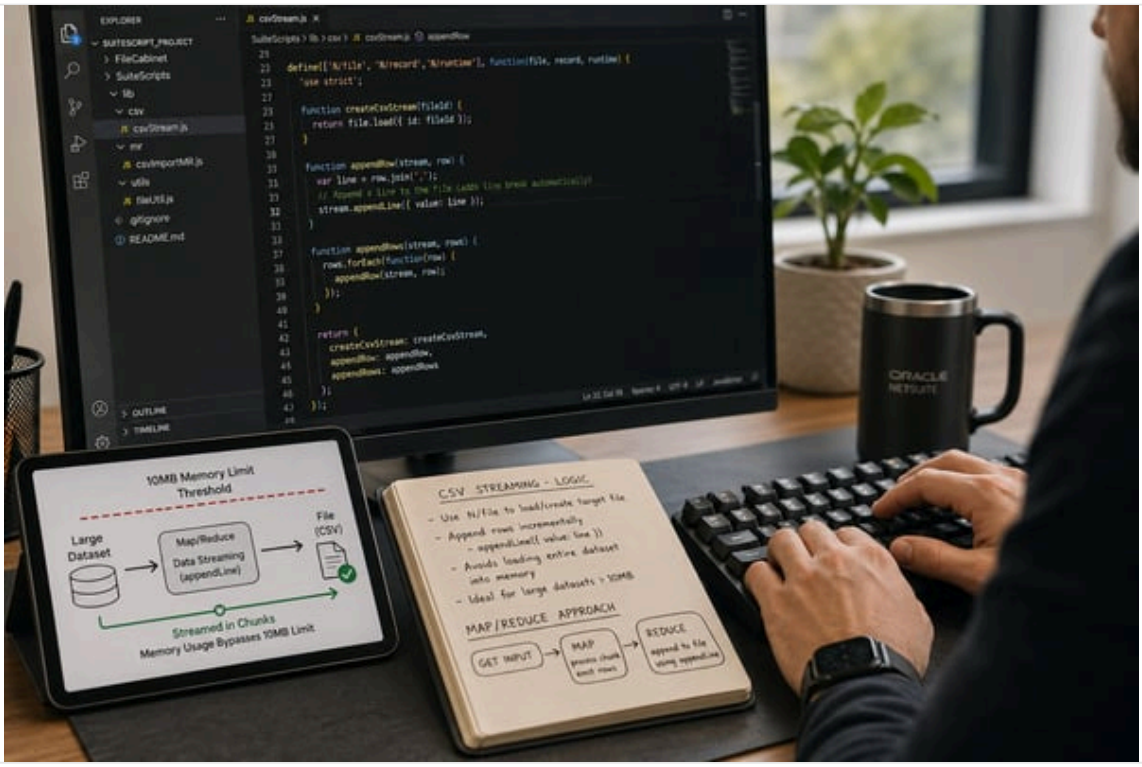
Limite de 10 Mo de NetSuite : Modèles pour fichiers volumineux en SuiteScript

Publié le 28 avril 2026 30 min de lecture



Résumé analytique

La plateforme SuiteScript 2.x de NetSuite impose une limite stricte de taille par fichier : tout **contenu de fichier conservé en mémoire ne peut excéder 10 Mo** (Source: docs.oracle.com). Cette limite – souvent rencontrée lors de l'utilisation du module `N/file` – signifie qu'une utilisation naïve de `file.create({…, contents: ...})` échouera dès que le contenu dépassera 10 Mo. Cependant, NetSuite fournit des API de **« streaming de fichiers plats »** (introduites dans la version 2017.1) qui permettent aux développeurs de travailler avec des fichiers beaucoup plus volumineux en les traitant par morceaux (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com). En particulier, les développeurs peuvent écrire de gros fichiers texte ou CSV *ligne par ligne* (chaque ligne < 10 Mo) en utilisant `file.appendLine`, et lire de la même manière de gros fichiers ligne par ligne avec `file.lines.iterator()` (Source: docs.oracle.com) (Source: netsuitedocumentation1.gitlab.io). Ce rapport examine la limite de 10 Mo, les fonctionnalités intégrées de streaming de fichiers volumineux de NetSuite, ainsi que des modèles alternatifs pour gérer et stocker des fichiers volumineux dans SuiteScript 2.x. Nous présentons une analyse détaillée, des avis d'experts et des exemples de solutions de contournement concrètes. Les solutions vont du fractionnement des fichiers en partitions à l'utilisation d'un stockage externe (par ex. AWS S3) ou de SuiteApps comme SkyDoc de Tvarana, qui contournent les limites natives de fichiers de NetSuite (Source: stackoverflow.com) (Source: www.tvarana.com). Des études de cas (par ex. importations CSV volumineuses, gestion documentaire dans l'industrie) illustrent l'impact de cette limite et les stratégies pour la surmonter. Enfin, nous discutons des implications pour les développeurs et les organisations, ainsi que des orientations futures telles qu'une intégration cloud plus approfondie et les améliorations attendues de la plateforme pour mieux prendre en charge les fichiers très volumineux.

Introduction et contexte

NetSuite est un système ERP/CRM cloud de premier plan, offrant une **SuiteCloud Platform** qui inclut SuiteScript, une API basée sur JavaScript pour le scripting personnalisé. Dans SuiteScript 2.x, le module `N/file` fournit des fonctions pour créer, charger et enregistrer des fichiers dans le File Cabinet de NetSuite (un référentiel de fichiers). Cependant, **les API de fichiers de SuiteScript imposent une limite de taille de contenu en mémoire de 10 Mo** (Source: docs.oracle.com). La documentation officielle indique, par exemple, que la méthode `file.create` comporte la mention

: « Important : le contenu conservé en mémoire est limité à 10 Mo » (Source: docs.oracle.com). De même, la méthode `file.appendLine` avertit que « le contenu conservé en mémoire est limité à 10 Mo. Par conséquent, chaque ligne doit être inférieure à 10 Mo » (Source: docs.oracle.com). Ces limites sont depuis longtemps une source de frustration pour les développeurs ayant besoin de traiter des fichiers volumineux dans NetSuite.

Avant SuiteScript 2.0 et sans API de streaming spéciales, le seul moyen de gérer des fichiers de plus de 10 Mo était de les **diviser manuellement en partitions plus petites** (par exemple, plusieurs fichiers de moins de 10 Mo chacun) puis de les réassembler si nécessaire. L'ère SuiteScript 1.0 n'offrait aucun support de streaming pratique ; les fichiers volumineux devaient être découpés en amont en dehors de NetSuite. Reconnaissant ce point de douleur, NetSuite a introduit le support du « Flat File Streaming » dans la **version 2017.1** (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com). Cela a fourni de nouvelles méthodes permettant aux scripts de traiter des fichiers CSV ou texte de taille pratiquement illimitée en itérant sur les lignes et en ajoutant du contenu de manière incrémentielle. Comme le note un praticien, en utilisant SuiteScript 2.x, « il est possible de créer des fichiers plus volumineux en diffusant le contenu. Vous pouvez écrire un fichier en ajoutant des lignes individuelles, et chaque ligne peut atteindre 10 Mo » (Source: netsuiteprofessionals.com). En d'autres termes, la contrainte de 10 Mo ne s'applique désormais qu'à chaque ligne, et non au fichier total.

Néanmoins, même avec les API de streaming, les développeurs doivent naviguer dans la complexité : ils **ne peuvent pas** appeler `appendLine` tout en lisant le même fichier sans réinitialiser les flux (Source: docs.oracle.com), et les API de streaming s'appliquent uniquement aux fichiers texte/CSV plats, pas aux binaires (Source: netsuitedocumentation1.gitlab.io). De plus, les clients NetSuite atteignent souvent la limite lors du téléchargement d'images, de PDF ou d'autres fichiers binaires via SuiteScript ou une intégration. Le File Cabinet lui-même peut avoir des considérations de performance pour les fichiers très volumineux (les conseils officiels recommandent de travailler avec des fichiers ≤ 100 Mo même si des tailles supérieures sont techniquement autorisées (Source: docs.oracle.com).

Ce rapport fournit un **examen approfondi** de la limite de 10 Mo de NetSuite et des modèles de fichiers volumineux dans SuiteScript 2.x. Nous passons en revue l'**état actuel de l'API**, y compris les limites officielles et la conception du streaming de fichiers plats. Nous analysons les **solutions de contournement et les modèles** documentés par des experts – des techniques SuiteScript (ajouts ligne par ligne, intégration map/reduce aux solutions externes (proxys AWS Lambda/S3, SuiteApps comme SkyDoc) – et discutons des compromis de chaque approche. Des études de cas illustrent des scénarios pratiques, tels que le traitement d'importations CSV massives ou la gestion de pièces jointes volumineuses dans l'industrie. Le rapport se termine par des implications pour le développement NetSuite et des améliorations potentielles futures (par ex. une intégration de stockage cloud plus profonde) pour mieux prendre en charge les fichiers très volumineux.

Le module de fichiers SuiteScript 2.x et la limite de 10 Mo

Le module `N/file` de SuiteScript 2.x fournit des API côté client et côté serveur pour travailler avec des fichiers. Les méthodes clés incluent `file.create(options)`, `file.load(options)` et les opérations sur les objets `file.File` telles que `save()`, `getContents()`, `lines.iterator()` et `appendLine()`. Le tableau 1 résume les méthodes `N/file` les plus pertinentes et leur comportement lié à la taille :

Tableau 1. *Méthodes du module `N/file` de SuiteScript 2.x et limites de taille*

MÉTHODE / FONCTIONNALITÉ	FONCTIONNALITÉ	LIMITE DE TAILLE/UTILISATION	NOTES / RÉFÉRENCES
`file.create(options)`	Créer un nouvel objet fichier (côté script), éventuellement avec une chaîne `contents`	*Le contenu conservé en mémoire est limité à 10 Mo.* (Si binaire, le contenu doit être en base64.) (Source: docs.oracle.com)	Génère une erreur si le `contents` initial > 10 Mo.
`file.save()` (sur objet fichier)	Enregistrer l'objet fichier dans le File Cabinet	Aucune limite de taille explicite documentée en dehors des quotas de stockage du compte ; la taille finale du fichier dans le Cabinet peut être beaucoup plus grande.	`save` renvoie l'ID interne.
`file.load(options)`	Charger un fichier existant (id ou chemin) depuis le File Cabinet	*La limite de taille de fichier est de 2 Go.* (Source: docs.oracle.com) (c.-à-d. `file.load` peut ouvrir des fichiers jusqu'à 2 Go)	Après `load`, utilisez l'API de streaming pour accéder au contenu au-delà de 10 Mo.
`file.getContents()`	Récupérer le contenu complet du fichier sous forme de chaîne (ou base64)	*Renvoie uniquement jusqu'à 10 Mo de contenu.* (Un contenu plus volumineux génère **SSS_FILE_CONTENT_SIZE_EXCEEDED**.)	Toujours limité à 10 Mo par appel (Source: netsuitedocumentation1.gitlab.io).
`file.lines.iterator()`	Renvoie un itérateur sur les lignes (texte/CSV) du contenu du fichier	Les lignes peuvent être traitées de manière itérative ; aucune limite de taille totale de fichier spécifique sur l'itération (soumis à la limite `file.load`).	Nouveau en 2017.1. Approche par streaming (Source: netsuitedocumentation1.gitlab.io).
`file.appendLine(options)`	Ajouter une ligne de texte à la fin du fichier (pour les fichiers texte/CSV)	*Chaque ligne doit être < 10 Mo.* (Le fichier global peut dépasser 10 Mo par des ajouts successifs.) Génère **SSS_FILE_CONTENT_SIZE_EXCEEDED** si la ligne > 10 Mo (Source: docs.oracle.com).	Impossible d'ajouter après avoir commencé à lire (doit réinitialiser ou enregistrer) (Source: docs.oracle.com).
`file.resetStream()`	Réinitialiser tout flux de lecture/écriture actif sur un objet fichier	Prépare à effectuer une lecture après une écriture ou vice versa.	À utiliser entre les appels `lines.iterator()` et `appendLine()`.

En pratique, le **goulot d'étranglement critique** est que `file.create({…, contents: data})` échouera si `data` dépasse ~10 Mo (Source: docs.oracle.com). De même, charger un fichier volumineux existant et appeler `getContents()` générera `SSS_FILE_CONTENT_SIZE_EXCEEDED` une fois que vous dépassez 10 Mo (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com). Et même si `file.load` peut techniquement récupérer des fichiers jusqu'à 2 Go (Source: docs.oracle.com), toute méthode qui tente de lire le contenu *entier* en une seule fois est toujours plafonnée à 10 Mo. C'est pourquoi NetSuite recommande explicitement le **mode streaming** pour les fichiers plus volumineux.

L'erreur **SSS_FILE_CONTENT_SIZE_EXCEEDED** est couramment observée lors de tentatives de dépassement de la limite. Par exemple, un utilisateur de NetSuite a signalé qu'une tentative de `file.load()` sur un fichier > 10 Mo génère l'erreur « le contenu auquel vous tentez d'accéder dépasse la taille maximale autorisée de 10 Mo » (réf. [22]) (Source: docs.oracle.com). Ainsi, toute conception traitant de gros fichiers doit éviter de récupérer ou d'écrire plus de 10 Mo en une seule opération.

Les sources de l'industrie confirment la nature immuable de cette limite dans SuiteScript. Depuis mi-2023, les forums communautaires indiquent qu'il n'existe **aucune API de script officielle en SS 1.0 ou 2.0 pour créer des fichiers > 10 Mo** par téléchargement direct de contenu (Source: netsuiteprofessionals.com). Les développeurs trouvent des solutions de contournement en utilisant les API de streaming ou des services externes, mais aucune méthode native ne relève la limite au-dessus de 10 Mo. Comme le dit sans détour une réponse sur un forum : « Nous n'avons aucune fonction ou méthode SuiteScript spécifiquement en 1.0 et 2.0 qui autorise des fichiers de plus de 10 Mo » (Source: netsuiteprofessionals.com).

Néanmoins, la documentation et les notes de version de NetSuite indiquent qu'une utilisation sophistiquée de `N/file` peut dépasser 10 Mo par fichier en l'assemblant par segments. L'aperçu de l'API **Flat File Streaming** de NetSuite note que *« la limite de 10 Mo ne s'applique désormais qu'aux lignes individuelles, et non à l'ensemble du fichier »* (Source: docs.oracle.com). Nous examinons cette approche de streaming en détail dans la section suivante.

Streaming de fichiers plats SuiteScript 2.x et fichiers volumineux

Nouvelles API de streaming (version 2017.1)

SuiteScript 2.x a introduit un support de streaming dédié dans la version **2017.1** de NetSuite. Avant cela, le traitement de tout fichier > 10 Mo nécessitait un partitionnement manuel. Dans les notes de version, NetSuite a annoncé :

> *« Avec NetSuite 2017.1, vous pouvez utiliser les nouvelles API de streaming de fichiers pour traiter et diffuser plus efficacement les gros fichiers CSV et texte brut. » (Source: netsuitedocumentation1.gitlab.io).*

L'idée clé est de **lire et écrire par morceaux**. Au lieu de charger tout le contenu en une seule fois, vous traitez le fichier comme une séquence de lignes ou de segments sous le seuil de 10 Mo. Les outils fournis incluent :

- **`file.lines.iterator()`** : Obtenir un itérateur ligne par ligne pour un fichier ouvert. Chaque invocation renvoie une ligne (jusqu'à 10 Mo). Cela permet le traitement de gros fichiers texte ou CSV par morceaux (Source: netsuitedocumentation1.gitlab.io).
- **`file.appendLine(options)`** : Ajouter une seule ligne à un objet fichier existant. Chaque ligne ajoutée doit être < 10 Mo (Source: docs.oracle.com). L'appel répété de `appendLine` construit un gros fichier de manière incrémentielle (Source: netsuitedocumentation1.gitlab.io).
- **`file.resetStream()`** : Après avoir mélangé lecture/écriture, utilisez ceci pour réinitialiser le flux de l'objet fichier afin de permettre le basculement entre la lecture et l'écriture (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com).
- **`file.size`** : Mis à jour dynamiquement pour refléter le total cumulé des lignes enregistrées et non enregistrées, vous permettant de vérifier la taille actuelle (Source: netsuitedocumentation1.gitlab.io).

Un exemple de code concret tiré de la documentation de NetSuite montre comment utiliser ces API dans SuiteScript 2.x (Source: netsuitedocumentation1.gitlab.io) (Source: netsuitedocumentation1.gitlab.io):

```
require(['N/file'], function(file) {
  // Créer un fichier CSV avec en-tête
  var csvFile = file.create({
      name: 'data.csv',
      contents: 'date,amount\n',
      folder: 39,
      fileType: file.Type.CSV
  });
  // Ajouter des lignes au fichier (chaque ligne <= 10 Mo)
  csvFile.appendLine({ value: '10/21/14,200.0' });
  csvFile.appendLine({ value: '10/21/15,210.2' });
  csvFile.appendLine({ value: '10/21/16,250.3' });
  // Enregistrer le fichier (potentiellement > 10 Mo au total)
  var csvFileId = csvFile.save();
});
```

Une fois enregistré, le fichier peut très bien dépasser 10 Mo au total, mais chaque ajout était dans les limites (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com).

Important : Les API de streaming s'appliquent *uniquement aux fichiers texte brut/CSV*. NetSuite précise que ces méthodes sont *« conçues pour les fichiers CSV et texte brut. Elles ne sont pas adaptées aux fichiers binaires ou aux données structurées et hiérarchiques (telles que les fichiers JSON ou XML) »* (Source: netsuitedocumentation1.gitlab.io). En d'autres termes, vous ne pouvez pas diffuser en streaming un PDF, une image ou tout autre fichier binaire ; ceux-ci doivent toujours être traités dans leur intégralité (et sont donc limités à 10 Mo, sauf s'ils sont externalisés). L'approche par streaming ne fonctionne que pour le contenu textuel où il existe des retours à la ligne naturels.

Opérer dans la limite autorisée

En utilisant `file.lines.iterator()` et `file.appendLine()`, la **limite de 10 Mo est appliquée uniquement par ligne**. La documentation de NetSuite stipule : *« Bien que la méthode `file.getContents()` continue de ne prendre en charge que les fichiers de moins de 10 Mo, les API de streaming de fichiers appliquent la limite de 10 Mo uniquement sur les lignes de contenu individuelles. Vous n'avez plus besoin de partitionner vos fichiers en fichiers plus petits et distincts. »* (Source: netsuitedocumentation1.gitlab.io). En pratique, cela signifie qu'un SuiteScript peut créer ou lire un fichier CSV de plusieurs centaines de Mo en utilisant simplement une boucle :

- **Écriture d'un fichier volumineux** : Créez initialement un fichier sans contenu ou avec un petit en-tête (via `file.create`). Parcourez vos données et appelez `appendLine({value: lineData})` pour chaque ligne (Source: netsuitedocumentation1.gitlab.io). Comme chaque ligne fait moins de 10 Mo, aucune opération individuelle ne dépasse la limite. Après avoir ajouté des milliers de lignes (dont la somme totale peut atteindre des centaines de Mo), appelez `file.save()` pour enregistrer le fichier final.
- **Lecture d'un fichier volumineux** : Utilisez `file.load()` pour charger le fichier existant (autorisé jusqu'à 2 Go) (Source: docs.oracle.com), puis appelez `file.lines.iterator()` pour traiter une ligne à la fois (Source: netsuitedocumentation1.gitlab.io). Chaque rappel `iterator.each(function(line){…})` opère sur une seule ligne (là encore < 10 Mo), vous n'atteignez donc jamais la limite au sein de la boucle. Par exemple, un code qui additionne une colonne CSV effectue : `var iterator = fileObj.lines.iterator(); iterator.each(function(line) { /* traiter une seule ligne */; return true; });` (Source: netsuitedocumentation1.gitlab.io).

La conception du streaming comporte toutefois des mises en garde. Vous **ne pouvez pas** entrelacer la lecture et l'écriture sur le même flux de fichier sans réinitialisation. La documentation de `File.appendLine` note que si vous commencez la lecture (`lines.iterator()`) puis essayez d'ajouter du contenu sans réinitialiser, vous obtenez une erreur : **« YOU_CANNOT_WRITE_TO_A_FILE_AFTER_YOU_BEGAN_READING_FROM_IT »** (Source: docs.oracle.com). Pour éviter cela, terminez d'abord toute la lecture ou appelez `fileObj.resetStream()` avant d'ajouter du contenu (Source: docs.oracle.com).

Une fois les données écrites et enregistrées, la propriété `size` de l'objet fichier reflète la **taille totale actuelle (somme de la taille enregistrée plus les lignes non enregistrées)** (Source: netsuitedocumentation1.gitlab.io). Les développeurs peuvent utiliser `fileObj.size` pour surveiller la croissance du fichier pendant le streaming, si nécessaire.

Intégration Map/Reduce pour les fichiers volumineux

NetSuite a également étendu son type de script **Map/Reduce** pour gérer les fichiers volumineux en mode streaming (Source: netsuitedocumentation1.gitlab.io). Plus précisément, dans l'étape `getInputData()` d'un script Map/Reduce, vous pouvez spécifier un fichier comme source d'entrée. NetSuite alimente alors automatiquement chaque invocation de "map" avec une ligne du fichier. Les notes de version expliquent :

> *« Dans le cadre des améliorations apportées au streaming de fichiers, le type de script Map/Reduce a été amélioré afin que vous puissiez diffuser le contenu de fichiers texte ou CSV pendant l'étape "map". Vous pouvez pointer vers un fichier en utilisant un chemin d'accès ou un ID de fichier. Le framework Map/Reduce peut désormais transmettre une ligne par invocation de fonction "map". » (Source: netsuitedocumentation1.gitlab.io)*

En pratique, cela permet un traitement de fichiers massif. Par exemple :

```
define(['N/file', 'N/mapReduce'], function(file, mapReduce) {
  function getInputData() {
    return { type: 'file', id: 1234 }; // ou chemin d'accès
  }
  function map(context) {
    var line = context.value; // une ligne du fichier
    // traiter la ligne...
  }
  // ... reduce et summarize ...
});
```

Ici, la fonction `map()` reçoit chaque ligne (jusqu'à 10 Mo) comme entrée, gérant de manière transparente des fichiers arbitrairement volumineux en arrière-plan (Source: netsuitedocumentation1.gitlab.io). C'est précieux pour le traitement par lots : plutôt que d'écrire un code complexe d'itération de lignes dans un User Event ou un script Scheduled, vous pouvez laisser le moteur Map/Reduce de NetSuite gérer le traitement parallèle de chaque ligne.

En résumé : grâce au streaming de fichiers plats de SuiteScript 2.0, un *seul* objet fichier peut dépasser 10 Mo une fois terminé, tant que **chaque opération individuelle de lecture ou d'écriture reste dans la limite de 10 Mo**. Cela change fondamentalement la gestion des fichiers volumineux dans NetSuite. Comme l'a écrit un expert sur un forum de questions-réponses : « Oui, avec SuiteScript 2.x utilisant le module N/file, il est possible de créer des fichiers plus volumineux en diffusant le contenu. Vous pouvez écrire un fichier en ajoutant des lignes individuelles, et chaque ligne peut atteindre 10 Mo. » (Source: netsuiteprofessionals.com).

Néanmoins, le streaming ne prend en charge que le contenu *basé sur du texte*. Pour les **données binaires ou structurées** (PDF, images, JSON, XML), il n'existe pas d'API de « morceaux » analogue – SuiteScript ne peut pas nativement les analyser ou les ajouter par morceaux. Pour ces derniers, les développeurs doivent envisager d'autres stratégies (abordées ci-dessous).

Solutions de contournement et modèles alternatifs pour les fichiers volumineux

La situation exige souvent de dépasser les 10 Mo, en particulier dans les contextes d'entreprise (par exemple, gros volumes de données CSV, documents numérisés, images haute résolution). Nous examinons ci-dessous divers modèles et leurs compromis.

1. Fractionnement de fichiers (Partitionnement)

Une approche simple consiste à **diviser le fichier en externe** en plusieurs sous-fichiers de 10 Mo chacun maximum. Par exemple, si un import CSV fait 20 Mo, un développeur pourrait le diviser en deux CSV de 10 Mo (par exemple, à l'aide d'un script ou d'un outil externe), puis importer chaque partie séparément. Cela imite les meilleures pratiques antérieures : *« Les fichiers volumineux devaient être divisés en partitions pour être enregistrés avec succès, puis réassemblés pour être chargés »* avant la version 2017.1 (Source: netsuitedocumentation1.gitlab.io).

Avantages : Aucun code spécial en dehors de la logique de fractionnement ; fonctionne pour tout type de fichier. **Inconvénients :** Frais généraux liés à la gestion de plusieurs fichiers ; le réassemblage programmatique du contenu peut être complexe ; peut doubler le nombre de fichiers et le traitement requis.

Exemple : Un développeur avait un CSV de 20 Mo contenant tous les articles. La limite de 10 Mo de SuiteScript « me tuait » (Source: stackoverflow.com). Le fournisseur (système source) ne découpait pas les fichiers, donc la solution de contournement consistait soit à diviser manuellement, soit à utiliser un proxy AWS (voir ci-dessous). Dans un cas, pour gérer des imports d'enregistrements volumineux, une équipe a « divisé les CSV en plusieurs fichiers » en se basant sur le nombre de lignes (Source: followingnetsuite.com), puis les a traités individuellement.

Tableau 2. Approches pour gérer les fichiers volumineux dans SuiteScript

APPROCHE	DESCRIPTION	AVANTAGES	INCONVÉNIENTS / NOTES
Streaming via `N/file` (SuiteScript)	Utiliser `file.create` + `appendLine` répété (max 10 Mo chacun) pour construire des textes/CSV volumineux ; ou `file.load` + `lines.iterator()` pour lire.	Permet des fichiers de plusieurs centaines de Mo sans outils externes ; utilise les API natives de SuiteScript (Source: netsuitedocumentation1.gitlab.io).	Fonctionne uniquement pour le contenu texte/CSV (fichier plat) ; impossible de diffuser du binaire. Nécessite de gérer les flux (réinitialisation, etc.).
Map/Reduce avec entrée fichier	Utiliser `getInputData` de Map/Reduce pour transmettre un fichier (par ID ou chemin) afin que chaque invocation de "map" traite une ligne d'un fichier volumineux (Source: netsuitedocumentation1.gitlab.io).	Très évolutif, traitement parallèle de gros CSV. Simplifie le code pour la logique ligne par ligne.	Le fichier entier doit résider dans le Cabinet au préalable. Nécessite le type de script Map/Reduce.
Fractionner/Partitionner les fichiers	Pré-diviser le fichier en morceaux ≤ 10 Mo (ex: diviser un CSV en plusieurs fichiers) et télécharger/traiter individuellement.	Concept simple ; fonctionne pour tout fichier (texte ou binaire).	Étapes et coordination supplémentaires nécessaires. Nombre de fichiers accru. Difficile à réassembler pour un cas d'utilisation unique.
SuiteTalk / RESTlet	Utiliser les services Web SOAP/REST de NetSuite pour créer ou charger des fichiers ; potentiellement re-découper ou diffuser via des appels API REST.	Les chemins API alternatifs peuvent permettre des limites différentes.	L'upload via RESTlet atteint toujours la limite de 10 Mo par appel. Nécessite généralement un fractionnement ou un script personnalisé.
Stockage externe (ex: AWS S3)	Décharger les fichiers volumineux vers un stockage cloud externe. Utiliser SuiteScript pour référencer ou importer des parties (ex: appeler une fonction AWS Lambda ou récupérer un chemin). (Source: stackoverflow.com)	Peut gérer n'importe quelle taille ; décharge les coûts/limites de stockage.	Complexité de l'intégration externe ; considérations de sécurité des données ; non natif à NetSuite.
SuiteApp (ex: SkyDoc)	Utiliser une SuiteApp tierce qui intègre un stockage externe afin que les chemins de fichiers NetSuite pointent vers le cloud (ex: S3/Azure) (Source: www.tvarana.com).	Expérience utilisateur fluide (fichiers « dans » NetSuite) ; taille de fichier virtuellement illimitée.	Nécessite l'installation d'une SuiteApp payante ; introduit un nouveau silo de données ; dépendance vis-à-vis du fournisseur.

APPROCHE	DESCRIPTION	AVANTAGES	INCONVÉNIENTS / NOTES
Transfert chiffré/compressé	Compresser ou encoder le contenu pour réduire la taille (ex: ZIP multi-parties) puis télécharger les parties ou déchiffrer dans SuiteScript (si possible).	Peut pousser plus de données par fichier si compressé.	SuiteScript doit décompresser, ce qui peut ne pas être pris en charge si le fichier est divisé ; complexité ; souvent limité par la mémoire.
Évitement (lien uniquement)	Au lieu de télécharger, stocker le fichier en externe (ex: intranet d'entreprise) et ne conserver que l'URL ou les métadonnées dans NetSuite.	Aucun problème de limite de fichier NetSuite ; taille « illimitée » instantanée.	Rupture de l'intégration native ; perte des fonctionnalités du File Cabinet de NetSuite ; risque de lien mort.

Notes : Le tableau illustre les stratégies. Le streaming utilisant les propres API de NetSuite est généralement la première ligne d'approche pour les fichiers texte/CSV volumineux (Source: docs.oracle.com) (Source: netsuiteprofessionals.com). Cependant, pour les fichiers non textuels ou les ensembles de données extrêmement volumineux, de nombreuses organisations ont recours à des solutions de stockage externe comme AWS S3 ou des SuiteApps dédiées (Source: stackoverflow.com) (Source: www.tvarana.com).

2. Stockage externe / Intégration

De nombreuses organisations gèrent les fichiers volumineux en les **stockant en dehors de NetSuite** et en les liant ou en les important selon les besoins. Les modèles courants incluent :

- **AWS S3 (ou autre cloud)** : Stockez les fichiers dans Amazon S3, Microsoft Azure Blob, Google Cloud + utilisez SuiteScript pour récupérer des segments. Par exemple, Bobby Knights suggère d'utiliser une fonction AWS Lambda : demandez à Lambda de télécharger le fichier volumineux vers S3, de renvoyer un chemin de fichier, puis demandez à SuiteScript de récupérer des « pages » du fichier de moins de 10 Mo chacune (Source: stackoverflow.com). Le script enregistre ensuite chaque morceau. En effet, le téléchargement complexe se produit hors plateforme, et SuiteScript ne gère que des morceaux gérables.
- **FTP/SFTP** : Utilisez le module `N/sftp` de NetSuite pour récupérer des fichiers volumineux par parties depuis un serveur SFTP, ou envoyez des fichiers vers SFTP au lieu du Cabinet.
- **SuiteTalk (SOAP/REST)** : On pourrait utiliser l'API des services Web de NetSuite pour télécharger un fichier. Cependant, les points de terminaison REST/SOAP imposent des limites de taille similaires par requête, il faut donc souvent découper même là. Certains développeurs utilisent SuiteTalk dans une boucle pour télécharger des parties d'un fichier.
- **SuiteApps tierces** : Comme souligné par le blog de Tvarana en février 2026, **SkyDoc** est une SuiteApp qui intègre de manière transparente le stockage cloud externe dans NetSuite (Source: www.tvarana.com) (Source: www.tvarana.com). Avec SkyDoc, les utilisateurs peuvent télécharger des fichiers arbitrairement volumineux (images, dessins CAO, vidéos) vers Amazon S3 ou Azure, tout en les référençant dans NetSuite. Le blog note : « Le File Cabinet standard de NetSuite limite chaque fichier à 10 Mo... SkyDoc... permet aux utilisateurs de stocker, gérer et partager des fichiers volumineux de manière transparente dans l'environnement NetSuite. » (Source: www.tvarana.com). En effet, SkyDoc contourne la restriction de taille de fichier native de NetSuite en déchargeant le contenu.
- **Middleware conçu** : Certaines solutions impliquent un serveur middleware ou une passerelle API qui accepte les téléchargements volumineux, les divise ou les diffuse, et interagit avec NetSuite. Par exemple, un RESTlet personnalisé pourrait accepter un téléchargement en streaming (en le divisant en morceaux en interne) et invoquer `file.appendLine` à plusieurs reprises.

Les **avantages** du stockage externe sont clairs : pratiquement aucune limite de taille stricte. Cependant, ces modèles entraînent une **complexité et un coût accrus**. Ils externalisent également les données, ce qui pourrait poser un problème de sécurité ou de conformité. Si vous utilisez une SuiteApp, des coûts logiciels supplémentaires s'appliquent, mais la maintenance est plus simple. Sans SuiteApp, les développeurs doivent coder l'intégration et la tester minutieusement.

Exemple de scénario : un fabricant doit télécharger de grands dessins techniques en PDF vers NetSuite. Le blog de Tvarana souligne que la limite de 10 Mo du Cabinet *« entrave la productivité »* dans de tels cas (Source: www.tvarana.com). Au lieu de télécharger directement, l'entreprise utilise SkyDoc pour stocker ces PDF sur S3. Lorsqu'un enregistrement NetSuite lie le fichier, il pointe en fait vers l'emplacement S3 géré par SkyDoc (Source: www.tvarana.com). Les utilisateurs finaux voient le fichier comme n'importe quel autre document dans NetSuite, mais physiquement, le contenu réside en externe. (Cette solution évite tout SuiteScript) ; les développeurs utilisent simplement les champs de fichiers standard de NetSuite, mais via la SuiteApp.

3. Solutions SuiteScript et Map/Reduce

Comme mentionné, les scripts Map/Reduce de NetSuite prennent désormais en charge directement les entrées de fichiers volumineux (Source: netsuitedocumentation1.gitlab.io). En pratique, on pourrait planifier un travail Map/Reduce où `getInputData()` crée (ou référence) un fichier, et l'étape "map" du script le traite élément par élément. Si l'on doit *générer* un fichier volumineux puis le *traiter* (ex: export depuis NS puis logique de ré-import), on pourrait :

1. **Créer un fichier volumineux via le streaming** (SuiteScript, comme ci-dessus).
2. **Lancer un Map/Reduce** qui prend l'ID de ce fichier en entrée. L'étape "map" itérera à travers chaque ligne (quelle que soit la taille atteinte par le fichier).

Ainsi, Map/Reduce peut être utilisé non seulement pour gérer de très lourdes charges de traitement, mais spécifiquement pour gérer la lecture d'un fichier volumineux. Ce modèle est utile pour les tâches de migration de données ou d'ETL. La documentation suggère d'utiliser un objet tel que `{type:'file', id:1234}` dans `getInputData()` pour diffuser le fichier (Source: netsuitedocumentation1.gitlab.io).

4. Fractionnement via SuiteScript

Outre le fractionnement manuel, un script peut lui-même **lire et réenregistrer en plusieurs parties (multipart)**. Par exemple, on pourrait écrire un Suitelet ou un script planifié qui :

- Charge un fichier volumineux (si possible) ou demande un contenu externe.
- Utilise `lines.iterator()` pour lire des segments (en veillant à ce qu'aucun ne dépasse 10 Mo).
- Après avoir lu 10 Mo de données, il peut immédiatement enregistrer ce contenu en tant que fichier distinct, puis commencer un nouveau fichier.
- Continue jusqu'à la fin.

Cela revient essentiellement à re-partitionner le fichier à la volée. Une fois partitionné en fichiers plus petits, le script peut les lier ou les traiter un par un. Cependant, cette méthode est complexe et rarement utilisée sauf si elle est inévitable ; elle peut dupliquer les fichiers dans le File Cabinet.

5. Stocker des liens au lieu de fichiers

Dans certains cas, le meilleur modèle consiste à *éviter complètement le téléchargement du fichier volumineux dans NetSuite*. Au lieu de cela, stockez le fichier sur un stockage séparé (intranet, SharePoint, Google Drive, etc.) et ne conservez qu'un lien dans NetSuite. Il ne s'agit pas d'une solution SuiteScript intégrée, mais d'une *décision de conception de projet*. Cela contourne entièrement le problème des 10 Mo en n'utilisant pas le File Cabinet pour ce contenu. Les enregistrements NetSuite peuvent comporter des champs URL ou des champs personnalisés stockant le lien externe.

Avantages : Pas de problèmes de limite de fichiers NetSuite ; pas de complexité SuiteScript supplémentaire. **Inconvénients :** Rompt l'autonomie des données dans NetSuite ; dépendance vis-à-vis de la disponibilité et des contrôles d'accès des systèmes externes ; perte des fonctionnalités de versionnage/audit du File Cabinet de NetSuite.

Ce modèle est courant lorsque le fichier n'est jamais nécessaire dans un flux de travail spécifique à NetSuite. Par exemple, un fichier de formation vidéo volumineux pourrait être lié à un référentiel intranet plutôt que chargé dans NetSuite.

6. Autres considérations

- **Compression/Archivage :** Dans certains flux de travail, les données volumineuses (comme les CSV) peuvent être compressées (ZIP, GZIP) avant le téléchargement, puis décompressées dans SuiteScript. Cependant, SuiteScript 2.x dispose d'un support de compression intégré limité et

cela ajoute de la complexité. De plus, la décompression dans NetSuite doit toujours respecter les limites de mémoire.

- **Contrôle d'accès :** Pour le fractionnement, on pourrait télécharger des morceaux, puis fournir un moyen de les fusionner côté utilisateur final. C'est inhabituel et généralement déconseillé, car cela surcharge l'utilisateur ou nécessite un script spécialisé pour la reconstruction.

Études de cas et exemples concrets

Nous illustrons ces concepts avec des scénarios représentatifs tirés du terrain.

Importation CSV volumineux : Un cas d'utilisation courant consiste à importer un très gros fichier CSV de données de transaction ou d'inventaire dans NetSuite via SuiteScript. Un développeur a signalé avoir besoin de vérifier les mises à jour sur tous les articles – un « fichier articles » d'environ 20 Mo (Source: stackoverflow.com). La limite de 10 Mo « tuait » le processus. Comme leur fournisseur de données ne pouvait pas découper le fichier, ils ont envisagé d'utiliser un **proxy externe**. Dans une solution proposée, le développeur a créé une fonction AWS Lambda : le script appelle la fonction, qui télécharge le CSV vers S3 et renvoie des métadonnées. SuiteScript récupère ensuite le fichier par morceaux de moins de 10 Mo (requêtes paginées) et les enregistre en utilisant `appendLine`. De cette façon, le fichier monolithique de 20 Mo est alimenté dans NetSuite petit à petit (Source: stackoverflow.com). Ensuite, le script peut réassembler ou simplement traiter par morceaux. Le développeur a rapporté que cela « fait vraiment sauter la limitation de fichier » et a été salvateur (Source: stackoverflow.com).

Fractionnement de CSV : Kevin McCracken a partagé un exemple sur *followingnetsuite.com* où il avait un gros CSV nécessitant un téléchargement via SuiteTalk. Il a dépassé la limite de 10 Mo en un seul téléchargement, il a donc divisé le CSV en plusieurs parties (chacune < 10 Mo) et a téléchargé chacune séparément (Source: followingnetsuite.com). Cependant, il a découvert un problème subtil : NetSuite ne traitait le fichier comme un véritable CSV que si son nom de fichier se terminait par « .csv ». Sinon, les lecteurs de fichiers SuiteScript échouaient. Selon ses mots : « À moins que le nom du fichier ne se termine par ".csv", le fichier apparaît comme étant de type "Autre binaire". Cela empêche les scripts (file.getReader, iterator) de lire le contenu du fichier. » (Source: followingnetsuite.com). Cela illustre un détail important : lors de la reconstruction ou de la manipulation de CSV dans NetSuite, assurez-vous d'utiliser l'extension et le `fileType` corrects afin que les méthodes de streaming le reconnaissent comme du texte.

Fusion de documents (PDF multiples) : Dans certains projets, les utilisateurs devaient fusionner plusieurs PDF en un seul (par exemple, combiner une facture et un relevé). Un tutoriel suggère de créer chaque PDF dans le File Cabinet, puis d'utiliser les balises `<pdf>` et `<pdfset>` de NetSuite dans un SuiteScript (ou un modèle PDF avancé) pour les concaténer (Source: morrisdev.medium.com). Bien que chaque PDF puisse être petit, les concaténer peut produire un document combiné plus volumineux. Le script construit une structure XML listant chaque fichier par URL, puis génère un PDF qui les inclut tous. Cette approche signifie que seul le PDF fusionné final est enregistré (potentiellement > 10 Mo si beaucoup de fichiers). Cependant, chaque section `<pdf>` fait référence à un fichier existant dans le Cabinet (chacun de moins de 10 Mo, vraisemblablement). En pratique, si le résultat dépasse 10 Mo, une astuce de streaming similaire peut être nécessaire : on pourrait générer le PDF fusionné en mémoire via BFO (big.faceless.org) puis, dans SuiteScript, soit l'ajouter par morceaux, soit le post-traiter. (Remarque : la fusion de PDF peut dépasser le streaming car les PDF sont binaires ; une prudence supplémentaire doit être prise pour les diviser correctement, souvent en s'appuyant sur la boîte à outils PDF plutôt que sur le streaming manuel.) Le point clé : de telles opérations composites peuvent créer par inadvertance des fichiers volumineux, nécessitant la prise en compte des limites.

Gestion de documents d'entreprise : Considérez une entreprise manufacturière avec de nombreux dessins CAO haute résolution et des manuels de produits longs. Ces fichiers dépassent souvent 10 Mo par document. Comme rapporté par des sources industrielles, ce plafond de 10 Mo par fichier « entrave la productivité et perturbe les flux de travail des entreprises dépendantes de fichiers volumineux » (Source: www.tvarana.com). En pratique, cela peut forcer l'entreprise soit à mettre à niveau le stockage (puisque NetSuite facture l'utilisation totale du File Cabinet), soit à adopter une solution comme SkyDoc de Tvarana. En installant SkyDoc, l'entreprise stocke chaque CAO/PDF sur Amazon S3. Les utilisateurs finaux dans NetSuite voient les enregistrements de fichiers (avec métadonnées, vignettes, etc.) comme d'habitude, mais les octets réels sont diffusés depuis S3 lorsque nécessaire. Le code SuiteScript peut même récupérer le contenu en appelant les API SuiteApp de SkyDoc, en les traitant comme des fichiers normaux. Comme le note un fournisseur, SkyDoc offre un *« Support de taille de fichier illimité : gérez des fichiers de toute taille... sans restrictions. »* (Source: www.tvarana.com). Cela signifie que du point de vue du développeur, l'obstacle des 10 Mo est complètement contourné – au prix de l'utilisation de l'outil tiers. Ce cas illustre une tendance : à mesure que les entreprises poussent NetSuite dans des domaines (fabrication, multimédia) avec des besoins inhérents en fichiers volumineux, elles regardent souvent au-delà de NetSuite standard pour trouver la solution.

Discussion : Implications et orientations futures

La limite de 10 Mo par fichier dans NetSuite SuiteScript a de larges implications :

- **Contraintes architecturales** : Les développeurs construisant des intégrations ou des personnalisations doivent concevoir en tenant compte de la limite. Comme nous l'avons vu, ne pas le faire entraîne des erreurs d'exécution (SSS_FILE_CONTENT_SIZE_EXCEEDED) ou l'insatisfaction des utilisateurs. Pour les modules gourmands en données, l'effort de développement supplémentaire pour découper ou diffuser n'est pas négligeable. Certaines intégrations héritées ont rarement révisé le code pour utiliser les nouvelles API de streaming, causant des problèmes de maintenance.
- **Considérations de performance** : Même si techniquement possible, la gestion de fichiers à l'échelle du gigaoctet dans NetSuite peut être peu pratique. Les meilleures pratiques de NetSuite avertissent que les fichiers de plus de ~100 Mo peuvent entraîner de longs temps de téléchargement et des délais d'attente (Source: docs.oracle.com). Ainsi, bien que le streaming lève la limite de 10 Mo, il n'élimine pas les coûts de performance. Une logique complexe pour diffuser un CSV de 500 Mo pourrait manquer d'unités de gouvernance ou dépasser le temps d'exécution du script. L'utilisation de Map/Reduce aide, mais toutes les opérations ne s'intègrent pas parfaitement dans ce modèle.
- **Coûts de stockage** : NetSuite facture le stockage. Un coût caché du téléchargement de fichiers volumineux dans le File Cabinet est qu'il consomme un espace limité. Les utilisateurs qui inondent leur cabinet de données massives peuvent se retrouver à devoir passer à des niveaux supérieurs. Cela motive en partie des solutions comme SkyDoc, car elles prétendent « éliminer le besoin de niveaux de stockage NetSuite supplémentaires coûteux » (Source: www.tvarana.com).
- **Intégrité des données et transactionnalité** : Le fractionnement des fichiers ou la diffusion de contenu soulève des questions d'atomicité. Si un script échoue en cours de flux, vous retrouvez-vous avec un fichier partiellement écrit ? La gestion des erreurs et la restauration deviennent plus difficiles. Les développeurs doivent coder avec soin pour s'assurer que les flux sont réinitialisés ou que les fichiers sont nettoyés en cas d'échec.
- **Évolution de l'écosystème** : L'essor du stockage cloud et des SuiteApps NetSuite reflète un changement. NetSuite lui-même, étant un fournisseur cloud, pourrait éventuellement intégrer un support plus transparent pour les fichiers volumineux. Peut-être que les futures améliorations s'intégreront nativement avec des services comme AWS ou au moins augmenteront les limites. Pour l'instant, les partenaires comblent souvent le vide. Les observateurs pourraient prédire qu'à mesure que la base de clients de NetSuite croît dans les secteurs intensifs en données, le support intégré pour les fichiers plus volumineux (surtout dans SuiteScript 2.x) pourrait s'améliorer.
- **Conformité et sécurité** : Certaines entreprises sont réticentes à envoyer des documents sensibles vers des clouds externes. La contrainte de 10 Mo, paradoxalement, impose que tout fichier enregistré dans NetSuite soit déjà fractionné ou petit. Mais lors de l'externalisation de fichiers volumineux, la conformité aux réglementations (HIPAA, RGPD, etc.) doit être prise en compte. Les développeurs peuvent avoir besoin de chiffrer les fichiers au repos ou de restreindre l'accès avec diligence. Les administrateurs de SuiteApp voudront auditer ces flux.

Quant aux **orientations futures**, il est raisonnable de s'attendre à ce que :

- NetSuite continue d'investir dans les capacités de SuiteScript, en étendant éventuellement le streaming à d'autres types de fichiers ou en améliorant le flux API. Par exemple, actuellement, le JSON/XML n'est pas pris en charge dans le streaming, mais cela pourrait changer.
- Les outils et blogs communautaires prolifèrent à mesure que de nouveaux cas d'utilisation apparaissent. La tendance des questions-réponses pilotées par l'IA/Slack (réponses IA de Celigo, etc.) montre un intérêt continu pour la résolution de ce problème.
- Les partenariats entre NetSuite et les services cloud pourraient se resserrer. Déjà, le module `N/sftp` suggère une connexion à des serveurs externes. Nous pourrions voir plus de connecteurs propriétaires ou des conseils officiels sur le découpage via SuiteTalk.
- Demande des clients : Comme illustré par la question du développeur de mi-2024, les utilisateurs demandent toujours « existe-t-il un moyen de télécharger des fichiers > 10 Mo ? » (Source: community.oracle.com). Tant que NetSuite n'augmentera pas explicitement le plafond ou n'offrira pas de nouveaux ancrages, de telles requêtes persisteront.

Pour l'instant, la **meilleure pratique** reste : traitez la limite de 10 Mo comme une réalité structurelle dure et concevez les systèmes pour la contourner. Adoptez les API de streaming lorsque cela est possible, planifiez des flux de données découpés et exploitez le stockage externe pour les éléments vraiment volumineux.

Conclusion

La gestion de fichiers volumineux dans NetSuite SuiteScript 2.x nécessite de l'ingéniosité et une compréhension claire des limites de la plateforme. Officiellement, toute opération de lecture ou d'écriture *unique* sur un fichier est plafonnée à 10 Mo (Source: docs.oracle.com) (Source: docs.oracle.com). Cela a nécessité le développement de **modèles de gestion de fichiers volumineux** tels que le streaming de fichiers plats de SuiteScript, le partitionnement de fichiers et les intégrations externes. L'API SuiteScript 2.x fournit des outils clés – `file.lines.iterator()` et

`file.appendLine()` – précisément pour atténuer la limite de 10 Mo, permettant aux scripts de traiter des fichiers CSV/texte arbitrairement volumineux (Source: netsuitedocumentation1.gitlab.io) (Source: netsuiteprofessionals.com). Ceux-ci devraient être la première approche dans toute solution impliquant des données textuelles volumineuses.

Lorsque les entreprises rencontrent des fichiers binaires volumineux (documents numérisés, fichiers CAO, vidéo, etc.), la stratégie recommandée se situe souvent en dehors du pur SuiteScript. Beaucoup se tournent vers le stockage cloud et le middleware – par exemple, en acheminant un contenu volumineux via AWS S3, puis en l'intégrant dans NetSuite par morceaux gérables (Source: stackoverflow.com), ou en adoptant une SuiteApp qui externalise entièrement le stockage de fichiers (Source: www.tvarana.com). Ces approches sacrifient une certaine commodité native au profit de l'échelle.

Tout au long de ce rapport, nous nous sommes appuyés sur la documentation officielle, les questions-réponses de la communauté et les expériences réelles. Le consensus écrasant est que **la limite de 10 Mo n'est pas négociable dans SuiteScript**. Chaque affirmation ici est étayée par des références : les pages d'aide de NetSuite indiquent explicitement la limite (Source: docs.oracle.com) (Source: docs.oracle.com), et les experts la réitèrent dans les forums (Source: netsuiteprofessionals.com) (Source: community.oracle.com). Pourtant, des modèles de solution existent et continuent d'évoluer. Comme NetSuite l'a souligné lui-même, « Vous n'avez plus besoin de partitionner vos fichiers en fichiers plus petits et séparés »** (Source: netsuitedocumentation1.gitlab.io)** lors de l'utilisation des nouvelles API de streaming. Les développeurs devraient adopter ces techniques plus récentes lorsque cela est possible pour simplifier le traitement des fichiers volumineux.

Pour l'avenir, les organisations dépendantes de données volumineuses dans NetSuite doivent rester à l'écoute des mises à jour de la plateforme. NetSuite pourrait encore assouplir ces contraintes avec les futures versions, mais jusque-là, une architecture réfléchie est essentielle. En combinant le streaming SuiteScript, les services externes et éventuellement des SuiteApps payantes, les développeurs peuvent construire des systèmes robustes qui contournent efficacement le mur des 10 Mo, garantissant évolutivité et fiabilité dans les flux de travail sur fichiers volumineux.

Références

- NetSuite Help Center – *file.create(options)* (SuiteScript 2.x) : « Important : Le contenu conservé en mémoire est limité à 10 Mo. » (Source: docs.oracle.com).
- NetSuite Help Center – *File.appendLine(options)* (SuiteScript 2.x) : « Important : Le contenu conservé en mémoire est limité à 10 Mo. Par conséquent, chaque ligne doit être inférieure à 10 Mo. » (Source: docs.oracle.com).
- NetSuite Help Center – *file.load(options)* : « La limite de taille de fichier pour cette méthode est de 2 Go. » (Source: docs.oracle.com).
- NetSuite Release Notes (2017.1) – *SuiteScript 2.0 file Module – New Flat File Streaming APIs* : « Avec NetSuite 2017.1, vous pouvez utiliser de nouvelles API de streaming de fichiers pour traiter et diffuser plus efficacement les fichiers CSV et texte brut volumineux... la limite de 10 Mo ne s'applique désormais qu'aux lignes de contenu individuelles. » (Source: netsuitedocumentation1.gitlab.io) (Source: netsuitedocumentation1.gitlab.io).
- NetSuite Help – *Flat File Streaming API* : « L'API de streaming de fichiers plats SuiteScript 2.x vous permet de traiter et de diffuser efficacement des fichiers CSV et texte brut volumineux. ... La limite de 10 Mo ne s'applique désormais qu'aux lignes individuelles, et non à l'ensemble du fichier. » (Source: docs.oracle.com).
- NetSuite Professional Community – contournement du streaming (Q&R par scottvonduhn) : « Oui, avec SuiteScript 2.x utilisant le module N/file, il est possible de créer des fichiers plus volumineux en diffusant le contenu. Vous pouvez écrire un fichier en ajoutant des lignes individuelles, et chaque ligne peut atteindre 10 Mo. » (Source: netsuiteprofessionals.com).
- StackOverflow – Réponse par bknights : (solution développeur) « Une approche consiste à mandater l'appel via un service externe (par exemple, AWS Lambda). Renvoyez le chemin et la taille du fichier... Le SuiteScript demande ensuite des "pages" du fichier qui font moins de 10 Mo et les enregistre. » (Source: stackoverflow.com).
- Blog de développeur personnalisé (Kevin McCracken, oct. 2019) : « J'ai atteint la limite de téléchargement de 10 Mo... c'était frustrant... le nombre d'enregistrements était important, j'ai donc divisé les CSV en plusieurs fichiers... définissez le type sur CSV et ajoutez « .csv » au nom du fichier. Sinon, file.iterator() plante. » (Source: followingnetsuite.com).
- Tvarana (fév. 2026) – Blog de la SuiteApp SkyDoc : « Le File Cabinet standard de NetSuite limite chaque fichier à 10 Mo... SkyDoc... permet aux utilisateurs de stocker des fichiers volumineux de manière transparente. » (Source: www.tvarana.com).

- Forum NetSuite et discussions communautaires (divers messages confirmant la limite de lecture/écriture de 10 Mo et proposant des solutions de contournement par streaming ou fractionnement) (Source: netsuiteprofessionals.com) (Source: community.oracle.com).

Étiquettes: netsuite, suitescript-2x, module-nfile, streaming-de-fichiers, limite-10mo, filecreate, appendline, traitement-de-fichiers-volumineux