Fiche technique

Albert API - Absence de rétention des données métier sur le chemin d'inférence

Mise à jour : 2/9/2026

1. Objet et périmètre

Albert API est le point d'accès interministériel mutualisé à des modèles d'intelligence artificielle, opéré par la DINUM sur infrastructures Outscale (SecNumCloud, tenant DINUM) et Nubo (DGFIP).

Le produit, présenté sous forme d'API, est formé de deux parties qui peuvent être utilisées indépendamment.

D'une part, **l'inférence**, c'est-à-dire les appels aux *endpoints* de génération de texte, de complétion, d'*embeddings*, de code et de transcription audio, quel que soit le modèle appelé et quelle que soit l'infrastructure qui le porte.

D'autre part, des **fonctions de RAG** (collections, dépôt de documents, *chunking*, base vectorielle, recherche documentaire). Ces fonctions reposent, par construction, sur la persistance des documents versés par l'administration utilisatrice : elles ne relèvent pas d'un régime de non-rétention et font l'objet d'une documentation distincte. Une administration qui n'utilise que les *endpoints* d'inférence n'active aucune de ces fonctions.

Le périmètre de la présente note est exclusivement l'inférence.

2. Définition retenue

Est qualifié de **donnée métier** tout contenu transmis par l'administration utilisatrice ou produit pour elle : texte des requêtes (*prompts*), texte des réponses générées (*complétions*), textes soumis à vectorisation, fichiers transmis en entrée.

Engagement : sur le périmètre défini au point 1, aucune donnée métier n'est écrite sur un support de stockage durable, **à aucune fin** : ni supervision, ni suivi des usages, ni *monitoring* technique, ni filtrage de contenu (*guardrails*), ni sécurité, ni amélioration ou entraînement de modèles ou du service.

La donnée métier n'existe qu'en mémoire des processus qui la traitent, pour la seule durée de traitement de la requête. Elle n'est ni journalisée, ni mise en cache persistant, ni indexée, ni tracée, ni exportée vers un tiers.

3. Ce qui est effectivement conservé : des métadonnées, sans contenu

La non-rétention porte sur le contenu, non sur l'existence des appels. Le service conserve un jeu restreint de métadonnées, nécessaires à l'application des quotas, au dimensionnement de la capacité et au suivi de l'exploitation. La liste ci-dessous est exhaustive.

Métadonnée conservée	**Finalité**	**Durée de conservation**
Horodatage de la requête	Exploitation, capacité	24 mois
Identifiant du compte / de la clé d'API appelante	Quotas, suivi d'usage par entité	24 mois
Nom du modèle et *endpoint* appelé	Capacité, pilotage de l'offre	24 mois
Nombre de *tokens* en entrée et en sortie	Quotas, capacité, facturation interne	24 mois
Code de retour HTTP, latence, durée de génération	Supervision, qualité de service	24 mois
Adresse IP source	Sécurité, limitation de débit	24 mois

Aucune de ces métadonnées ne permet, seule ou par recoupement, de reconstituer le contenu d'une requête ou d'une réponse.

4. Administration

Accès des administrateurs. L'exploitation s'effectue via un accès administrateur nominatif et tracé, transitant par un bastion et un tunnel chiffré, depuis des postes d'administration dédiés dits *Securix*, durcis par la cellule cybersécurité de la DINUM en collaboration avec l'ANSSI. Ces postes sont indispensables pour accéder aux infrastructures de production et d'administration sur Nubo et Outscale : aucun accès n'est possible depuis un poste bureautique ordinaire. Cet accès porte sur les machines et les composants d'exploitation ; il n'ouvre sur aucun magasin contenant des données métier d'inférence, puisqu'il n'en existe aucun.

Absence de tiers. Les modèles à poids ouverts sont hébergés et exécutés sur les infrastructures précitées. Aucun appel n'est émis vers un fournisseur de modèle externe : le contenu des requêtes ne quitte jamais le périmètre Outscale SecNumCloud / Nubo.

5. Conséquences opérationnelles

Ce choix d'architecture est structurant et emporte des conséquences que l'administration utilisatrice doit intégrer :

- **aucune restitution a posteriori** n'est possible — ni d'une conversation, ni d'un contenu produit : la DINUM ne peut pas fournir ce qu'elle ne conserve pas ;

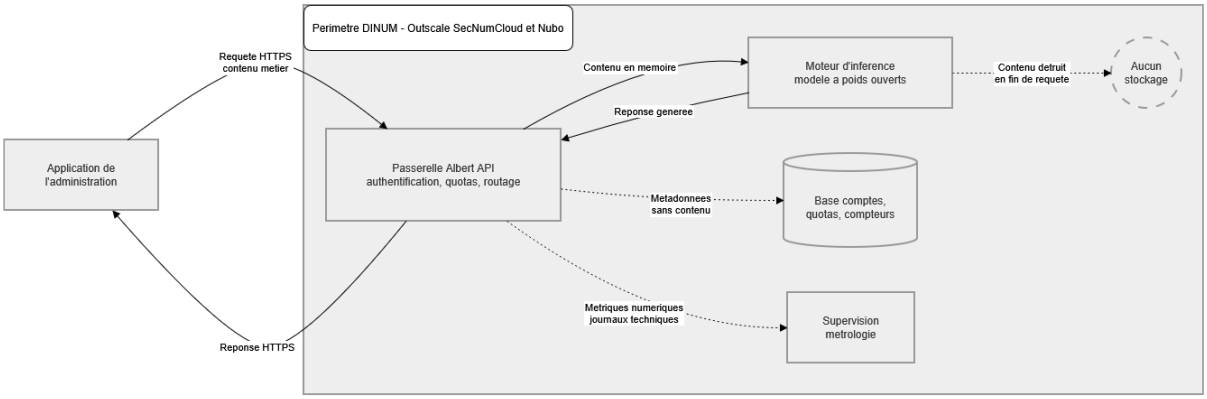
- **aucune analyse de contenu a posteriori** n'est possible en cas d'incident de sécurité ou de signalement d'usage abusif ; la maîtrise des usages, la modération et la journalisation applicative, si elles sont requises, relèvent de l'application appelante et de l'administration qui l'opère ;
- **la DINUM intervient en qualité de sous-traitant au sens de l'article 28 du RGPD.** L'administration utilisatrice demeure responsable de traitement : elle définit les finalités et les données transmises. La DINUM n'accède à aucune donnée à caractère personnel contenue dans les requêtes, ne les conserve pas et n'en fait aucun usage propre ;
- **aucune réutilisation** des contenus, en particulier à des fins d'entraînement, d'ajustement ou d'évaluation de modèles, n'est possible ni pratiquée.

6. Vérifiabilité

Les affirmations de cette note sont vérifiables par l'administration utilisatrice. La DINUM s'engage à fournir sur demande:

1. **le code source de la passerelle**, publié et auditable en permanence : https://github.com/eta-lab-ia/OpenGateLLM.
2. **les éléments du dossier de sécurité** relatifs à l'hébergement et à l'homologation : https://ia.numerique.gouv.fr/outils-ia/albert-api/infrastructure-securisee/.
3. **un atelier technique** avec l'équipe Albert API, pour instruire les questions résiduelles d'une analyse d'impact relative à la protection des données.

Annexe 1 : flux de données sur le chemin d'inférence pure



Les flux en trait plein transportent de la donnée métier ; ils sont éphémères. Les flux en pointillés transportent exclusivement des métadonnées et sont les seuls à alimenter un stockage durable.

Annexe 2 : traitement du contenu, composant par composant

Composant	Rôle	Voit le contenu	Écrit le contenu
Reverse proxy (nginx)	Terminaison TLS, routage	En transit	Non. Journalisation des corps de requête désactivée. Mise en tampon transitoire sur disque détruit à la fin de la requête
API Albert (OpenGateLLM / FastAPI)	Authentification, quotas, routage vers les modèles	Oui, en mémoire	Non
Base de données relationnelle (PostgreSQL)	Comptes, clés d'API, quotas, compteurs	Non	Non
Cache technique (Redis)	Sessions, compteurs de limitation de débit	Non	Non
Moteurs d'inférence (vLLM)	Exécution des modèles	Oui, en mémoire vive et mémoire GPU	Non. Journalisation des requêtes désactivée ; *KV-Cache* en mémoire GPU, non persisté, détruit à l'arrêt du processus (voir Annexe 3)
Base vectorielle (Elasticsearch)	RAG uniquement	Non sollicitée sur le périmètre inférence	Sans objet
Collecte et agrégation de journaux (Promtail, Loki)	Journaux applicatifs et système	Non. Les journaux collectés sont dépourvus de contenu métier	Non
Métrologie (Prometheus, Grafana, Node Exporter)	Séries temporelles numériques	Non	Non
Observabilité LLM (Langfuse) Sondes de disponibilité (Gatus)	Suivi des appels, explicitement configuré pour ignorer le contenu des requêtes	Non	Non

Tests synthétiques	Requêtes de test générées par le service, sans donnée d'utilisateur	Sans objet	
Restitution statistique (Metabase)	Tableaux de bord d'usage	Non. Agrégats de métadonnées uniquement	Non

Annexe 3 : Le cas particulier du *KV-Cache*

Mécanisme. Pour générer un texte, un modèle calcule pour chaque *token* du contexte des représentations intermédiaires (clés et valeurs) qu'il doit conserver le temps de la génération, faute de quoi il les recalculerait à chaque *token* produit. vLLM réserve à cet effet, au démarrage du processus, un espace fixe de mémoire GPU découpé en blocs. Ces blocs contiennent des tenseurs numériques, non le texte lui-même.

Durée de rétention. Les blocs alloués à une requête sont libérés dès la fin de celle-ci. Une optimisation standard, la mise en cache de préfixes (*prefix caching*), conserve toutefois dans l'espace GPU les blocs correspondant aux préfixes déjà calculés, afin de les réutiliser si une requête ultérieure présente **exactement** le même préfixe (cas typique : le *prompt* système d'une application, identique à chaque appel). Ces blocs sont évincés au fil de la charge selon une politique du moins récemment utilisé. Leur durée de vie n'est donc pas déterministe : elle est bornée par la pression sur la mémoire GPU et, en tout état de cause, par la durée de vie du processus. Un redémarrage du moteur détruit intégralement l'espace.

Propriétés de sécurité. Ces blocs résident exclusivement en mémoire GPU. Ils ne sont jamais écrits sur disque, jamais sauvegardés, jamais répliqués, jamais exportés. Ils ne sont ni lisibles ni adressables depuis l'API : aucun appel ne permet d'en obtenir le contenu. Une réutilisation ne se produit que si le préfixe soumis est strictement identique, c'est-à-dire si l'appelant possède déjà le contenu concerné.