Annexe gratuite tirée d'Agents en production

Checklist de conception des outils et serveurs MCP

Annexe gratuite tirée d'Agents en production : les contrôles qui rendent un outil facile à utiliser pour un modèle et difficile à détourner pour un attaquant

Écrit par des agents IA (Claude), sous la direction de Yann

Édition 2026 à 2027

B

Checklist de conception des outils et serveurs MCP

Les vérifications qui rendent un outil facile à utiliser pour un modèle et difficile à détourner pour un attaquant

Tech Solo No-code

Le lecteur d'un outil est un modèle : un nom vague, une trace de pile ou 40 000 *tokens* (unités de texte facturées par le modèle) de JSON l'embrouillent. Cette checklist transforme les chapitres 3, 4 et 11 en points de revue.

En bref : ce que vous allez obtenir

- Des vérifications de conception : noms, descriptions, paramètres, valeurs de retour, erreurs.
- Des vérifications de sécurité : effets de bord, authentification, chaîne d'approvisionnement.
- Des vérifications MCP pour la révision (2026-07-28, situation fin 2026).

Comment utiliser cette checklist

Utilisez-la en revue de conception et comme modèle de *pull request* (demande de fusion). Chaque point renvoie à du code, à un schéma ou à un résultat d'*eval* (évaluation automatisée).

1. Forme et granularité

- [ ] L'outil correspond à une tâche de l'agent (`schedule_event`), pas à un point d'accès brut (Anthropic).
- [ ] Les étapes qui s'exécutent toujours ensemble forment un seul outil.
- [ ] Quelques outils à fort impact, pas un par point d'accès.
- [ ] Il n'existe aucun quasi-doublon qu'on pourrait confondre.

2. Noms et descriptions

- [ ] Le nom est préfixé par le service ou la ressource (`asana_projects_search`).
- [ ] La description dit ce que fait l'outil et quand l'utiliser (ou non).
- [ ] La description montre un appel correct.
- [ ] Elle ne donne aucune instruction sur d'autres outils.

3. Paramètres

- [ ] Les noms de paramètres sont sans ambiguïté (`user_id`, pas `user`).
- [ ] Des énumérations remplacent le texte libre quand les valeurs sont connues.
- [ ] Les paramètres obligatoires et facultatifs sont explicites, avec leurs valeurs par défaut.
- [ ] Le schéma utilise JSON Schema 2020-12, la valeur par défaut depuis la spécification 2025-11-25.
- [ ] Le serveur valide toutes les entrées.

4. Valeurs de retour

- [ ] Les résultats portent des noms, pas des UUID nus.
- [ ] Les charges volumineuses proposent un `response_format` (concis ou détaillé).

- [ ] Les résultats sont paginés, filtrés ou tronqués, avec une taille maximale par défaut.
- [ ] `structuredContent` est fourni quand l'appelant analyse le résultat.
- [ ] Les résultats vides sont explicites (« 0 ligne ne correspond au filtre X »).

5. Erreurs

- [ ] Les erreurs disent comment corriger l'appel (paramètre, valeurs valides), pas une trace de pile.
- [ ] Les échecs de validation sont renvoyés comme erreurs d'exécution d'outil pour que le modèle se corrige (SEP-1303).
- [ ] Les erreurs HTTP de l'amont sont converties en erreurs d'outil ; une erreur 500 n'est jamais un succès.
- [ ] Les résultats distinguent les erreurs qu'on peut relancer de celles qu'on ne peut pas relancer.

6. Effets de bord et sécurité

- [ ] Les outils en lecture seule définissent `readOnlyHint: true` ; les autres sont des écritures, avec `destructiveHint`, `idempotentHint` et `openWorldHint` explicites (les valeurs par défaut supposent le pire).
- [ ] Le serveur applique lui-même les portées et bloque les écritures : les annotations sont des indications non fiables.
- [ ] Les outils d'écriture sont idempotents ou acceptent une clé d'idempotence.
- [ ] Les actions destructrices ou externes proposent une simulation ou une confirmation humaine.
- [ ] Les capacités dangereuses inutiles sont supprimées, pas seulement protégées.
- [ ] Des limites de débit et des quotas protègent les systèmes en aval contre les boucles.

7. Transport, authentification et état MCP

- [ ] Les serveurs distants utilisent Streamable HTTP sur HTTPS, pas le transport HTTP+SSE déprécié.
- [ ] Le serveur vérifie que chaque *token* d'accès a été émis pour lui (pas de transmission directe de *tokens* d'accès).
- [ ] Les portées sont minimales par outil, sans jokers ; l'élévation de droits utilise `WWW-Authenticate`.
- [ ] L'enregistrement utilise les Client ID Metadata Documents ; le Dynamic Client Registration déprécié ne reste que pour les anciens clients.
- [ ] Rien ne dépend de `Mcp-Session-Id` ni de `initialize` (tous deux supprimés) ; le serveur implémente `server/discover` et garde l'état dans des identifiants qu'il émet.
- [ ] Les nouveaux serveurs évitent Sampling, Roots et Logging, dépréciés.
- [ ] Les identifiants d'état sont liés à l'utilisateur : détenir un identifiant n'est pas une authentification.
- [ ] Les URL OAuth et sortantes sont validées (http(s) uniquement, plages privées bloquées, pas de shell).
- [ ] `tools/list` renvoie les outils dans un ordre déterministe pour aider la mise en cache des prompts.
- [ ] Un plan de migration daté sert la révision précédente jusqu'à ce que les principaux clients prennent en charge 2026-07-28 (prise en charge par Claude en cours de déploiement, fin 2026) et abandonne les fonctionnalités dépréciées sous 12 mois.

8. Exploitation

- [ ] Le serveur propage le contexte de trace OpenTelemetry depuis `_meta`.
- [ ] Chaque appel émet un *span* (segment d'une trace) : nom de l'outil, latence, taille du résultat, statut d'erreur.
- [ ] Les serveurs locaux en stdio tournent en bac à sable, avec des listes d'autorisation pour le système de fichiers et le réseau.

- [ ] Les secrets viennent d'un coffre ou d'un trousseau, jamais de valeurs en clair dans la configuration.
- [ ] Le serveur et ses dépendances sont figés ; les versions consignent les changements de description.

9. Evals

- [ ] Des tâches réalistes en plusieurs étapes sollicitent l'outil, avec des cas réservés (*holdout*, jeu de test réservé).
- [ ] Les evals suivent l'exactitude, les appels d'outils, les tokens et les erreurs ; quelqu'un lit les transcriptions.
- [ ] Un changement de description ou de schéma lance les evals de l'outil en CI (tests de bout en bout : chapitre 9).

Anti-pattern

Le serveur qui se contente d'envelopper un CRUD

Un serveur généré à partir d'une spécification OpenAPI de 120 points d'accès donne au modèle 120 outils : de mauvais choix, cinq appels là où un seul suffirait, un contexte saturé. Réduisez aux outils dont l'agent a besoin. « La date doit être au format ISO 8601 ; vous avez envoyé 28/11 » coûte moins cher qu'un changement de modèle.

Pour aller plus loin : le livre complet

Ceci était un extrait gratuit d'**Agents en production**, un guide pratique pour faire passer des agents du pilote à la production et les y maintenir. Le livre complet comprend un prologue, 21 chapitres en neuf parties, une postface sur la façon dont le livre a été fait et la checklist de mise en production, plus un cahier d'exercices compagnon et des compléments gratuits en ligne.

Le livre complet couvre :

- **Modèles mentaux :** ce qu'est vraiment un agent, et quand un simple *workflow* (enchaînement d'étapes prédéfinies dans le code) vaut mieux.

- **Briques de base :** des outils que les agents peuvent vraiment utiliser, MCP après la révision 2026-07-28, les *skills* (dossiers avec un `SKILL.md`, chargés à la demande), les sous-agents et les plugins, l'ingénierie du contexte et la mémoire.
- **Frameworks et architecture :** une carte indépendante des éditeurs du paysage des frameworks en 2026, un guide de décision et des architectures de référence pour les agents pilotés par les événements et durables.
- **Production :** des *harnesses* (le code qui encadre la boucle du modèle) de fiabilité, des *evals* (évaluations automatisées) en CI, l'observabilité et l'économie unitaire, la sécurité des agents et de MCP, la gouvernance et le règlement européen sur l'IA.
- **Business :** des recettes d'automatisation no-code, des guides pratiques pour indépendants qui font gagner ou économiser de l'argent, et un plan construire, acheter et recruter pour les managers.
- **L'usine logicielle agentique :** des agents qui mènent le logiciel de l'idée à la *pull request* (demande de fusion) fusionnée, le testent de bout en bout et continuent de l'améliorer, avec un projet de synthèse qui fait passer un système par chaque poste.
- **La suite :** un atlas des cas d'usage, les preuves sur la productivité, l'emploi, le coût et l'énergie, et des prévisions sourcées.
- **Annexe :** la checklist de mise en production (59 contrôles, dont 19 obligatoires). En ligne et gratuits : une checklist de conception des outils et serveurs MCP, un modèle de modélisation des menaces, les notes de terrain 2026, un glossaire et les consignes de chaque agent qui a écrit le livre. Le cahier d'exercices compagnon contient les exercices et leurs corrigés.

Quatre parcours de lecture (Tech, Manager, Solo, No-code) vous disent exactement quels chapitres lire et lesquels sauter. Les exercices, le projet fil rouge et les autoévaluations sont fournis avec leurs corrigés complets.

Écrit par des agents IA (Claude, un modèle conçu par Anthropic) sous la direction de Yann.

Disponible sur {{STORE_URL}}