 대기	`skills.write_approval` 켜기[S8]	IT 담당자
스킬 수정 도구 차단	운영 프로파일 도구 호출 전 훅	IT 담당자
원본 복원	반입 때 남긴 원본 매니페스트 기준 복원	IT 담당자
변경 경로 단일화	개발 환경 수정 → 회귀 시험 → 스킬 업로드	업무 전문가·IT 담당자
승인 대기 건 검토	업무 정의서와 대조한 승인·반려	업무 전문가
버전 변경 재확인	Hermes Agent 버전 변경 시 훅 동작 시험	IT 담당자

이 점검표를 모두 갖춰도 남는 위험은 사람의 승인 품질과 훅이 막지 못하는 새 도구 경로입니다. 따라서 운영 반영 뒤에도 질문 예시 회귀 시험을 주기적으로 다시 실행해, 스킬 파일이 바뀌지 않았더라도 결과가 달라지지 않았는지를 확인하는 편이 안전합니다.

9.3 파일럿 대상 업무의 선정 기준 — 반복 빈도, 규칙 명시 가능성, 데이터 반출 제한

첫 파일럿 업무는 MSAP.ai AAS 구성이 효과를 낼 조건을 가장 많이 갖춘 업무여야 하며, 그 조건은 반복 빈도, 규칙 명시 가능성, 결과 판정 가능성, 데이터 반출 제한 네 가지로 정리됩니다. NVIDIA 연구진이 작은 모델의 근거로 든 특징도 에이전트 호출이 적은 수의 전문 작업을 거의 변화 없이 반복한다는 점이었습니다[S4]. 첫 업무 선택은 기술 검증의 범위를 정하는 일이며, 조건이 덜 맞는 업무로 시작하면 구성 전체가 효과 없다는 잘못된 결론으로 이어질 수 있습니다.

9.3.1 부서장이 후보 업무를 고르는 선정 점검표

반복 빈도와 규칙 명시 가능성 점검

반복 빈도는 같은 형태의 요청이 얼마나 자주 들어오는지를 봅니다. 자주 반복되는 업무일수록 업무 정의서와 회귀 시험을 만드는 데 들인 노력이 여러 번 회수되고, 질문 예시도 실제 요청에서 빠르게 모입니다. 규칙 명시 가능성은 업무 전문가가 처리 규칙과 예외를 문장으로 쓸 수 있는지를 봅니다. 「거래처 이름과 기간을 받아 미수금 목록을 조회한다」처럼 입력값·처리 규칙·완료 상태를 한 문장씩 쓸 수 있으면 프론티어 AI가 결정적 코드로 옮길

수 있습니다. 반대로 「상황을 봐서 적절히 처리한다」 외에 쓸 말이 없다면 그 업무는 아직 스킬로 만들 준비가 되지 않은 것입니다.

결과 판정 가능성과 데이터 반출 제한 점검

결과 판정 가능성은 결과가 맞았는지를 업무 전문가가 코드 없이 바로 판정할 수 있는지를 봅니다. 8.1절에서 정한 합격선의 세 번째 지표가 정답이므로, 정답을 가릴 수 없는 업무는 파일럿이 끝나도 합격 여부를 말할 수 없습니다. 데이터 반출 제한은 처리하는 자료가 조직 밖으로 나가면 안 되는지를 봅니다. 고객·금액 자료처럼 외부 AI 서비스에 보낼 수 없는 자료를 다루는 업무일수록, 사내 게이트웨이 뒤의 로컬 LLM에서 실행되는 MSAP.ai AAS 구성의 이점이 커집니다. 자료를 밖으로 보내도 되는 업무라면 외부 AI 서비스로도 충분할 수 있으므로, 이 항목은 이 구성을 택할 이유를 확인하는 질문이기도 합니다.

부서장용 선정 점검표

부서장은 후보 업무 두세 개를 아래 표로 점검하고, 「예」가 가장 많은 업무 하나를 파일럿 대상으로 정합니다. Anthropic이 권하는 대로 가장 단순한 해법에서 시작하려면, 같은 점수라면 쓰기가 없는 조회 업무를 먼저 고르는 편이 유리합니다[S1].

점검 항목	확인 질문
반복 빈도	같은 형태의 요청이 정기적으로 반복되는가
규칙 명시 가능성	입력값·처리 규칙·완료 상태를 문장으로 쓸 수 있는가
결과 판정 가능성	업무 전문가가 정답 여부를 바로 판정할 수 있는가
데이터 반출 제한	자료가 조직 밖으로 나가면 안 되는가
쓰기 영향 범위	잘못 기록했을 때 되돌릴 수 있는가
업무 전문가 확보	정의서를 쓰고 정답을 판정할 담당자가 있는가

업무 하나에 프로파일 하나를 두는 시범 구성

파일럿 업무를 정했다면 그 업무에 프로파일 하나를 배정하는 것으로 시범 구성을 시작합니다. 프로파일은 성격·모델·스킬·비밀값·기억·대화 기록·예약 작업을 각자 따로 갖는 독립 단위이므로, 업무 하나에 프로파일 하나를 두면 그 업무에 무엇을 맡기고 무엇을 보게 할지가 프로파일 경계와 그대로 일치합니다. 처음부터 여러 부서에 에이전트를 두면 어느 업무에서 무엇이 잘못됐는지 가려내기 어려우므로, 가장 단순한 구성에서 시작해 필요할 때만 넓히는 편이 안전합니다[S1]. 업무마다 요청을 받는 창구와 결과를 확인할 사람도 이 단계에서 함께 정해 두면, 서버를 준비하기 전에 종이 한 장으로 시범 구성이 정해집니다.

9.3.2 파일럿에 맞지 않는 업무의 예 — 범용 상담과 매번 판단이 달라지는 업무

선정 점검표를 통과하지 못하는 업무 예

파일럿에 맞지 않는 업무는 대부분 9.1의 한계와 겹칩니다. 사내 규정 전반을 묻는 범용 상담은 질문 범위가 정해지지 않아 회귀 시험으로 고정할 질문 예시를 만들 수 없고, 정답 여부도 사례마다 해석이 갈립니다. 영업 전략 수립이나 인사 평가처럼 매번 판단이 달라지는 업무는 규칙 문장으로 고정할 수 없어, 스킬이 결국 모델의 추측

에 기대게 됩니다. 1년에 한두 번 수행하는 업무는 반복 빈도가 낮아 업무 정의서와 시험을 만드는 노력이 회수되지 않습니다. 대표적인 부적합 업무와 그 이유는 다음과 같습니다.

업무 예	맞지 않는 이유
사내 규정 범용 상담	질문 범위가 열려 있어 질문 예시로 고정 불가
영업 전략 수립·인사 평가	판단 기준이 사례마다 달라짐
여러 로그를 엮는 장애 원인 분석	긴 추론과 매번 다른 처리 경로
자유 형식 메일에서 여러 값을 뽑아 바로 기록	되묻기 반복과 쓰기 오류 위험이 겹침
연 1~2회 수행하는 업무	정의서·시험 작성 노력이 회수되지 않음

부적합 업무에서 반복 부분만 떼어 내는 방법

부적합 업무 전체를 포기할 필요는 없으며, 그 안에서 반복되는 부분만 떼어 내면 파일럿 대상이 될 수 있습니다. 예를 들어 장애 원인 분석 자체는 사람이나 큰 모델이 맡더라도, 분석 전에 매번 하는 서버 목록 조회와 최근 변경 이력 수집은 입력·규칙·완료 상태가 분명한 조회 업무입니다. 영업 전략 수립도 판단은 사람이 하고, 판단에 쓰는 거래처별 실적 집계를 스킬로 만들 수 있습니다. 업무를 판단 부분과 반복 부분으로 나누는 작업 자체가 업무 정의서의 첫 단계이므로, 부서장은 업무 이름보다 그 안의 반복 처리 단위를 기준으로 후보를 적는 편이 좋습니다.

9.4 부서 반복 업무 하나로 시작하는 MSAP.ai AAS 파일럿 — 업무 정의서 작성에서 부서 확산까지

MSAP.ai AAS 파일럿은 부서 반복 업무 하나를 정해 업무 정의서를 쓰고, 개발 환경에서 스킬을 만들어 실제 운영 모델로 시험한 뒤, 합격선을 통과한 스킬만 스킬 업로드로 폐쇄망에 반입하는 순서로 진행됩니다. 단계마다 무엇을 승인할지가 정해져 있어야 파일럿 결과가 부서 확산 판단으로 바로 이어지며, 이 순서는 GPU 규모를 업무 측정 뒤에 정하도록 짜여 있습니다.

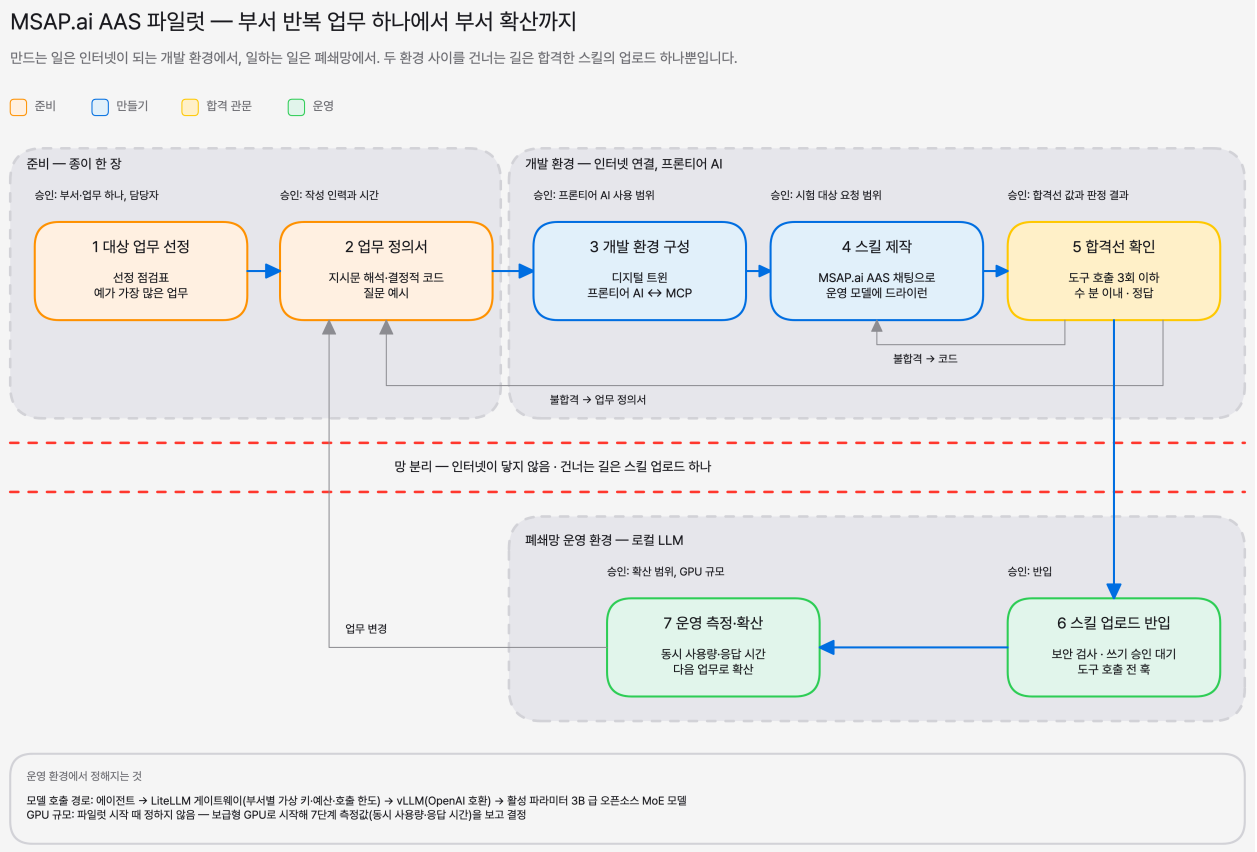


그림 9-1. 부서 반복 업무 하나로 시작해 부서 확산까지 이어지는 MSAP.ai AAS 파일럿 단계

9.4.1 지시문 해석과 결정적 코드, 질문 예시를 담은 첫 업무 정의서 작성

첫 업무 정의서의 작성 범위

첫 업무 정의서는 2.3의 양식을 그대로 쓰되, 범위를 업무 하나와 명령 한두 개로 좁혀 작성합니다. 지시문 해석 항목에는 요청 첫 줄이 어떤 형태일 때 어떤 명령을 부를지, 값이 모자랄 때 어떤 질문을 출력하고 멈출지를 적습니다. 결정적 코드 항목에는 값 추출 규칙, 후보 선정 기준, 사람 확인이 필요한 단계, 쓰기 대상, 쓰기 뒤 재조회로 확인할 항목을 문장으로 적고, 이 문장을 코드로 옮기는 일은 프론티어 AI가 맡습니다. Anthropic이 스크립트를 담은 스킬이 코드만 줄 수 있는 결정적 신뢰성을 준다고 설명한 대로, 이 항목이 길고 구체적일수록 작은 모델에게 남는 판단이 줄어듭니다(Agent Skills [S2]). 질문 예시 항목에는 실제로 들어온 요청 문장과 기대 결과를 짝지어 수십 건 단위로 모으며, 이 묶음이 그대로 회귀 시험이 됩니다.

업무 정의서 항목별 담당

업무 정의서는 업무 전문가가 소유하지만 모든 항목을 혼자 채우지는 않습니다. 업무 전문가는 규칙과 정답을 말로 정의하고, 프론티어 AI는 그 말을 코드와 시험으로 옮기며, IT 담당자는 반입 조건과 비밀값 분리를 맡습니다. 첫 파일럿에서는 쓰기가 없는 조회 업무로 시작해 결과 판정과 회귀 시험 운영에 익숙해진 뒤 등록 업무로 넓히는 편이 위험이 적습니다. 항목별 첫 작성 범위와 담당을 정리하면 다음과 같습니다.

정의서 항목	첫 작성 범위	담당
지시문 해석	첫 줄 형태별 명령, 값이 모자랄 때의 질문	업무 전문가

정의서 항목	첫 작성 범위	담당
결정적 코드	값 추출·후보 선정·확인 단계·쓰기·재조회 규칙 문장	업무 전문가 작성, 프론티어 AI 코드화
질문 예시	실제 요청 문장과 기대 결과 수십 건	업무 전문가
합격선	도구 호출 3회 이하, 수 분 이내, 정답	의사결정권자 승인
반입 조건	회귀 시험 통과, 비밀값 분리 보관	IT 담당자

9.4.2 개발 환경 구성부터 스킬 업로드 반입, 합격선 확인, 부서 확산까지의 결정 지점

개발 환경 구성과 스킬 제작 단계의 결정 항목

업무 정의서가 나오면 IT 담당자는 인터넷이 되는 개발 환경에 비식별 데이터베이스와 목업 인터페이스로 내부 시스템을 흉내 낸 디지털 트윈을 구성하고, 프론티어 AI를 MCP로 MSAP.ai AAS에 연결합니다. MCP 명세는 호스트가 도구를 부르기 전에 사용자 동의를 받도록 정하고 있으므로, 의사결정권자는 이 단계에서 프론티어 AI의 사용 범위와 개발 환경에 둘 자료의 범위를 승인해야 합니다(MCP 명세 2025-06-18 [S3]). 업무 전문가는 그다음 프론티어 AI와 함께 스킬을 만들고, MSAP.ai AAS 채팅으로 실제 운영 모델에 요청을 넣어 드라이런합니다. 합격 판정은 스킬을 만든 프론티어 AI의 실행 결과가 아닌 운영 모델의 결과로 하며, 합격선을 통과하지 못하면 업무 정의서나 코드로 되돌아가 고칩니다.

스킬 업로드 반입과 운영 측정 단계의 결정 항목

합격선을 통과한 스킬은 스킬 업로드로 폐쇄망 운영 환경에 반입되고, 이때 IT 담당자는 보안 검사 결과와 9.2.2의 운영 설정 점검표를 확인한 뒤 반입을 승인합니다. 운영 환경의 모델 호출은 LiteLLM 게이트웨이 하나를 거쳐 vLLM으로 서빙하는 로컬 LLM에 전달되며, 게이트웨이의 가상 키로 부서·사용자별 모델과 예산, 호출 한도를 나눠 관리할 수 있습니다(LiteLLM 게이트웨이 문서 [S9]). vLLM은 OpenAI 호환 방식으로 모델을 서빙하므로 게이트웨이 뒤의 모델을 바꿔도 스킬과 에이전트 쪽 구성은 그대로 유지됩니다(vLLM 온라인 서빙 문서 [S10]). 따라서 GPU 규모는 파일럿 시작 때 크게 정하지 않고, 보급형 GPU 장비에서 활성 3B 급 MoE 모델로 시작해 운영 측정에서 나온 동시 사용량과 응답 시간을 보고 정합니다. 부서 확산 범위도 같은 방식으로, 첫 업무의 운영 측정 결과와 업무 전문가 확보 여부를 보고 다음 업무를 정하는 순서로 넓힙니다.

단계별 결정 항목표

단계마다 의사결정권자와 IT 담당자가 정할 항목을 정리하면 다음과 같습니다. 앞 단계의 승인 없이는 다음 단계로 넘어가지 않으며, 합격선 확인에서 불합격이면 업무 정의서 작성이나 스킬 제작으로 되돌아갑니다.

단계	의사결정권자가 정할 항목	IT 담당자가 정할 항목
대상 업무 선정	부서·업무 하나, 업무 전문가 지정	데이터 반출 제한 여부 확인
업무 정의서 작성	작성 인력과 투입 시간 배정	비밀값·접근 권한 목록
개발 환경 구성	프론티어 AI 사용 범위 승인	디지털 트윈, MCP 연결
스킬 제작·드라이런	시험 대상 요청 범위	시험용 에이전트 프로파일
합격선 확인	합격선 값과 판정 결과 승인	도구 호출 수·응답 시간 기록

단계	의사결정권자가 정할 항목	IT 담당자가 정할 항목
스킬 업로드 반입	반입 승인	보안 검사, 쓰기 승인 대기·훅 설정
운영 측정·부서 확산	확산 범위, GPU 규모	게이트웨이 가상 키·예산, 모델 서빙

파일럿 결과가 보여 주는 MSAP.ai AAS 구성의 기술적 이점

이 순서를 마친 조직이 확인하게 되는 이점은 세 가지입니다. 첫째, 판단을 결정적 코드로 옮긴 스킬은 활성 3B급 오픈소스 MoE 모델에서도 요청당 도구 호출을 한 번 안팎으로 줄여, 보급형 GPU에서 수 분 이내의 응답을 낼 수 있습니다. 둘째, 만드는 일은 인터넷이 되는 개발 환경의 프론티어 AI가 맡고 일하는 일은 폐쇄망의 로컬 LLM이 맡으므로, 고객·금액 자료를 밖으로 보내지 않고도 최신 모델의 설계 능력을 쓸 수 있습니다. 셋째, 업무 정의서와 질문 예시 회귀 시험이 스킬의 정본으로 남기 때문에 운영 모델을 교체하거나 업무 규칙이 바뀌어도 같은 시험으로 결과 일치를 확인하고 변경을 승인할 수 있습니다. 이 세 가지 이점은 큰 모델과 고가 GPU를 먼저 정하는 도입으로는 얻기 어려우며, 업무 전문가가 업무를 말로 정의하는 단계에서 출발합니다.

> 🎯 *결정 포인트 의사결정권자가 이 문서를 덮고 정할 일은 하나입니다. 우리 부서의 반복 업무 하나를 선정 점검표로 골라, 지시문 해석·결정적 코드·질문 예시를 담은 업무 정의서를 쓰고 MSAP.ai AAS 파일럿 대상으로 지정하는 것입니다.*

부록 — 인용한 공개 자료 10건과 본문 용어 정의

본 부록은 백서 본문에 인용된 참고문헌과 본문에 등장하는 주요 기술 용어의 정의를 모아 정리한 참조 자료입니다.

Appendix A — 참고문헌 (References)

1. Anthropic (Erik Schluntz, Barry Zhang). (2024). "Building effective agents". https://www.anthropic.com/research/building-effective-agents — 워크플로와 에이전트의 구분, 단순한 구성부터 시작하는 설계 원칙, 실수 방지 도구 설계 사례
2. Anthropic (Barry Zhang, Keith Lazuka, Mahesh Murag). (2025). "Equipping agents for the real world with Agent Skills". https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills — SKILL.md 디렉터리 구조와 점진적 공개, 스크립트가 주는 결정적 신뢰성
3. Model Context Protocol 프로젝트. (2025). "Model Context Protocol Specification (2025-06-18 개정본)". https://modelcontextprotocol.io/specification/2025-06-18 — 호스트·클라이언트·서버 역할, JSON-RPC 2.0 메시지, 도구 호출 전 사용자 동의 원칙
4. Belcak 외 7인, NVIDIA Research. (2025). "Small Language Models are the Future of Agentic AI" (arXiv:2506.02153, v3 2026-09-22 개정). https://arxiv.org/abs/2506.02153 — 반복 업무 에이전트에서 작은 언어 모델의 역할을 주장한 입장 논문
5. NVIDIA (Shashank Verma, Neal Vaidya). (2023). "Mastering LLM Techniques: Inference Optimization". https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization/ — 프리필과 디코드 단계, 디코드 속도를 정하는 메모리 대역폭
6. Jiang 외 25인, Mistral AI. (2024). "Mixtral of Experts" (arXiv:2401.04088). https://arxiv.org/abs/2401.04088 — 전문가 혼합 구조와 활성 파라미터(접근 47B, 활성 13B)
7. Qwen 팀(알리바바). (연도 표기 없음, 2026-11-11 접속). "Qwen3-30B-A3B 모델 카드". https://huggingface.co/Qwen/Qwen3-30B-A3B — 전체 30.5B·활성 3.3B 파라미터, 전문가 128개 중 8개 선택, Apache-2.0 라이선스
8. Nous Research. (2026). "Hermes Agent 문서 — Skills". https://hermes-agent.nousresearch.com/docs/user-guide/features/skills — 스킬 디렉터리 관리, 점진적 공개, 스킬 허브 설치와 보안 검사
9. BerriAI. (연도 표기 없음, 2026-11-11 접속). "LiteLLM AI Gateway(Proxy) 문서". https://docs.litellm.ai/docs/simple_proxy — OpenAI 호환 게이트웨이, 가상 키·예산·비용 집계, 배포 간 부하 분산
10. vLLM 프로젝트. (연도 표기 없음, 2026-11-11 접속). "vLLM Online Serving (OpenAI 호환 서버)". https://docs.vllm.ai/en/latest/serving/online_serving/ — OpenAI 호환 서빙과 도구 호출, `parallel_tool_calls` 설정

Appendix B — 용어 정의 (Glossary)

본문에 등장하는 주요 기술 용어와 그 뜻을 정리했습니다.

용어	정의
가상 키(Virtual Key)	LiteLLM 게이트웨이가 사용자나 팀마다 발급하는 접속 키로, 쓸 수 있는 모델·예산·속도 상한을 정하고 비용을 따로 집계하는 데 씁니다.
결정적 코드(Deterministic Code)	같은 입력에 언제나 같은 결과를 돌려주는 프로그램입니다. 스킬에서는 값 추출, 후보 선정, 순서 진행, 확인 게이트, 쓰기, 재조회 검증을 맡아 결과의 일관성을 보장합니다.
되묻기	요청에 필요한 값이 정말 없거나 후보가 둘 이상일 때 진행기가 정해진 질문을 사용자에게 보내고 멈추는 단계입니다. 찾을 수 있는 값은 먼저 찾아 보여 주고 확인만 받는 편이 좋습니다.
드라이런(Dry Run)	업무 시스템에 실제로 쓰지 않고 쓰기 결과를 미리 계산해 사람에게 보여 주는 실행입니다. 승인 문구가 온 뒤에만 실제 쓰기로 넘어갑니다.
디지털 트윈 개발 환경	운영 업무 시스템과 같은 모양의 데이터와 인터페이스를 비식별 값으로 다시 만든 개발용 복제 환경입니다. 프론티어 AI는 이 환경에서만 스킬을 만들고 시험합니다.
로컬 LLM	조직 안의 서버에 올려 외부 모델 공급사를 거치지 않고 실행하는 언어 모델입니다. MSAP.ai AAS 운영 환경에서는 활성 파라미터 3B 급 오픈소스 MoE 모델을 vLLM으로 서빙합니다.
메모리 대역폭(Memory Bandwidth)	메모리에서 연산 장치로 데이터를 옮기는 속도입니다. 디코드 단계에서는 토큰마다 활성 파라미터를 읽어야 하므로 이 속도가 토큰 생성 속도를 좌우합니다.
메모리 한계 작업(Memory-bound)	연산량보다 메모리에서 데이터를 옮기는 속도가 지연을 정하는 작업입니다. NVIDIA는 LLM의 디코드 단계를 이 유형으로 분류합니다.
멱등 쓰기(Idempotent Write)	같은 요청을 여러 번 실행해도 한 번 실행한 것과 결과가 같은 쓰기입니다. 같은 키가 있으면 새로 만들지 않아 같은 영업기회가 두 건 생기는 일을 막습니다.
목업 인터페이스(Mock Interface)	CRM·결재·자산 관리 같은 내부 업무 시스템의 호출 규약을 흉내 내 정해진 응답을 돌려주는 가짜 서버입니다. 스크립트는 접속 대상 설정만 바꾸면 운영 환경에서 그대로 돕니다.

용어	정의
비식별 데이터베이스	운영 데이터베이스의 테이블 구조와 컬럼 이름, 값의 형식은 그대로 두고 값만 가짜로 바꾼 데이터베이스입니다. 동명 고객이나 빈 값 같은 예외 상황은 일부러 남겨 둡니다.
스킬(Skill)	SKILL.md 지시문·스크립트·참고 파일을 한 디렉터리에 묶어 에이전트가 필요할 때 불러 쓰는 업무 단위입니다.
스킬 업로드	MSAP.ai AAS 콘솔이나 MCP 도구로 스킬 디렉터리 묶음을 직접 올려 특정 에이전트에 켜는 반입 경로입니다.
스킬 허브(Skills Hub)	검토를 마친 스킬을 조직 내부 저장소에 게시하고 필요한 에이전트가 설치해 가는 반입 경로입니다. Hermes Agent는 설치할 때마다 보안 검사를 거칩니다.
실수 방지 설계(Poka-yoke)	도구의 입력 형식을 제한해 모델이 실수할 여지를 처음부터 없애는 설계입니다. 상대 경로 대신 절대 경로만 받게 해 경로 오류를 없앤 사례가 대표적입니다.
실행 모델	운영 환경에서 완성된 스킬의 명령을 실행하는 모델입니다. 판단은 코드가 맡고, 실행 모델은 진행기 명령 한 줄을 실행해 결과 원문을 그대로 전달합니다.
업무 전문가	코드를 쓰지 않아도 자기 업무의 판단 기준과 예외를 가장 정확히 아는 사람입니다. 업무 정의서와 질문 예시를 쓰고 결과의 정답을 판정합니다.
업무 정의서	업무 전문가가 한 장으로 채우는 문서로, 프론티어 AI가 스킬을 만들 때 받는 유일한 업무 입력입니다. 지시문 해석 규칙, 결정적 코드로 옮길 판단, 질문 예시가 들어갑니다.
에이전트(Agent)	LLM이 처리 과정과 도구 사용을 실행 중에 스스로 정하는 시스템입니다. 모델이 단계마다 결과를 보고 다음에 부를 도구를 직접 고릅니다.
워크플로(Workflow)	LLM과 도구를 미리 정한 코드 경로로 엮는 시스템입니다. 처리 순서를 코드가 정하고 모델은 정해진 단계 안에서 글을 만들거나 분류합니다.
이종 구성	용도에 따라 크기와 성격이 다른 모델을 섞어 쓰는 구성입니다. MSAP.ai AAS는 스킬을 만드는 프론티어 AI와 스킬을 실행하는 작은 모델을 나눠 둡니다.
입장 논문(Position Paper)	실험으로 결론을 증명하기보다 특정 관점과 주장을 제시하는 논문입니다. 작은 언어 모델이 에이전트의 미래라는 NVIDIA 연구진의 글이 이 형식입니다.

용어	정의
작은 언어 모델(SLM, Small Language Model)	일반 소비자용 장비에 올려 한 사용자의 에이전트 요청을 실용적인 지연 시간으로 처리할 수 있는 크기의 언어 모델입니다.
재조회 검증	쓰기를 마친 뒤 코드가 업무 시스템을 다시 읽어 값이 맞는지 대조하는 단계입니다. 모델의 「완료했습니다」라는 답과 실제 시스템 상태를 구분하고 감사 추적용 기록을 남깁니다.
점진적 공개(Progressive Disclosure)	에이전트가 처음에는 스킬의 이름과 설명만 보고, 관련 있을 때 SKILL.md 본문을, 실제로 필요할 때만 참고 파일을 여는 3단계 읽기 방식입니다.
지시문 해석	사용자가 보낸 요청 문장을 읽고 스킬 안의 여러 명령 가운데 어느 것을 부를지 고르는 단계입니다. 판별 규칙은 규칙표와 정규식 같은 코드로 두고, 모르면 정해진 질문을 내고 멈춥니다.
진행기	스킬 안에서 해석, 값 추출, 되묻기, 드라이런, 확인, 쓰기, 재조회를 정해진 순서로 실행하는 결정적 스크립트입니다. 상태 파일에 단계를 기록해 멈춘 곳에서 이어서 실행합니다.
질문 예시	업무 전문가가 정한 다양한 요청 문장과 기대 결과의 묶음입니다. 스킬을 만드는 재료이면서 회귀 시험으로 고정되어 운영 모델의 성능을 재는 시험 문항이 됩니다.
총소유비용(TCO, Total Cost of Ownership)	장비 구입비뿐 아니라 운영·전력·관리 비용까지 합친 도입 전체 비용입니다.
판단 모델	실행 모델과 별도로 두는 모델로, 코드가 만든 결과가 사용자의 요청을 충족했는지와 실패했다면 어떤 유형인지를 판정하는 일만 맡습니다.
평가자-개선자(Evaluator-Optimizer)	한 모델이 결과를 만들고 다른 모델이 평가와 피드백을 주는 워크플로 유형입니다.
폐쇄망	외부 인터넷과 분리된 조직 내부망입니다. MSAP.ai AAS 운영 환경은 폐쇄망 안에 두고 외부 모델 공급사로 가는 경로를 만들지 않습니다.
프로파일(Profile)	Hermes Agent에서 에이전트 하나가 쓰는 모델 설정, 스킬, 도구, 비밀값을 묶은 실행 단위입니다. 완성된 스킬은 운영 프로파일에 올라가 실행됩니다.
프론티어 AI	Claude처럼 현재 가장 높은 추론 성능을 내는 상용 대형 모델입니다. 개발 환경에서 업무 정의서를 읽고 규칙 파일·결정적 스크립트·회귀 시험을 만듭니다.

용어	정의
프리필과 디코드(Prefill and Decode)	LLM 추론의 두 단계입니다. 프리필은 입력 프롬프트 전체를 한 번에 처리하고, 디코드는 답을 토큰 하나씩 만들어 냅니다.
합격선	스킬을 운영에 들일지 정하는 기준입니다. MSAP.ai AAS는 실제 요청 하나에 도구 호출 3회 이하, 몇 분 안의 응답, 정답을 예시 기준으로 씁니다.
확인 게이트	업무 시스템에 쓰기 직전 사람의 승인을 받는 단계입니다. 작은 모델이 승인 없이 쓰거나 이미 승인된 작업을 다시 묻지 않도록 코드가 쥡니다.
활성 파라미터(Active Parameters)	MoE 모델이 토큰 하나를 만들 때 실제로 계산에 쓰는 파라미터입니다. 토큰마다 읽는 가중치 양이 줄어 같은 장비에서 생성 속도가 빨라집니다.
회귀 시험(Regression Test)	질문 예시마다 기대 결과를 붙여 두고 스킬을 고칠 때마다 다시 돌려 결과가 그대로인지 확인하는 시험 묶음입니다.
훅(Hook)	에이전트 실행 흐름의 정해진 시점에 끼어드는 코드입니다. 모델 호출 전 훅은 첫 줄로 진행기를 바로 실행하고, 도구 호출 전 훅은 스킬 수정을 막으며, 출력 직후 훅은 모델의 답을 진행기 원문으로 바꿉니다.
CRM(Customer Relationship Management)	고객사와 영업기회, 견적 같은 영업 정보를 관리하는 고객 관계 관리 시스템입니다.
CVE(Common Vulnerabilities and Exposures)	공개된 보안 취약점마다 붙이는 공통 식별 번호입니다.
Hermes Agent	Nous Research가 공개한 오픈소스 에이전트 실행 환경입니다. 스킬을 SKILL.md 디렉터리로 관리하며, MSAP.ai AAS가 이를 웹 콘솔과 MCP 서버로 다룹니다.
JSON-RPC 2.0	JSON 형식으로 원격 함수를 부르고 결과를 받는 메시지 규약입니다. MCP의 호스트·클라이언트·서버는 이 형식으로 통신합니다.
LiteLLM	여러 LLM 제공자를 OpenAI 호환 접속 주소 하나로 묶는 오픈소스 게이트웨이입니다. 운영 환경의 모든 모델 호출이 이곳을 거칩니다.
LLM(Large Language Model)	대량의 글로 학습해 문장을 이해하고 만들어 내는 대형 언어 모델입니다.
MCP(Model Context Protocol)	LLM 애플리케이션이 외부 자료와 도구를 같은 규약으로 부를 수 있게 만든 개방형 프로토콜입니다. MSAP.ai AAS는 MCP 서버를 맡아 프론티어 AI가 스킬을 만들고 올리고 켜는 일을 직접 하게 합니다.

용어	정의
MoE(Mixture of Experts, 전문가 혼합)	층마다 라우터가 여러 전문가 가운데 일부만 골라 계산하는 모델 구조입니다. 전체 파라미터보다 훨씬 적은 활성 파라미터만 계산에 씁니다.
MSAP.ai AAS(AI Agent Studio)	업무 전문가가 말로 정의한 업무를 프론티어 AI가 스킬로 만들고, 조직 안의 작은 모델이 그 스킬을 같은 결과로 실행하도록 만들기와 넘기기를 잇는 도구입니다.
SKILL.md	스킬 디렉터리의 중심 파일입니다. 앞머리 YAML 블록에 스킬 이름(name)과 설명(description)을 적고, 본문에는 실행 명령과 금지 사항을 짧게 적습니다.
vLLM	오픈소스 LLM 서빙 엔진으로, OpenAI 호환 서버를 띄워 로컬 LLM을 Chat Completions 형식과 도구 호출로 제공합니다.



AI Agent 스킬 설계

CONTACT

WEB
msap.ai
www.msap.ai/

EMAIL
hello@msap.ai

TEL
02-6953-5427
0269535427

YOUTUBE
@msaptv
www.youtube.com/@msaptv

LINKEDIN
linkedin.com/showcas...
www.linkedin.com/showcase/msap-ai/

FACEBOOK
facebook.com/opennaru
www.facebook.com/opennaru

SCAN

© 2026 SMART CF, MSAP.ai. All rights reserved.