

Protocolo de Ingesta de Contexto para Agentes de IA

Framework AGENTES-Y-SKILLS

Gemini Code Assist

21 de enero de 2026

Índice

1. Introducción	1
2. Secuencia de Suministro de Instrucciones	1
2.1. 1. Nivel Meta: El Protocolo Cognitivo (docs/manual-operativo.md)	1
2.2. 2. Nivel Contexto: El Mapa del Territorio (Agent.md)	2
2.3. 3. Nivel Funcional: Activación Bajo Demanda (skills/*.md)	2
3. Ejemplo Aplicado: Proyecto de Ciberseguridad	2
4. Manejo de Errores y Alucinaciones	3
4.1. Protocolo de Recuperación	3
5. Mantenimiento y Evolución del Contexto	3
6. Conclusión	3
7. Resumen del Flujo	3

1. Introducción

Para colaborar eficazmente con un Agente de Software en un proyecto nuevo (por ejemplo, de Ciberseguridad), no se debe volcar toda la información de golpe. Se debe seguir un orden jerárquico de suministro de instrucciones basado en los archivos `.md` del framework. Esto asegura que el agente entienda primero *cómo pensar* y luego *qué hacer*.

2. Secuencia de Suministro de Instrucciones

2.1. 1. Nivel Meta: El Protocolo Cognitivo (docs/manual-operativo.md)

Cuándo suministrar: Al inicio de la sesión o en el *System Prompt*.

Propósito: Configurar el comportamiento del modelo antes de que vea el código del proyecto.

- Define la restricción de tokens (brevedad).
- Establece la arquitectura de 'Lazy Loading' (no alucinar información no cargada).
- Instruye sobre el uso de la orquestación para tareas grandes.

Instrucción al Agente: 'Lee docs/manual-operativo.md para entender tus restricciones operativas y tu estándar de respuesta.'

2.2. 2. Nivel Contexto: El Mapa del Territorio (Agent.md)

Cuándo suministrar: Inmediatamente después del manual, como contexto principal del proyecto.
Propósito: Situar al agente en el dominio específico (Ciberseguridad en este caso).

- **Visión:** 'Este es un proyecto de pentesting/auditoría...'
- **Mapa:** Dónde están los scripts de nmap, los reportes, etc.
- **Índice de Skills:** Qué herramientas tiene disponibles (ej. `skill-vuln-scan`).

Instrucción al Agente: 'Analiza `Agent.md` para comprender la estructura del repositorio y tus objetivos actuales.'

2.3. 3. Nivel Funcional: Activación Bajo Demanda (skills/*.md)

Cuándo suministrar: Nunca de forma proactiva al inicio. Solo cuando el *Trigger* se active.
Propósito: Ahorrar ventana de contexto.

- El agente leerá `Agent.md`, verá que existe una skill llamada `network-audit`.
- Cuando tú le pidas: 'Escanea la red', el agente buscará y leerá `skills/network-audit/skill.md`.
- Este archivo contiene los *One-Shot prompts* (ejemplos) de cómo usar herramientas específicas de seguridad.

3. Ejemplo Aplicado: Proyecto de Ciberseguridad

Si inicias un proyecto de auditoría de seguridad, el flujo de creación de archivos sería:

1. **Copiar docs/manual-operativo.md:** Sin cambios, es la base lógica.
2. **Crear Agent.md Raíz:**

```
# Agent.md: Contexto Maestro - SecurityOps-Audit

## 1. Vision General
Plataforma de auditoria de seguridad automatizada.
Objetivo: Detectar CVEs en infraestructura critica.

## 2. Mapa del Repositorio
* /targets: Listas de IPs y dominios autorizados (Scope).
* /scans: Logs crudos de Nmap/OpenVAS.
* /skills:
    * skill-recon: Reconocimiento pasivo y activo.
    * skill-exploit-check: Validacion de vulnerabilidades.
    * skill-report: Generacion de informes ejecutivos.

## 3. Protocolo de Seguridad (ROE)
* STRICT: Solo operar sobre activos listados en /targets.
* LOGGING: Registrar todos los comandos ejecutados.
* MODE: Solo lectura/escaneo, no explotacion sin confirmacion.
```

3. **Definir Skills Específicas:** Crear `skills/skill-recon/skill.md` con los comandos exactos:

```
# Skill: Reconocimiento (Recon)

## Metadata
* Trigger: Activar al requerir 'escanear', 'enumerar' o 'mapear'.
* Scope: /scans, /targets
* Tools: nmap, subfinder

## Resources (One-Shot)
### Escaneo de Puertos
```

```
nmap -sS -sV -oN /scans/nmap_res.txt -iL /targets/scope.txt
### Subdominios
subfinder -dL /targets/domains.txt -o /scans/subs.txt
```

4. Configurar Skill de Reporte: Crear skills/skill-report/skill.md para consolidar hallazgos:

```
# Skill: Report Generation

## Metadata
* Trigger: 'generar_reporte', 'crear_informe', 'exportar_pdf'
* Scope: /scans, /reports
* Tools: pandoc, file-system

## Resources (One-Shot)
### Compilaci\'on de Hallazgos
cat /scans/*.txt > /reports/raw_findings.md
### Generaci\'on de PDF
pandoc /reports/final.md -o /reports/audit.pdf --toc
```

4. Manejo de Errores y Alucinaciones

El framework está diseñado para minimizar alucinaciones, pero si ocurren, el archivo `docs/manual-operativo.md` sirve como ancla de la verdad.

4.1. Protocolo de Recuperación

Si el agente inventa rutas o herramientas no listadas en `Agent.md`:

1. **Invocar Restricciones:** Recordar al agente la sección 'Restricción de Eficiencia' del manual.
2. **Verificación de Skills:** Preguntar: '¿Has cargado la skill necesaria para esta tarea? Revisa los Triggers en `Agent.md`.'
3. **Prompt de Corrección:** 'Detente. Tu respuesta contradice el `docs/manual-operativo.md`. Vuelve a leer `Agent.md` y limita tu respuesta a los recursos listados allí.'

5. Mantenimiento y Evolución del Contexto

El contexto de la IA se degrada si no se mantiene actualizado con los cambios del código.

- **Sincronización Periódica:** Al finalizar hitos importantes, instruye: 'Ejecuta `skill-sync` y actualiza el mapa en `Agent.md` con los nuevos directorios creados.'
- **Refactorización de Skills:** Si una skill crece demasiado (más de 500 líneas), pide al agente: 'Divide `skill-recon` en `skill-recon-active` y `skill-recon-passive`.'

6. Conclusión

Este protocolo transforma la interacción con la IA de un chat desestructurado a un flujo de ingeniería sistemático. Al respetar la jerarquía `Manual` → `Agent` → `Skill`, se garantiza consistencia, se minimizan alucinaciones y se permite la escalabilidad del proyecto.

7. Resumen del Flujo

```
Manual Operativo → Define CÓMO trabajar.
                    ↓
Agent.md → Define DÓNDE trabajar.
                    ↓
Skills (Lazy Load) → Define QUÉ herramientas usar (solo al necesitarlas).
```