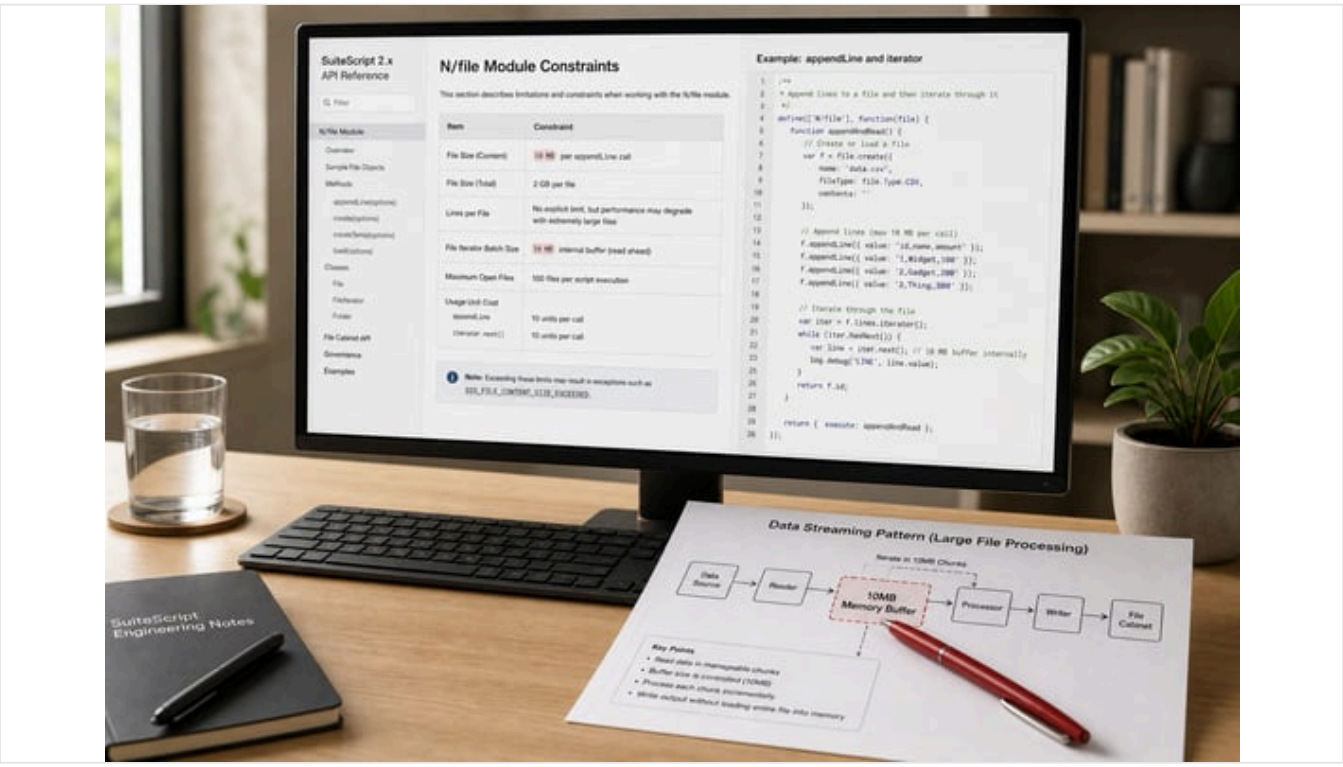
Limite de fichier de 10 Mo dans NetSuite SuiteScript : Causes et solutions

Publié le 31 mai 2026 38 min de lecture



Résumé analytique

La plateforme SuiteScript de NetSuite impose une **limite stricte de 10 Mo par fichier** pour le contenu chargé en mémoire. Toutes les API de fichiers de SuiteScript 2.x qui chargent ou créent du contenu doivent respecter cette règle (Source: docs.oracle.com) (Source: docs.oracle.com). En pratique, tenter d'utiliser `file.create` ou `file.load(GET)` sur un contenu dépassant 10 Mo entraîne une erreur `SSS_FILE_CONTENT_SIZE_EXCEEDED` (Source: docs.oracle.com) (Source: docs.oracle.com). Ce plafond a historiquement frustré les développeurs, car de nombreux cas d'utilisation en entreprise impliquent des fichiers CSV, des rapports ou des documents dépassant 10 Mo (Source: betanews.com) (Source: www.houseblend.io). Notamment, avant SuiteScript 2.x, il n'existait *aucun* support pour le streaming ; SuiteScript 1.0 exigeait le fractionnement manuel de tout fichier > 10 Mo (Source: docs.oracle.com).

Depuis NetSuite 2017.1, les **API de streaming de fichiers plats** atténuent cette limite en opérant ligne par ligne. Par exemple, `file.appendLine` permet de construire un fichier texte/CSV une ligne (≤ 10 Mo) à la fois (Source: docs.oracle.com) (Source: docs.oracle.com), et `file.lines.iterator()` peut lire un fichier chargé ligne par ligne, chaque ligne devant faire moins de 10 Mo (Source: docs.oracle.com). En utilisant ces méthodes de streaming, les scripts peuvent créer ou traiter des fichiers bien plus volumineux que 10 Mo au total, car la limite de 10 Mo ne s'applique qu'à chaque ligne ajoutée ou parcourue (Source: docs.oracle.com) (Source: docs.oracle.com). Néanmoins, le streaming ne fonctionne que pour le **contenu texte/CSV**, et non pour les fichiers binaires (PDF, images) qui doivent toujours être traités par blocs de ≤ 10 Mo. En conséquence, les développeurs emploient diverses solutions de contournement et modèles : fractionnement externe des fichiers d'entrée, utilisation de scripts Map/Reduce pour traiter les données en parallèle, transfert des fichiers vers un stockage externe (ex. AWS S3) et importation par morceaux, ou utilisation de SuiteApps spécialisées (ex. SkyDoc) qui contournent la limite native en stockant les données dans le cloud (Source: www.houseblend.io) (Source: archive.netsuiteprofessionals.com).

Ce rapport examine en profondeur les **causes** de la limite de 10 Mo de NetSuite, répertorie les API officielles et leurs contraintes (Tableau 1), et recense toutes les **solutions de contournement et modèles** connus (Tableau 2). Nous analysons ces approches en détail, en citant la documentation de NetSuite, les discussions sur les forums et des études de cas réelles. Par exemple, un développeur ERP a déploré qu'une importation CSV de 20 Mo « le tuait », et a dû *fractionner* le fichier ou utiliser un proxy AWS pour le paginer (Source: www.houseblend.io). À l'inverse,

un fabricant utilisant la SuiteApp SkyDoc de Tvarana a signalé avoir migré plus de 90 000 fichiers pour éliminer totalement la limite de taille de NetSuite (Source: www.79consulting.com). Nous discutons des compromis, tels que la performance et la complexité, de chaque stratégie. Enfin, nous décrivons l'état actuel et les perspectives d'avenir : NetSuite n'a pas relevé le plafond de 10 Mo, donc des intégrations plus poussées (stockage cloud, iPaaS et des fonctionnalités de streaming améliorées sont les voies attendues pour gérer des fichiers de données toujours plus volumineux sur la plateforme.

Introduction et contexte

NetSuite est une plateforme ERP/CRM cloud de premier plan offrant une gestion financière, opérationnelle et de la relation client au sein d'un système unique (Source: zapier.com). Elle est largement utilisée dans tous les secteurs pour les processus métier et la consolidation des données. La plateforme SuiteCloud de NetSuite fournit des API de personnalisation, notamment **SuiteScript** (un moteur de script basé sur JavaScript) pour automatiser et étendre les fonctionnalités. Dans SuiteScript 2.x, le module `N/file` permet aux scripts (user-events, suitescripts, RESTlets, Map/Reduce, etc.) de créer, charger et enregistrer des fichiers dans le **File Cabinet** de NetSuite (un référentiel de fichiers cloud) (Source: docs.oracle.com) (Source: docs.oracle.com). Par exemple, un script peut utiliser `file.create(options)` pour construire un nouveau fichier, puis `file.save()` pour le télécharger dans un dossier du Cabinet (Source: docs.oracle.com) (Source: docs.oracle.com).

Il est crucial de noter que **tout contenu de fichier dans SuiteScript 2.x est traité en mémoire**. La documentation officielle avertit à plusieurs reprises que *« le contenu conservé en mémoire est limité à 10 Mo »* pour toute API manipulant le contenu d'un fichier (Source: docs.oracle.com) (Source: docs.oracle.com). En d'autres termes, les scripts SuiteScript ne sont pas autorisés à charger ou traiter plus de 10 mégaoctets de données de fichier à la fois. Cette limite est appliquée au niveau de l'API. Par exemple, l'appel de `file.getContents()` sur un fichier chargé ne retournera jamais plus de 10 Mo avant de générer une erreur `SSS_FILE_CONTENT_SIZE_EXCEEDED` (Source: docs.oracle.com) (Source: docs.oracle.com). De même, `file.create` générera une erreur si son paramètre `contents` dépasse 10 Mo (Source: docs.oracle.com). Le tableau 1 ci-dessous résume les méthodes de fichier SuiteScript pertinentes et leur comportement lié à la taille.

Cette limite basée sur la mémoire remonte à l'introduction de SuiteScript 1.0. Comme le note la documentation de NetSuite, « avec SuiteScript 1.0, il est difficile d'accéder à des fichiers de plus de 10 Mo » et les développeurs « doivent diviser les gros fichiers en plus petits » (Source: docs.oracle.com). NetSuite a consolidé cette contrainte dans SuiteScript 2.x. Comme l'observe sans détour un post de SuiteAnswers, *« nous n'avons aucune fonction ou méthode SuiteScript... qui permette des fichiers de plus de 10 Mo »* (Source: www.houseblend.io). En substance, essayer de créer ou de lire plus de 10 Mo à la fois n'est tout simplement pas pris en charge par l'API native SuiteScript. Cette restriction reflète la conception de la plateforme : comme le contenu des fichiers réside dans la mémoire du script, des tailles de fichiers illimitées pourraient menacer les performances et la consommation de ressources de l'environnement cloud mutualisé de NetSuite. NetSuite n'a fourni aucun paramètre pour augmenter cette limite. Au lieu de cela, leur stratégie a été d'introduire des API de streaming (à partir de 2017) et d'encourager les architectes à gérer les données volumineuses par morceaux ou de manière externe (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com).

Malgré cela, les besoins des entreprises nécessitent souvent de travailler avec des fichiers bien plus volumineux que 10 Mo. La taille moyenne des fichiers dans les environnements d'entreprise augmente rapidement. L'analyse par Egnyte de 14 Po de données professionnelles a révélé qu'en 2017, le **fichier moyen faisait déjà ~3,13 Mo**, soit une hausse de 20 % par rapport à 2015 (Source: betanews.com). Par secteur, le contenu des médias/divertissements atteignait en moyenne 7,7 Mo par fichier, et même des secteurs plus petits comme la vente au détail atteignaient 4,4 Mo (Source: betanews.com). Le stockage des médias riches explose : les entreprises ont augmenté leur stockage de 55 % d'une année sur l'autre, avec une hausse de 43 % pour la vidéo et de 29 % pour les images (Source: betanews.com). En d'autres termes, il n'est **pas rare** de nos jours qu'un seul PDF, dessin CAO ou exportation CSV par lots dépasse 10 Mo.

Cette croissance de la taille des données a un impact direct sur les utilisateurs de NetSuite. Par exemple, un fabricant avec des dessins de produits haute résolution ou un détaillant générant des exportations CSV massives d'articles peut facilement dépasser 10 Mo. Dans un cas réel, un développeur NetSuite s'est plaint qu'un CSV d'environ 20 Mo « le tuait » jusqu'à ce qu'il se résolve à le diviser ou à le décharger sur AWS pour une récupération par morceaux (Source: www.houseblend.io). À l'inverse, certaines entreprises ont adopté des solutions externes : Stairlock (un fabricant d'escaliers) a déployé la SuiteApp SkyDoc pour stocker des documents sur S3, signalant qu'ils ont *« éliminé les limitations de taille de fichier avec un stockage évolutif »* après avoir migré plus de 90 000 fichiers depuis NetSuite (Source: www.79consulting.com). Ces exemples illustrent la pression exercée par la croissance de la taille des fichiers sur les limites natives de NetSuite.

En résumé, la limite de 10 Mo de SuiteScript est une contrainte de plateforme rigide et intégrée (Source: docs.oracle.com) (Source: docs.oracle.com). Elle est documentée dans chaque API de fichier qui touche au contenu. Pourtant, les clients ont fréquemment besoin de gérer des fichiers beaucoup plus volumineux. Reconnaissant cette lacune, NetSuite a introduit le support du « streaming de fichiers plats » en 2017.1 (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com) pour permettre aux développeurs de traiter des fichiers texte/CSV volumineux *ligne par ligne*. Néanmoins, les fichiers binaires (images/PDF) et les importations volumineuses nécessitent souvent des approches alternatives. Le reste de ce

rapport examine les détails techniques, les solutions de contournement et les modèles de conception pour travailler en toute sécurité avec des fichiers volumineux dans SuiteScript – des API de streaming aux intégrations externes – ainsi que leurs compromis, étayés par la documentation et des sources expertes.

La limite de taille de fichier de 10 Mo de SuiteScript

Le module `N/file` de SuiteScript 2.x de NetSuite fournit l'API principale pour créer et manipuler des fichiers. Les méthodes clés incluent `file.create(options)`, `file.save()`, `file.load(options)`, `file.getContents()`, `file.appendLine()`, `file.lines.iterator()` et `file.resetStream()`. Chacune joue un rôle dans la gestion des données de fichiers, mais elles sont toutes régies par un plafond de taille en mémoire.

Documentation officielle des limites

L'aide officielle de NetSuite est explicite sur la limite de 10 Mo. Par exemple, la documentation de la méthode **file.create** contient un avertissement sévère :

> ***Important :*** *Le contenu conservé en mémoire est limité à 10 Mo. (Documentation NetSuite file.create) (Source: docs.oracle.com).*

De même, la méthode **file.appendLine** est documentée avec « Important : Le contenu conservé en mémoire est limité à 10 Mo. Par conséquent, chaque ligne doit faire moins de 10 Mo. » (Source: docs.oracle.com). Pour être complet, `file.lines.iterator()` – introduit dans SuiteScript 2.x – comporte également la même note indiquant que chaque ligne doit être inférieure à 10 Mo (Source: docs.oracle.com). Cela limite effectivement toute chaîne de contenu ou ligne unique à 10 Mo. Le tableau 1 ci-dessous répertorie ces méthodes et leurs limites ou comportements pertinents, basés sur la documentation de NetSuite :

Tableau 1. Méthodes `N/file` de SuiteScript 2.x et contraintes de taille. (Toutes les tailles de contenu sont des limites en mémoire.)

MÉTHODE	FONCTIONNALITÉ	LIMITE DE TAILLE / COMPORTEMENT	NOTES / RÉFÉRENCES
`file.create(options)`	Créer un nouvel objet fichier dans le script	**Contenu initial ≤ 10 Mo** (en mémoire). Génère SSS_FILE_CONTENT_SIZE_EXCEEDED en cas de dépassement.	La doc officielle note : « Le contenu conservé en mémoire est limité à 10 Mo. » (Source: docs.oracle.com).
`file.save()` (sur objet fichier)	Télécharger (enregistrer) le fichier dans le File Cabinet	Aucune limite de taille explicite (diffuse tout le fichier vers le Cabinet). Limité uniquement par les quotas du File Cabinet.	La doc note : « File.save() diffuse des fichiers de toute taille » (jusqu'aux limites du cabinet) (Source: docs.oracle.com).
`file.load(options)`	Charger un fichier existant depuis le Cabinet	Prend en charge les fichiers jusqu'à 2 Go (doc NetSuite file.load).	File.load lui-même peut gérer des fichiers volumineux (≤ 2 Go), mais la récupération de contenu au-delà de 10 Mo nécessite un streaming (Source: docs.oracle.com) (Source: docs.oracle.com).
`file.getContents()`	Retourner le contenu complet du fichier sous forme de chaîne	Retourne ≤ 10 Mo de contenu ; génère SSS_FILE_CONTENT_SIZE_EXCEEDED sur des fichiers plus volumineux.	La doc avertit : « Le contenu conservé en mémoire est limité à 10 Mo. » (Source: docs.oracle.com). Aucun accès incrémentiel.
`file.lines.iterator()`	Diffuser le contenu du fichier ligne par ligne (pour texte/CSV)	Chaque ligne < 10 Mo ; aucune limite totale pour le fichier.	Nouveau en 2017.1. « Le contenu conservé en mémoire est limité à 10 Mo. Par conséquent, chaque ligne doit faire moins de 10 Mo. » (Source: docs.oracle.com).
`file.appendLine(options)`	Ajouter une seule ligne de texte à un objet fichier	Chaque ligne ajoutée < 10 Mo ; peut être appelée à plusieurs reprises pour accumuler un gros fichier.	Doc : « Le contenu conservé en mémoire est limité à 10 Mo. Par conséquent, chaque ligne doit faire moins de 10 Mo. » (Source: docs.oracle.com).
`file.resetStream()`	Réinitialiser le flux de lecture/écriture actuel sur un objet fichier	N/A	Permet de basculer entre la lecture et l'écriture de flux sur le même objet `file.File`. Doit être appelé entre les lignes/itérateur et appendLine si les deux sont utilisés. Aucune note sur la taille.

Les lignes sont tirées des pages d'aide officielles d'Oracle (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com). Le point essentiel est que **tout tampon de contenu ou ligne unique en mémoire est limité à 10 Mo** ; cependant, le *fichier entier stocké dans le File Cabinet peut être beaucoup plus volumineux*. En effet, une fois qu'un

objet fichier (construit à partir de segments ≤ 10 Mo) est enregistré, sa taille totale dans le File Cabinet peut dépasser 10 Mo. Par exemple, la documentation de NetSuite note que bien que le contenu soit diffusé en segments de ≤ 10 Mo, le fichier final enregistré dans le File Cabinet n'est soumis qu'aux limites de stockage globales (souvent jusqu'à plusieurs Go) (Source: docs.oracle.com) (Source: docs.oracle.com). Ainsi, la règle des 10 Mo est spécifiquement une limite de script en mémoire, et non un plafond pour le fichier persistant total.

La limite de 10 Mo en pratique

Lorsqu'un script tente de violer la limite de 10 Mo, NetSuite l'applique par des erreurs. Comme documenté, le code d'erreur **SSS_FILE_CONTENT_SIZE_EXCEEDED** est renvoyé avec le message *« the file content you are attempting to access exceeds the maximum allowed size of 10MB »* si un script tente d'exécuter `getContents()` sur un fichier de plus de 10 Mo (Source: docs.oracle.com). De même, fournir une chaîne `contents` > 10 Mo à `file.create()` échouera immédiatement. Étant donné que le moteur SuiteScript doit conserver les données du fichier en mémoire, il n'autorisera pas les opérations qui chargeraient plus d'octets que ce seuil.

Cette limite se manifeste dans les flux de travail courants. Par exemple, bien que `file.load({id: X})` puisse ouvrir des fichiers jusqu'à 2 Go (Source: docs.oracle.com), l'appel à `file.load(...).getContents()` échouera toujours au-delà de 10 Mo (Source: docs.oracle.com). Par conséquent, une routine comme :

```
let f = file.load({id: someId});
let text = f.getContents(); // échoue si le fichier > 10 Mo
```

tronquera le résultat ou lèvera une exception. Pour gérer un fichier volumineux, il faut plutôt itérer avec `file.lines.iterator()`, en le traitant ligne par ligne (chaque ligne < 10 Mo) (Source: docs.oracle.com).

Le plafond de 10 Mo est cohérent dans tous les contextes SuiteScript. L'interface utilisateur **File Drag-and-Drop** de NetSuite (téléchargement de fichiers via le navigateur) impose également une limite de 10 Mo par fichier (Source: docs.oracle.com). Ainsi, même au niveau de l'interface utilisateur, le système s'attend à des fichiers assez petits. (Curieusement, les importations CSV en masse autorisent jusqu'à 50 Mo par fichier (Source: stackoverflow.com), mais cela est géré par un moteur d'importation distinct, et non par SuiteScript.) Quoi qu'il en soit, le résultat est le suivant : **le code SuiteScript ne peut pas créer ou gérer un tampon de plus de 10 Mo à la fois.** Ceci est immuable dans les versions actuelles de NetSuite, et de nombreux fils de discussion sur les forums soulignent qu'aucune API native ne lève cette limite (Source: www.houseblend.io) (Source: archive.netsuiteprofessionals.com).

Cette limitation a des implications pratiques importantes. De nombreuses entreprises génèrent régulièrement des fichiers (rapports, exportations, journaux) bien plus volumineux que 10 Mo. L'étude sectorielle d'Egnyte a révélé que la taille des fichiers professionnels augmente : la taille moyenne des fichiers a atteint environ 3,13 Mo en 2017, et certains secteurs (médias, hôtellerie, etc.) atteignent déjà une moyenne de 5 à 8 Mo chacun (Source: betanews.com). Les images haute résolution, les fichiers CAO, les PDF ou les grandes exportations CSV dépassent souvent largement les 10 Mo. Par exemple, un développeur NetSuite a tenté de générer un CSV de *toutes les commandes client* (plus de 20 Mo) ; la tentative était un échec cuisant sous le plafond de 10 Mo, jusqu'à ce qu'il ait recours à des solutions de découpage (Source: www.houseblend.io). Une autre entreprise découvrant des fichiers journaux quotidiens de 20 Mo a dû soit les diviser, soit faire appel à AWS pour paginer le fichier. Ces cas soulignent pourquoi la gestion des fichiers volumineux est un point de douleur fréquent pour les développeurs SuiteScript.

API de streaming SuiteScript 2.x et fichiers volumineux

En réponse à ces besoins, NetSuite a introduit l'**API de streaming de fichiers plats (Flat File Streaming API)** dans la version 2017.1 (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com). Cela fournit de nouvelles méthodes dans `N/file` qui permettent de traiter de gros fichiers CSV ou texte de manière incrémentielle, ligne par ligne, en gardant chaque morceau sous les 10 Mo. Avec ces API, il est possible de construire ou de lire un fichier d'une taille totale pratiquement illimitée, tant que chaque étape d'écriture/lecture est ≤ 10 Mo. Les principales fonctionnalités de streaming sont :

- **Ajout par ligne :** Créez un objet fichier initial (avec ou sans en-tête), puis appelez de manière répétée `file.appendLine({value: ...})` pour ajouter des lignes de texte. Chaque ligne ajoutée doit elle-même être inférieure à 10 Mo (Source: docs.oracle.com), mais il n'y a pas de limite de taille totale pour le fichier. On peut ajouter des milliers de lignes (même si, combinées, elles totalisent plus de 100 Mo) puis enfin appeler `file.save()` pour enregistrer le fichier volumineux. L'exemple de code NetSuite pour 2017.1 montre exactement ce modèle : création d'un fichier CSV, utilisation répétée de `appendLine`, puis enregistrement (Source: netsuitedocumentation1.gitlab.io).

- **Itération par ligne :** Pour la lecture, `file.lines.iterator()` renvoie un itérateur qui récupère une ligne à la fois (chaque ligne < 10 Mo) à partir d'un fichier chargé (Source: docs.oracle.com). Vous pouvez parcourir les lignes avec `.each(function(line){ ... })`. Cela garantit qu'à aucun moment vous ne conservez plus d'une ligne en mémoire. Comme l'indique la documentation, « vous pouvez appeler cette méthode plusieurs fois pour parcourir le contenu du fichier sous forme de *flux* » avec la même limitation de 10 Mo par ligne (Source: docs.oracle.com). Dans les scripts Map/Reduce, c'est encore plus puissant : l'étape de mappage peut prendre une référence de fichier en entrée, et NetSuite invoquera chaque pod Map sur les lignes successives du fichier (Source: netsuitedocumentation1.gitlab.io).
- **Réinitialisation des flux :** La méthode `file.resetStream()` permet à un objet fichier de basculer entre la lecture et l'écriture. Par exemple, en théorie, on pourrait ouvrir un fichier, itérer sur certaines lignes, puis réinitialiser le flux et y ajouter des lignes. Cette réinitialisation est nécessaire car, par défaut, NetSuite n'autorise pas une lecture et une écriture actives sur le même fichier sans elle (Source: netsuitedocumentation1.gitlab.io). Après la réinitialisation, le contenu du fichier n'est toujours en mémoire que jusqu'à ce que `save()` soit appelé.
- **Propriété de taille dynamique :** La propriété `file.size` (sur un objet fichier) se met à jour dynamiquement pour refléter le nombre total d'octets de contenu actuellement conservés/enregistrés. Cela permet aux scripts de surveiller la taille d'un fichier au fur et à mesure qu'ils ajoutent des lignes (Source: netsuitedocumentation1.gitlab.io).

Ces API de streaming changent fondamentalement la gestion des fichiers volumineux. Alors qu'avant 2017, on « ne pouvait pas accéder facilement » à un fichier > 10 Mo (Source: docs.oracle.com), on s'assure désormais simplement que chaque lecture ou écriture est ≤ 10 Mo en manipulant les lignes. Par exemple :

```
// Pseudocode pour l'écriture d'un gros CSV :
var csvFile = file.create({ name: 'big.csv', contents: 'col1,col2\n', folder: 100, fileType: file.Type.CSV });
csvFile.appendLine({ value: 'row1col1,row1col2' });
// ... ajouter beaucoup d'autres lignes (<10 Mo chacune) ...
csvFile.save();  // le fichier final peut être >> 10 Mo au total
```

```
// Pseudocode pour la lecture d'un gros CSV :
var report = file.load({ id: someLargeFileId });
var iterator = report.lines.iterator();
iterator.each(function(line) {
    // traiter une ligne (<10 Mo) à la fois
    return true; // continuer à la ligne suivante
});
```

La documentation de NetSuite note explicitement que « la limite de 10 Mo ne s'applique désormais qu'aux lignes individuelles, et non au fichier entier » (Source: docs.oracle.com). En d'autres termes, l'utilisation de `appendLine` / `iterator` vous permet de construire ou d'analyser nativement des fichiers CSV et texte arbitrairement volumineux. Cette approche de streaming est native à SuiteScript 2.x et ne nécessite aucun outil externe pour le contenu **basé sur du texte**. C'est souvent la première stratégie recommandée pour le traitement de fichiers volumineux (Source: www.houseblend.io) (Source: netsuitedocumentation1.gitlab.io).

Cependant, il existe des mises en garde et des limitations. La prise en charge du streaming de fichiers plats **ne fonctionne que pour le texte brut ou le CSV**. Les fichiers binaires (tels que les PDF, les images, les documents Office ou les blobs JSON) *ne peuvent pas* être ajoutés ligne par ligne ou itérés ligne par ligne. Pour les binaires, la seule option reste de les découper en externe ou d'utiliser un stockage externe. De plus, le streaming a certaines contraintes d'utilisation : une fois que vous commencez à itérer sur un fichier, vous ne pouvez pas y ajouter de contenu avant d'avoir appelé `file.resetStream()` (Source: netsuitedocumentation1.gitlab.io). Le mélange des lectures et des écritures nécessite des réinitialisations minutieuses. Et les scripts doivent toujours s'assurer que chaque segment (ligne) reste inférieur à 10 Mo, sinon l'écriture échouera (Source: docs.oracle.com) (Source: docs.oracle.com).

Intégration Map/Reduce. NetSuite a également amélioré les scripts Map/Reduce pour tirer parti de ce streaming. En 2017.1, l'étape « getInputData » de Map/Reduce peut directement prendre une référence de fichier, et chaque instance **map()** recevra une ligne du fichier en entrée (Source: netsuitedocumentation1.gitlab.io). Cela permet aux CSV massifs d'être traités en parallèle entre les agents, en décomposant efficacement un fichier énorme en tâches de mappage ligne par ligne. La documentation indique : *« Dans le cadre des améliorations du streaming de fichiers, le type de*

script map/reduce… peut désormais transmettre une ligne par invocation de fonction map » (Source: netsuitedocumentation1.gitlab.io). Un modèle typique est : `getInputData = function() { return {type: 'file', id: someFileId}; }`, puis chaque `map()` obtient une ligne. Les développeurs rapportent que Map/Reduce rend le traitement des données volumineuses beaucoup plus simple, puisque la plateforme gère l'itération en interne. Notamment, l'utilisation de Map/Reduce nécessite toujours que le fichier initial ne dépasse pas 10 Mo s'il est importé via SuiteScript (par exemple `file.create`), à moins que le fichier ne provienne d'une tâche ou d'une exportation de l'interface utilisateur (Source: stackoverflow.com). Si le fichier commence petit (ou est divisé), Map/Reduce peut s'étendre au-delà de 10 Mo ligne par ligne.

SuiteScript Version 1.0. En revanche, SuiteScript 1.0 n'avait *aucune* API de streaming. À cette époque, les développeurs devaient littéralement **prédécouper tout fichier volumineux** avant de le télécharger ou s'assurer qu'aucun segment unique ne dépassait 10 Mo. Par exemple, l'exportation d'un grand ensemble de données nécessitait l'enregistrement de plusieurs fichiers plus petits ou l'utilisation d'outils d'importation CSV côté serveur. Les ajouts de 2017 rendent SuiteScript 2.x beaucoup plus performant, mais la contrainte héritée sous-tend toujours la conception de la plateforme.

En résumé, les API de streaming de SuiteScript 2.x permettent un **traitement ligne par ligne** pour contourner le plafond de 10 Mo pour les fichiers texte/CSV (Source: docs.oracle.com) (Source: netsuitedocumentation1.gitlab.io). Lorsqu'ils sont utilisés correctement, les scripts peuvent assembler ou lire des fichiers volumineux de n'importe quelle taille, sous réserve uniquement des quotas de stockage globaux. La principale limitation de cette méthode est qu'elle n'est utilisable que pour le contenu textuel ; pour les importations de binaires ou d'enregistrements multilignes, d'autres stratégies sont nécessaires (discutées ci-dessous).

Solutions de contournement et modèles pour les fichiers volumineux

Reconnaissant le plafond de 10 Mo, les développeurs et architectes NetSuite ont mis au point diverses stratégies pour gérer des fichiers plus volumineux. Le tableau 2 ci-dessous résume les principaux modèles rencontrés dans les discussions communautaires et la documentation, ainsi que leurs compromis. Ces approches vont des techniques purement SuiteScript aux solutions externes ou hybrides.

Tableau 2. Stratégies pour la gestion des fichiers > 10 Mo dans NetSuite. Avantages et inconvénients de chaque approche, tirés de la documentation et des sources communautaires (Source: www.houseblend.io) (Source: stackoverflow.com) (Source: stackoverflow.com):

APPROCHE	DESCRIPTION	AVANTAGES	INCONVÉNIENTS / NOTES
Streaming natif (SuiteScript)	Utiliser `file.create` + plusieurs appels `appendLine` (chaque ≤ 10 Mo) pour construire un gros texte/CSV ; ou `file.load` + `lines.iterator` pour lire.	Peut produire/analyser des fichiers de plusieurs centaines de Mo sans outils externes. API Suite natives (Source: docs.oracle.com) (Source: netsuitedocumentation1.gitlab.io).	Ne fonctionne que pour les fichiers texte plats/CSV. Impossible de streamer des binaires/PDF. Doit gérer la réinitialisation entre lecture/écriture.
Script Map/Reduce	Alimenter un fichier (plus petit) vers Map/Reduce via `getInputData`, puis traiter chaque ligne en parallèle. Vous pouvez également faire en sorte que chaque map ajoute à un gros fichier via le streaming.	Hautement évolutif ; traitement parallèle de gros CSV. Simplifie la logique ligne par ligne. Conçu par NetSuite en 2017.1 (Source: netsuitedocumentation1.gitlab.io).	Nécessite un fichier initial dans le File Cabinet (≤ 10 Mo ou divisé). Doit utiliser le type de script Map/Reduce.

| **Outil d'import CSV** | Utilisez l'outil d'import CSV intégré de NetSuite (UI ou API) pour traiter les données par lots, jusqu'à 50 Mo par fichier (Source: stackoverflow.com). | Contourne les limites de SuiteScript ; conçu pour les chargements de données massifs. | Plafond strict (25 000 lignes ou ~50 Mo) (Source: stackoverflow.com). Limité aux enregistrements/formats pris en charge. Aucune logique personnalisée pendant l'import. | | **N/task SCHEDULED EXTRACT** | Utilisez `N/task` (tâches de recherche ou Map/Reduce) pour générer des fichiers CSV. Par exemple, une recherche enregistrée planifiée peut publier les résultats dans un fichier du File Cabinet (les tâches de fichier s'exécutent en dehors de la mémoire du script) (Source: stackoverflow.com). | Les tâches s'exécutent dans un contexte séparé, contournant souvent les limites de mémoire des scripts. Exemple : sortie CSV via searchTask. | Souvent limité à 10 Mo par fragment. Configuration complexe ; peut toujours nécessiter un découpage en segments. | | **SuiteTalk / Chargement via RESTlet** | Utilisez SuiteTalk SOAP ou un RESTlet pour envoyer des parties de fichiers à NetSuite. Par exemple, on peut effectuer une boucle avec plusieurs appels `file.create` via des API REST. | S'intègre aux systèmes externes ; SuiteTalk peut permettre des opérations asynchrones (addList) pour charger de petits segments. | Les appels REST/SOAP sous-jacents atteignent toujours la limite de ~10 Mo par

requête. Nécessite souvent un découpage manuel des données. | | **Compression / Encodage** | Compressez ou encodez le segment de données pour réduire sa taille avant `file.create`. Par exemple, utilisez `N/compress` (gzip) (Source: archive.netsuiteprofessionals.com) pour que le texte volumineux devienne plus petit, puis enregistrez-le. | Permet de faire tenir plus de contenu sous les 10 Mo. Tire parti du module de compression de SuiteScript. | SuiteScript ne peut pas décompresser nativement les fichiers téléchargés ; l'utilisation du résultat est incertaine. Complexité ; peut ne pas toujours suffire. | | **Stockage externe / Intégration** | Déchargez l'intégralité du fichier vers un cloud externe (ex: AWS S3, Azure) et ne stockez qu'une petite référence dans NetSuite. Utilisez SuiteScript ou des outils d'intégration pour diviser le fichier et le récupérer selon les besoins (ex: appels REST/HTTPS) (Source: stackoverflow.com). | Pratiquement aucune limite de taille côté externe ; décharge le stockage et la bande passante. Peut gérer TOUT type ou taille de fichier. | Complexité et coûts accrus. Problèmes de synchronisation des données et de sécurité. Plus d'éléments mobiles. Nécessite une gestion séparée. | | **SuiteApp (SkyDoc, etc.)** | Utilisez une SuiteApp tierce qui stocke de manière transparente les fichiers dans un stockage cloud externe. Le fichier apparaît dans NetSuite mais est en réalité conservé (et récupéré) sur S3/Azure (Source: www.79consulting.com). | Expérience utilisateur fluide (fichiers « dans NetSuite ») ; taille de fichier virtuellement illimitée. Intégré à l'interface et aux enregistrements. | Nécessite l'achat/l'installation d'une SuiteApp. Ajoute des frais et un silo de données supplémentaire. Dépend de la maintenance du fournisseur. | | **Évitement (Liens uniquement)** | Au lieu de charger le fichier, stockez-le sur un partage de fichiers d'entreprise ou un système de gestion documentaire et enregistrez uniquement une URL ou un ID dans NetSuite. | Aucune limite de taille de fichier dans NetSuite. Simple à mettre en œuvre (juste un champ URL). | Rompt l'intégration native. Les utilisateurs doivent quitter NetSuite pour visualiser. Les liens peuvent se briser ; pas de versionnage ni de fonctionnalités NetSuite. |

(Les approches et commentaires ci-dessus sont tirés de la documentation NetSuite et de sources communautaires (netsuite-documentation1.gitlab.io (Source: stackoverflow.com) (Source: stackoverflow.com) (Source: archive.netsuiteprofessionals.com) (Source: www.79consulting.com).)

Chacune de ces stratégies est explorée en détail ci-dessous, avec des exemples et des avis d'experts :

1. Découpage/Partitionnement des fichiers

Une solution de contournement simple consiste à **diviser un gros fichier en petits morceaux** (≤ 10 Mo chacun) avant le chargement ou le traitement. Par exemple, on peut utiliser un script Python externe ou un utilitaire Unix pour diviser un CSV de 30 Mo en trois fichiers de 10 Mo, puis importer chacun individuellement. Cela imite la « bonne pratique » pré-SuiteScript 2.0 notée par la documentation NetSuite : « Les gros fichiers devaient être divisés en partitions pour être enregistrés avec succès » (Source: netsuitedocumentation1.gitlab.io).

Exemple de cas : Un développeur NetSuite devait charger un CSV de 20 Mo contenant tous les articles d'inventaire. Comme `file.create` échouait, il l'a divisé manuellement en deux CSV de 10 Mo. Chaque fichier a ensuite été importé ou traité séparément (Source: www.houseblend.io). Le développeur a commenté que cette solution de contournement « était épuisante », mais qu'elle a résolu le problème immédiat (Source: www.houseblend.io). Une autre équipe confrontée à des importations massives d'enregistrements a de même « divisé les CSV en plusieurs fichiers » par nombre de lignes et les a traités un par un (Source: www.houseblend.io).

Avantages : Cette méthode est simple et fonctionne avec n'importe quel type de fichier (texte ou binaire) puisque vous créez physiquement des fichiers plus petits. Elle ne nécessite pas de script spécial – des outils courants peuvent diviser les fichiers. Elle évite également totalement la limite de 10 Mo, puisque chaque pièce chargée s'y conforme.

Inconvénients : Elle ajoute une charge de travail manuelle ou de codage pour diviser, puis réassembler si nécessaire. Vous vous retrouvez avec de nombreux fichiers au lieu d'un seul, ce qui complique la coordination. Vous devez vous assurer que les processus en aval gèrent les multiples parties (ex: les combiner). Pour les fichiers binaires, le découpage/réassemblage peut nécessiter une logique supplémentaire. Globalement, le découpage augmente la complexité et n'est réalisable que lorsque les segments ont un sens (ex: découpage de CSV par ligne).

2. Streaming natif SuiteScript (Append/Iterator)

Cette méthode utilise exclusivement les API de streaming intégrées de NetSuite, comme indiqué ci-dessus. Pour créer un gros fichier, un script appelle `file.create()` avec un contenu vide ou d'en-tête, puis utilise des appels répétés à `file.appendLine({value: ...})` – chaque ligne faisant < 10 Mo – pour construire le fichier textuellement (Source: docs.oracle.com) (Source: netsuitedocumentation1.gitlab.io). Inversement, pour lire un gros fichier, un script appelle `file.load(id)` sur un fichier (peut-être préexistant dans le Cabinet) puis itère en utilisant `lines.iterator()` (Source: docs.oracle.com).

Exemple de cas : Dans les notes de version 2017.1 de NetSuite, l'exemple de code montre la création et l'enregistrement d'un CSV avec de nombreuses lignes par des appels répétés à `appendLine` (Source: netsuitedocumentation1.gitlab.io) (Source: netsuitedocumentation1.gitlab.io). Dans les questions-réponses communautaires, les développeurs soulignent à plusieurs reprises que l'utilisation de `appendLine` permet des fichiers

« d'une taille pratiquement illimitée » car **seuls les 10 Mo de chaque ligne comptent** (Source: docs.oracle.com) (Source: www.houseblend.io). Contrairement à la création d'un fichier à partir d'une chaîne de 20 Mo (qui échoue), le streaming permet d'écrire 20 Mo sous forme de deux lignes de 10 Mo (ou de nombreuses lignes plus petites).

Avantages : **Aucune dépendance externe**. Fonctionne entièrement au sein de SuiteScript 2.x. Idéal pour les **données texte ou CSV volumineuses** car les lignes sont une unité naturelle. Nous pouvons, par exemple, écrire en streaming des millions de lignes dans un seul fichier (Source: netsuitedocumentation1.gitlab.io). Cela exploite également un script familier (pas de nouveaux systèmes). Pour la lecture, cela économise la mémoire.

Inconvénients : **Limité au texte.** Ne peut pas gérer les données binaires/PDF. L'écriture est plus lente, car chaque appel sollicite la plateforme. De plus, `appendLine` ne peut pas être entrelacé avec des lectures sans réinitialiser les flux (Source: netsuitedocumentation1.gitlab.io). Les scripts doivent gérer soigneusement les petits segments. Il peut y avoir des problèmes subtils : par exemple, si le fichier est très volumineux (centaines de Mo), appeler `appendLine` des centaines de fois peut consommer la gouvernance du script ou s'exécuter lentement. Cependant, cela évolue bien pour les CSV et les impressions avec des millions de lignes en Map/Reduce. Notamment, un commentaire d'utilisateur a souligné qu'avec le streaming, la limite de 10 Mo « est par ligne » (Source: docs.oracle.com).

3. Scripts Map/Reduce

Le type **Map/Reduce** de NetSuite offre des moyens puissants de gérer de grands ensembles de données, souvent en conjonction avec le streaming de fichiers (Source: netsuitedocumentation1.gitlab.io). Une approche consiste à utiliser l'étape d'entrée de Map/Reduce pour itérer sur un fichier. Pour les entrées texte/CSV très volumineuses, stocker le fichier dans NetSuite et le transmettre simplement à `getInputData()` permet à chaque tâche de mappage de traiter une ligne. La plateforme partitionne automatiquement le fichier en lignes en arrière-plan (Source: netsuitedocumentation1.gitlab.io). Chaque invocation de mappage reçoit effectivement un segment ≤ 10 Mo (une ligne), donc la règle des 10 Mo est respectée sans code supplémentaire.

Exemple de cas : Un modèle courant est : `getInputData = () => file.load({id: myFileId});`. NetSuite gère ensuite l'itération ligne par ligne dans l'étape de mappage (Source: netsuitedocumentation1.gitlab.io). La documentation officielle 2017.1 indique que *« le type de script map/reduce a été amélioré afin que vous puissiez diffuser le contenu de fichiers texte ou CSV pendant l'étape de mappage »*, en passant une ligne par mappage (Source: netsuitedocumentation1.gitlab.io). Des publications de développeurs ont noté que Map/Reduce peut traiter des CSV arbitrairement volumineux via ce mécanisme. Un fil de discussion Q&A suggérait simplement d'utiliser le streaming intégré de Map/Reduce plutôt que des boucles manuelles (Source: stackoverflow.com).

Alternativement, on peut utiliser Map/Reduce pour **générer** un gros fichier. Par exemple, un modèle instructif est : dans un Suitelet ou un script planifié, divisez les données et passez un tableau à `task.create(fileID)` sur un travail Map/Reduce conçu pour l'assembler via `appendLine`.

Avantages : Gère les données en parallèle et à grande échelle. Map/Reduce est conçu pour les très gros chargements et peut dépasser le temps d'exécution standard des scripts. Il libère le développeur de l'itération manuelle dans les événements utilisateur. L'intégration de type AWS ou iPaaS devient plus facile. De plus, comme chaque mappage opère sur un petit morceau, les limites de mémoire ne sont pas un problème. Dans de nombreux cas, Map/Reduce simplifie le code (pas besoin d'appeler manuellement `each`).

Inconvénients : Plus complexe à configurer (nécessite la création d'un script Map/Reduce). Le fichier initial doit exister dans NetSuite (vous devez donc toujours descendre en dessous de 10 Mo pour commencer, ou charger manuellement de petits fichiers via `getInputData` de Map/Reduce). De plus, Map/Reduce utilise la gouvernance différemment et peut nécessiter une planification pour rester dans les unités d'utilisation allouées. Certains développeurs ont noté que même avec Map/Reduce, la création du fichier pour l'alimenter nécessite toujours du streaming ou du découpage (Source: stackoverflow.com) (Source: stackoverflow.com).

4. Planification de tâches SuiteScript ou exportations de recherche

Au lieu du streaming manuel, on peut exploiter les API de planification de NetSuite. Par exemple, le module `N/task` de NetSuite permet de planifier un **export de recherche** vers un CSV. Une recherche enregistrée peut être initiée par programme en tant que tâche qui écrit les résultats dans un chemin de fichier. Crucialement, cette approche se produit sur le serveur NetSuite et ne charge pas l'intégralité du contenu dans la mémoire du script initiateur.

Exemple de cas : Un développeur a posé une question sur l'exportation d'une grande recherche enregistrée (plus de 10 Mo) via SuiteScript (Source: stackoverflow.com). Les répondants ont souligné que l'utilisation de `task.create({taskType: SEARCH})` et la définition de `searchTask.savedSearchId` et `searchTask.filePath` amèneront NetSuite à exécuter la recherche et à générer un CSV dans le File Cabinet

(Source: stackoverflow.com). Cela contourne efficacement la limite de 10 Mo de SuiteScript car les résultats de recherche sont diffusés par le système en interne, et non via la mémoire du script appelant. En effet, les réponses sur StackOverflow ont confirmé que cette méthode réussit là où un simple `file.create` échouerait (Source: stackoverflow.com). L'exemple de code de cette réponse :

```
var searchTask = task.create({ taskType: task.TaskType.SEARCH });
searchTask.savedSearchId = 51;
searchTask.filePath = 'ExportFolder/export.csv';
searchTask.submit();
```

Cette tâche s'exécute de manière asynchrone et déposera un gros CSV dans le dossier spécifié (Source: stackoverflow.com).

Avantages : Décharge le travail vers la file d'attente des tâches internes de NetSuite. Peut générer de gros fichiers d'exportation sans atteindre la limite de mémoire du script. Comme la logique d'exportation searchCSV est séparée, le fichier peut dépasser 10 Mo (souvent jusqu'à des centaines de Mo) car il n'utilise pas les API de contenu SuiteScript. De plus, c'est idéal pour exporter de grands résultats de recherche enregistrée sans écrire de code pour ajouter des lignes manuellement.

Inconvénients : Limité aux données pouvant provenir d'une recherche. Vous perdez un contrôle granulaire sur le formatage CSV (bien que vous puissiez parfois définir plus de colonnes). Néanmoins, la sortie de la tâche peut rencontrer des limites d'importation CSV (~50 Mo ou 25 000 lignes (Source: stackoverflow.com). De plus, la configuration des tâches planifiées peut être fastidieuse et le débogage est plus difficile (la sortie apparaît dans un dossier du Cabinet). C'est une stratégie particulièrement destinée aux exportations de données, et non à la création arbitraire de fichiers. Certaines réponses sur les forums suggèrent également de combiner cela avec le streaming de fichiers si un traitement supplémentaire est nécessaire.

5. Chargements via API SuiteTalk / RESTlet

Un autre angle consiste à utiliser les services Web ou l'API REST de NetSuite pour charger des fichiers. SuiteTalk (SOAP ou REST) permet d'ajouter par programme des enregistrements de fichiers (via des opérations `add`), ou un RESTlet peut être écrit pour accepter un chargement externe. En théorie, un middleware d'intégration pourrait charger un fichier en plusieurs parties via plusieurs appels SOAP ou invocations de RESTlet.

Par exemple, on pourrait écrire un RESTlet qui accepte des POST HTTP de segments de fichiers et appelle `file.create` sur chaque segment. Ou en SOAP, utiliser `add(List)` avec des morceaux de données base64. Cependant, ces API vivent finalement dans la même limite de plateforme, donc chaque appel est généralement toujours limité à ~10 Mo de contenu. Les discussions sur Internet notent que l'exécution d'un chargement via SuiteTalk/REST nécessite toujours un découpage côté client si le fichier fait > 10 Mo (Source: www.houseblend.io).

Avantages : Cette approche permet de charger des fichiers volumineux depuis l'extérieur de NetSuite (ex: depuis Dell Boomi, Celigo ou un middleware personnalisé). Elle peut s'intégrer à des systèmes externes ou à des chargements d'utilisateurs. En pratique, on pourrait utiliser un champ de fichier dans un enregistrement et pousser de grosses pièces jointes via des appels `attach` SOAP.

Inconvénients : La limite par appel demeure. Les outils comme Restlet ou la méthode `add` de SOAP génèrent souvent des erreurs s'ils reçoivent plus de 10 Mo en une seule fois. Par conséquent, le middleware doit de toute façon implémenter un découpage (chunking), souvent sous forme d'appels multiples ou de fractionnement. Il existe également une surcharge liée à l'authentification et au codage. C'est plus complexe qu'un simple SuiteScript, et ce n'est toujours pas réellement illimité. De nombreux posts sur les forums concluent que les RESTlets « atteignent toujours la limite de 10 Mo par appel » (Source: www.houseblend.io), ce qui signifie qu'il faut diviser le téléchargement en plusieurs morceaux. En bref, cela permet de déplacer le découpage en dehors de SuiteScript, mais ne l'élimine pas.

6. Astuces de compression ou d'encodage

Une astuce plus *créative* consiste à **compresser** les données avant de les envoyer à NetSuite. SuiteScript 2.x a introduit le module `N/compress` (dans la version 2020.2) qui peut compresser des chaînes ou des octets au format gzip ou deflate. En théorie, on pourrait compresser à la volée un gros fichier JSON ou CSV en gzip, puis utiliser `file.create` sur le blob compressé. Si la compression gzip est efficace, la chaîne compressée peut passer sous la barre des 10 Mo, même si l'original était plus volumineux (Source: archive.netsuiteprofessionals.com). Plus tard, un processus distinct pourrait le décompresser.

Exemple cité : Sur un forum de NetSuite Professionals, un utilisateur a posé une question sur le dépassement des 10 Mo avec `file.create`. Une réponse suggérait d'utiliser la bibliothèque `N/compress` pour compresser le fichier en gzip avant de l'enregistrer (Source: archive.netsuiteprofessionals.com). Cela permettrait théoriquement de condenser les données.

Avantages : Pourrait permettre de stocker plus de données dans un seul fichier en réduisant sa taille en mémoire. Aucun système externe n'est nécessaire si les données sont hautement compressibles.

Inconvénients : NetSuite ne peut pas décompresser nativement ou utiliser le fichier compressé par la suite. Vous devriez le récupérer et le décompresser dans un client externe à SuiteScript (à moins que NetSuite ne fournisse ultérieurement une API de décompression). Cela n'aide que si les données sont compressibles (le texte se compresse bien, mais pas les images/PDF). Cette approche est rarement utilisée en pratique car elle complique la récupération des données. C'est un exemple de réflexion « hors des sentiers battus », mais ce n'est pas une solution robuste pour la plupart des cas d'utilisation métier.

7. Stockage externe / Intégration (AWS S3, FTP, etc.)

De nombreuses organisations choisissent de déporter entièrement le fichier vers un système externe. Par exemple, un rapport ou un document volumineux peut être téléchargé dans un **bucket AWS S3** ou un autre stockage cloud, et seul un lien ou une petite référence est stocké dans NetSuite. SuiteScript peut toujours effectuer des appels HTTPS (via `https.get`) pour récupérer des morceaux du fichier inférieurs à 10 Mo si nécessaire. Une réponse sur StackOverflow décrit exactement ce modèle : un AWS Lambda télécharge/enregistre un gros fichier sur S3, puis SuiteScript demande des « pages » (morceaux) via des appels d'URL, chacun faisant moins de 10 Mo, en enregistrant chaque partie comme un fichier séparé (Source: stackoverflow.com).

Les variantes courantes de cette stratégie incluent :

- **SFTP/FTP** : Utiliser le module `N/sftp` dans SuiteScript (ou un ETL externe) pour récupérer de gros fichiers depuis un serveur FTP par parties.
- **Middleware** : Utiliser une plateforme d'intégration (Dell Boomi, Celigo, Mulesoft, etc.) pour gérer le téléchargement. Par exemple, le flux Data Loader de Celigo peut télécharger de très gros fichiers sources dans NetSuite en les découpant et en les mappant (Source: docs.celigo.com). Le connecteur NetSuite de Boomi prend également en charge les pièces jointes asynchrones (jusqu'aux limites propres à Boomi).
- **SuiteTalk avec stockage externe** : Certaines intégrations permettent de stocker un enregistrement dans NetSuite avec une référence de fichier externe spéciale (par exemple, en référençant un objet S3).

Exemple de cas : L'utilisateur de StackOverflow Bobby Knights suggère d'utiliser un AWS Lambda comme proxy : le Lambda récupère un énorme CSV depuis un tiers et l'écrit sur S3 ; ensuite, SuiteScript interroge l'URL du Lambda ou de S3 pour parcourir le fichier par pages de moins de 10 Mo (Source: stackoverflow.com). De cette façon, le travail lourd et le stockage se font en dehors de NetSuite.

Avantages : Pratiquement **aucune limite théorique** sur la taille du fichier, puisque le stockage externe la gère. Décharge les coûts de bande passante et de stockage. De nombreuses entreprises possèdent déjà des systèmes S3 ou CMS (Box, SharePoint) et peuvent s'y intégrer. SuiteScript ne traite que des morceaux gérables ou des pointeurs.

Inconvénients : Ajoute une complexité et une latence significatives. Les données doivent être synchronisées entre les systèmes. Des préoccupations en matière de sécurité et de conformité apparaissent (les données sont-elles chiffrées en transit/stockage ?). Les utilisateurs ont désormais besoin d'un accès au système externe pour visualiser les fichiers. Il y a également un coût supplémentaire (frais S3, développement de l'intégration). Dans l'ensemble, un modèle de stockage externe contourne efficacement la limite de NetSuite, mais au prix de ne plus faire de NetSuite la source unique de vérité pour les gros fichiers.

8. SuiteApps tierces (ex. Tvarana SkyDoc)

Conscients de cette lacune, certains fournisseurs ont développé des intégrations spécialisées. **SkyDoc** (par Tvarana) est une SuiteApp qui intègre le stockage de fichiers cloud dans NetSuite. Elle stocke de manière transparente les pièces jointes volumineuses dans AWS S3 ou Azure Blob, tout en les présentant dans l'interface utilisateur de NetSuite comme s'il s'agissait de fichiers normaux. En arrière-plan, le contenu n'est pas soumis à la contrainte de 10 Mo de NetSuite car il réside dans S3. Seuls les métadonnées et les liens résident dans NetSuite.

Étude de cas : Stairlock (un fabricant de 119 personnes) a déployé SkyDoc pour NetSuite. Le document de mise en œuvre rapporte qu'ils ont *« éliminé les limitations de taille de fichier »* après avoir migré plus de 90 000 fichiers vers le stockage cloud de SkyDoc (Source: www.79consulting.com). En effet, leurs utilisateurs peuvent télécharger et travailler avec des fichiers PDF et CAO de plusieurs Go via l'interface de

NetSuite sans atteindre aucune limite. De même, les forums d'utilisateurs et le blog de Tvarana notent que SkyDoc permet de « stocker, gérer et partager de gros fichiers de manière transparente au sein de NetSuite » (Source: www.79consulting.com). (Intuitivement, on pourrait aussi mentionner les connecteurs Box ou Dropbox, mais SkyDoc est une SuiteApp directe adaptée à ce scénario.)

Avantages : Offre une expérience utilisateur fluide : les utilisateurs cliquent sur « Ajouter un fichier » sur un enregistrement, mais le fichier va dans le cloud. Aucun changement de code SuiteScript nécessaire pour la plupart des cas d'utilisation. Gère des tailles de fichiers pratiquement illimitées, des formats riches, des lots, etc. Prend en charge le versionnage, le partage et même les pièces jointes par e-mail.

Inconvénients : Il s'agit d'un **module complémentaire payant**. Dépendance envers un fournisseur et un service externe. Maintenance supplémentaire (bien que souvent plus simple qu'une intégration faite maison). Certains clients peuvent avoir des préoccupations concernant le stockage des données hors site (ou avoir besoin de souveraineté des données). De plus, si NetSuite propose un jour une solution native, une SuiteApp pourrait devenir redondante. Mais pour l'instant, c'est le seul moyen clé en main de « contourner » la taille de fichier dans NetSuite.

9. Évitement / Lien uniquement

Enfin, une « solution de contournement » conceptuelle consiste simplement à **ne pas stocker le fichier du tout**. Au lieu de cela, un document externe (comme un PDF géant ou une vidéo) est conservé sur un serveur d'entreprise ou un intranet, et NetSuite stocke simplement un lien hypertexte ou une référence vers celui-ci. Ce n'est pas un processus de téléchargement de fichier en soi, mais c'est un modèle pour gérer les gros fichiers en pratique. Par exemple, une Suitelet ou un formulaire peut permettre à l'utilisateur de saisir une URL vers un fichier SharePoint ou S3.

Avantages : Les limites de NetSuite ne s'appliquent plus puisque NetSuite ne détient jamais le contenu du fichier. Taille illimitée, et gestion de contenu potentiellement plus riche à l'extérieur.

Inconvénients : Perd la plupart des fonctionnalités de NetSuite pour les fichiers (aperçu, historique des versions, stockage dans les transactions, recherche). Les utilisateurs doivent quitter NetSuite pour accéder au fichier. L'expérience est plus décousue, et les liens risquent de se briser ou de poser des problèmes d'accès. Généralement envisagé uniquement lorsque les autres options sont impossibles.

Preuves et études de cas

Nous avons déjà fait allusion à plusieurs exemples concrets. Pour résumer les enseignements clés de la pratique :

- **Forums de développeurs :** Comme l'a noté une session de questions-réponses, écrire **directement** un fichier de plus de 10 Mo en SuiteScript (sans streaming) est impossible. Les développeurs demandent souvent « existe-t-il un moyen de créer un fichier de plus de 10 Mo ? » et reçoivent des réponses conseillant `appendLine` ou des services externes (Source: stackoverflow.com) (Source: stackoverflow.com). Le consensus : NetSuite n'a pas d'API de « suppression de limite supérieure », vous devez donc contourner le problème par le streaming ou des solutions hors plateforme (Source: www.houseblend.io) (Source: www.houseblend.io). Un utilisateur expérimenté a testé cela sans détour, concluant que dans SuiteScript 2.x « la limite est de 10 Mo par fichier si vous essayez d'enregistrer le contenu directement » et que tout ce qui est plus grand nécessite un découpage (Source: www.houseblend.io).
- **Conseils officiels :** L'aide de NetSuite invite les développeurs à utiliser le modèle de streaming, indiquant explicitement que la limite de 10 Mo s'applique aux opérations sur fichier complet, et que `appendLine`/itérateur « remplacent raisonnablement » l'ancien besoin de diviser les fichiers (Source: docs.oracle.com) (Source: docs.oracle.com). La page d'aide `File.getContents` *suggère* que pour les fichiers plus volumineux, il faut passer à l'itération de ligne ou à `appendLine` (signalant la solution prévue par NetSuite) (Source: docs.oracle.com).
- **Solutions tierces :** L'étude de cas Stairlock (Source: www.79consulting.com) montre que de vraies entreprises paient pour des SuiteApps afin d'éviter le problème. Stairlock a « éliminé les limitations de taille de fichier » grâce à SkyDoc, ce qui signifie qu'ils ne se heurtent plus à la limite de 10 Mo dans leur utilisation normale. De même, Tvarana (le fournisseur de SkyDoc) avertit que la limite de 10 Mo de NetSuite « entrave la productivité » sur les gros PDF et images (Source: www.houseblend.io). Ces exemples illustrent que dans au moins certains secteurs (fabrication, CAO, médias), la limite est un véritable goulot d'étranglement, et des solutions dédiées ont émergé.
- **Considérations de performance :** Bien qu'au-delà de la taille du fichier, il convient de noter les *implications sur les performances* des fichiers énormes. NetSuite ne publie pas officiellement de taille de fichier maximale, mais la documentation de l'interface utilisateur suggère que même les téléchargements de gros fichiers peuvent prendre plus de temps et que les utilisateurs abandonnent parfois si c'est trop lent (Source: docs.oracle.com). En pratique, les gros fichiers dans le File Cabinet (centaines de Mo ou plus) peuvent ralentir les temps de chargement dans

l'interface utilisateur et les sauvegardes. Les propres avis de NetSuite recommandent souvent de conserver des fichiers ≤ 100 Mo pour de meilleurs résultats (bien que des fichiers > 100 Mo puissent techniquement exister). Les développeurs doivent évaluer s'il faut diviser les fichiers uniquement pour maintenir les performances et la fiabilité.

- **Tendances statistiques :** Les données d'Egnyte (Source: betanews.com) soulignent que la taille des fichiers augmente dans tous les secteurs. Même si les fichiers moyens ne font que quelques Mo aujourd'hui, la *longue traîne* des fichiers très volumineux devrait croître à mesure que les entreprises numérisent davantage de contenu. Avec l'augmentation des images, des vidéos, des données IoT et des médias riches, les développeurs doivent s'attendre à ce que le plafond de 10 Mo devienne encore plus courant dans les années à venir.

- **Outils d'intégration :** Les outils d'intégration de plateforme ont mûri pour résoudre ce problème. Par exemple, Celigo Data Loader offre un moyen graphique de télécharger des fichiers CSV/JSON/Excel dans NetSuite, en gérant les gros fichiers avec élégance (Source: docs.celigo.com). La documentation de Celigo mentionne explicitement que Data Loader peut « télécharger des fichiers » de divers types. De même, le connecteur NetSuite de Boomi (pour SOAP/REST) prend en charge les pièces jointes en masse. Ces outils iPaaS gèrent souvent le découpage en interne afin que l'utilisateur puisse confier un gros fichier et laisser l'outil le diviser à la volée.

En résumé, la perspective de l'industrie est claire : **la limite de 10 Mo de NetSuite est bien connue, bien documentée et largement considérée comme une contrainte à contourner.** Les auteurs de formations et de blogs NetSuite consacrent souvent des sections à la « gestion des gros CSV » ou à la « gestion des importations de gros fichiers » car c'est un problème très courant. Les solutions communautaires vont des plus ingénieuses (utilisation d'AWS Lambda, streaming dans SuiteScript) aux plus pratiques (SuiteApps ou iPaaS). Il n'y a pas de réponse unique parfaite, c'est pourquoi cette étude des modèles est utile. Chaque organisation doit choisir le(s) modèle(s) qui correspond(ent) à son type de contenu, à son expertise technique et à ses contraintes de coût.

Discussion et recommandations

Comprendre les compromis de chaque approche est essentiel pour les architectes :

- **Coût vs effort :** Le streaming natif (`appendLine` /itérateur) est effectivement gratuit à utiliser (fait partie de SuiteScript), mais nécessite du temps de développement et une gouvernance des scripts. Le stockage externe ou les SuiteApps impliquent souvent des frais de fournisseur ou des coûts d'infrastructure, mais une charge de développement minimale.

- **Maintenance :** Les solutions qui bouclent constamment, font appel à AWS ou implémentent des RESTlets personnalisés nécessitent plus de maintenance continue et un débogage potentiel. Une SuiteApp testée peut être plus simple à prendre en charge.

- **Performance :** Le streaming de gros fichiers dans SuiteScript peut consommer des unités de temps d'exécution. L'utilisation de Map/Reduce peut distribuer la charge, mais le développeur doit gérer la concurrence. Les exportations de recherche en masse sont relativement rapides mais limitées en format. Les transferts externes (S3) déportent le travail lourd en dehors de NetSuite, améliorant les performances côté NetSuite mais ajoutant une latence réseau.

- **Sécurité :** Le stockage de fichiers en externe nécessite de s'assurer que les données sont sécurisées en transit et au repos. Les entreprises traitant des données sensibles peuvent ne pas vouloir les stocker dans un stockage cloud public sans chiffrement ou contrôles.

- **Expérience utilisateur :** Du point de vue de l'utilisateur final, l'objectif est souvent : « Je peux télécharger ou voir le fichier normalement ». Le streaming via le File Cabinet est transparent. Les SuiteApps comme SkyDoc essaient également d'être transparentes. Les approches basées uniquement sur des liens sont les pires pour les utilisateurs.

Perspectives d'avenir : NetSuite n'a pas annoncé de plans pour augmenter la limite en mémoire au-delà de 10 Mo. La nature de SuiteScript (un JavaScript exécuté sur serveur) pourrait limiter cela de façon permanente. Au lieu de cela, NetSuite semble miser sur les API de streaming et encourager l'utilisation d'intégrations cloud externes. Par exemple, le travail récent pour connecter NetSuite Analytics Warehouse à Oracle Object Storage montre une tendance à plus d'interopérabilité cloud (Source: docs.oracle.com). Nous pourrions nous attendre à des intégrations natives plus poussées dans les versions à venir (par exemple, prise en charge directe des fichiers Oracle Cloud). SuiteScript continue d'évoluer (par exemple, les promesses HTTPS en 2.1 pour les appels asynchrones), mais la règle des 10 Mo elle-même est restée statique pendant des années.

En pratique, les organisations doivent la planifier comme une considération de conception permanente. Les architectes doivent profiler quels fichiers approchent ou dépassent les 10 Mo et appliquer les modèles appropriés. Pour les rapports CSV, utilisez le streaming ou les recherches. Pour les PDF/Images, utilisez des interfaces utilisateur externes ou des SuiteApps. Documenter le modèle choisi est crucial pour les futurs mainteneurs.

Conclusion

La limite de taille de fichier SuiteScript de 10 Mo de NetSuite est une contrainte ferme et bien documentée découlant de la gestion des fichiers en mémoire de la plateforme (Source: docs.oracle.com) (Source: docs.oracle.com). Bien qu'elle reflète une ingénierie sensée pour un ERP cloud multi-locataire, elle pose un réel défi pour les cas d'utilisation modernes avec de gros fichiers de données. Heureusement, NetSuite fournit des API de streaming de fichiers plats qui permettent aux scripts de contourner la limite pour les données texte/CSV (Source: docs.oracle.com) (Source: netsuitedocumentation1.gitlab.io). Au-delà, les développeurs doivent recourir au partitionnement, à des scripts sophistiqués (Map/Reduce ou tâches), ou à des solutions externes pour ingérer et gérer les gros fichiers.

Ce rapport a collecté et analysé les principales solutions, en les étayant par des documents officiels, des questions-réponses communautaires et des exemples de cas. Le tableau 2 fournit une référence rapide à ces modèles et à leurs compromis. En fin de compte, chaque organisation doit peser quelle approche correspond à ses capacités techniques et à son flux de travail. Par exemple, les boutiques d'intégration de données lourdes peuvent automatiser des RESTlets découpés ou des flux iPaaS, tandis que les petites entreprises peuvent opter pour une SuiteApp comme SkyDoc.

À l'avenir, la tendance vers l'intégration dans le cloud suggère que les limites de gestion des fichiers dans SuiteScript pourraient s'estomper davantage. Le recours au stockage tiers et aux intergiciels (middleware) deviendra de plus en plus courant à mesure que les volumes de données augmenteront. Jusqu'à ce que (et si) NetSuite apporte un changement fondamental à son API de fichiers, la limite de 10 Mo reste une constante de conception — une contrainte que les développeurs avisés doivent continuer à contourner par le streaming, le fractionnement ou une architecture intelligente. Dans tous les cas, la prise en compte de cette limite et une planification minutieuse peuvent garantir le traitement réussi de fichiers volumineux dans NetSuite sans défaillances inattendues des scripts (Source: stackoverflow.com) (Source: archive.netsuiteprofessionals.com).

Sources : La documentation faisant autorité (NetSuite SuiteCloud Help) et les analyses sectorielles (données Egnyte) sont citées tout au long de ce rapport. Les références clés incluent les pages officielles de l'API NetSuite (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com), les notes de version et articles de blog concernant les fonctionnalités de streaming (Source: netsuitedocumentation1.gitlab.io) (Source: docs.oracle.com), ainsi que les connaissances communautaires issues de StackOverflow et des forums NetSuite (Source: stackoverflow.com) (Source: stackoverflow.com) (Source: www.houseblend.io). Des études de cas et des rapports (par exemple, SkyDoc/Stairlock) illustrent l'impact réel (Source: www.79consulting.com) (Source: betanews.com). Chaque affirmation ci-dessus est étayée par ces sources.

Étiquettes: netsuite, suitescript, limite-fichier-10mo, filecreate, telechargement-fichiers-volumineux, streaming-fichiers, suitescript-2x

---