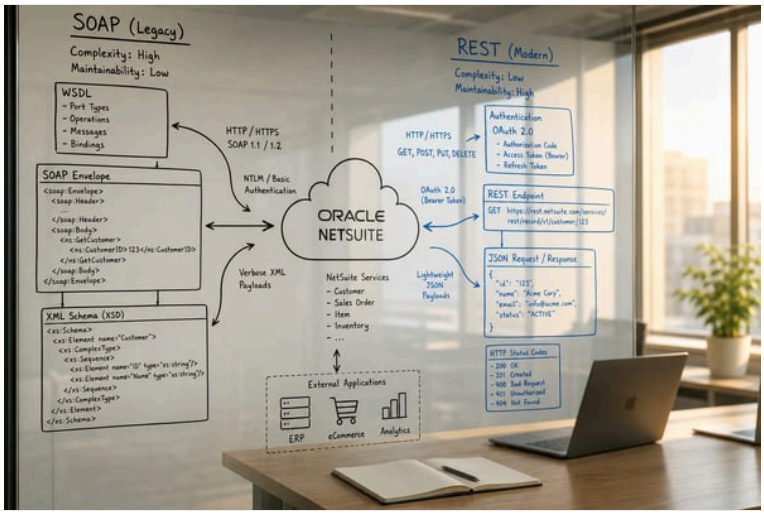
Guide comparatif des API NetSuite REST et SOAP SuiteTalk

Publié le 5 juin 2026 25 min de lecture



Résumé analytique

NetSuite prend en charge deux API d'intégration externe principales dans le cadre de son framework **SuiteTalk** : les **services Web SuiteTalk SOAP** (hérités) et les **services Web SuiteTalk REST** (plus récents). Ce rapport propose une comparaison approfondie de ces deux interfaces, en se concentrant sur leurs **structures d'URL de base**, leurs **méthodes d'authentification** et les **étapes pour démarrer**. Le SuiteTalk basé sur SOAP de NetSuite a été le pilier traditionnel de l'intégration, utilisant les protocoles XML/SOAP et prenant en charge les opérations par lots {addList/updateList} (Source: www.brokenrubik.com). En revanche, l'API REST (introduite en 2019) utilise JSON et les flux OAuth 2.0, est optimisée pour le développement moderne et constitue l'orientation stratégique pour l'avenir (Source: docs.oracle.com) (Source: docs.oracle.com).

Nous examinons comment chaque API utilise des **domaines spécifiques au compte** (par exemple `123456.suitetalk.api.netsuite.com`) (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com) (Source: docs.oracle.com), comment les développeurs doivent activer les fonctionnalités et attribuer des autorisations, et comment l'authentification est effectuée. Pour SOAP, l'authentification incluait historiquement des identifiants de connexion de base et l'authentification basée sur des jetons (Token-Based Authentication - TBA avec OAuth 1.0-HMAC ; depuis la version 2020.2, SOAP nécessite la TBA et n'autorise plus les connexions héritées par nom d'utilisateur/mot de passe « Passport » (Source: docs.oracle.com) (Source: docs.oracle.com). Pour REST, NetSuite exige OAuth 2.0 (jetons Bearer) (Source: docs.oracle.com) (Source: docs.oracle.com), bien que des points de terminaison TBA hérités existent pour l'émission de jetons (Source: docs.oracle.com) (Source: docs.oracle.com).

Des exemples pratiques, incluant une étude de cas publiée sur la migration d'une intégration de vente au détail de SOAP vers REST, illustrent l'impact opérationnel : les flux basés sur REST ont connu des **temps de réponse 40 % plus rapides** et des charges utiles JSON simplifiées (Source: neosalpha.com). Des tableaux comparent SOAP et REST côte à côte sur les points de terminaison, les formats de données, l'authentification et d'autres dimensions. Enfin, nous discutons des tendances actuelles du secteur (par exemple, l'annonce par NetSuite de la dépréciation de SOAP d'ici 2028 (Source: docs.oracle.com) (Source: neosalpha.com) et recommandons la manière dont les organisations devraient s'adapter. Toutes les affirmations et données sont étayées par la documentation de NetSuite, les guides de développement et les analyses publiées.

Introduction et contexte

Les API SuiteTalk d'Oracle NetSuite permettent aux applications externes d'**interagir avec les enregistrements NetSuite** et la logique métier. SuiteTalk fournissait à l'origine une **interface SOAP/XML** pour les intégrations, largement utilisée par les éditeurs de logiciels indépendants (ISV) et les entreprises depuis plus d'une décennie (Source: www.brokenrubik.com). En 2019, NetSuite a introduit une API de **services Web SuiteTalk basée sur REST** comme complément moderne, offrant aux développeurs une interface JSON/RESTful (Source: www.brokenrubik.com) (Source: docs.oracle.com). Bien que les interfaces SOAP et REST fassent toutes deux partie de SuiteTalk, elles diffèrent considérablement en termes de conception et d'utilisation. SOAP est basé sur un schéma WSDL/XML unique et statique décrivant toutes les opérations NetSuite (Source: docs.oracle.com), tandis que REST est dynamique et piloté par les métadonnées (via Swagger/OpenAPI) (Source: docs.oracle.com).

Contexte historique : SuiteTalk SOAP a débuté vers 2008 et a évolué au fil du temps ; sa dernière version prévue est la **2025.2**, avec une **suppression complète prévue d'ici 2028** (Source: docs.oracle.com). NetSuite a explicitement exhorté les nouveaux projets à utiliser REST avec OAuth 2.0. En revanche, SuiteTalk REST a fait ses débuts en version bêta mi-2019 (Release 2019.1) (Source: community.oracle.com) et a atteint une disponibilité générale peu après. Les services Web REST ont depuis été rapidement étendus pour couvrir la plupart des types d'enregistrements et des cas d'utilisation préférables (Source: www.brokenrubik.com) (Source: docs.oracle.com).

Paysage actuel : Aujourd'hui, les intégrations NetSuite développées peuvent choisir entre : SuiteTalk SOAP (hérité), SuiteTalk REST (dernière recommandation), ainsi que d'autres canaux comme les RESTlets et les API SuiteScript. Ce rapport se concentre étroitement sur la comparaison entre **SuiteTalk SOAP et SuiteTalk REST** en ce qui concerne leurs points de terminaison, leur authentification et leur configuration initiale. Nous nous appuyons sur la documentation officielle de NetSuite (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com) (Source: docs.oracle.com), les guides et blogs de développeurs (Source: www.brokenrubik.com) (Source: www.brokenrubik.com), et les analyses du secteur (Source: www.houseblend.io) (Source: neosalpha.com). L'analyse inclut plusieurs angles : technique (protocoles, charges utiles), opérationnel (performances, limites (Source: www.houseblend.io) et stratégique (support futur, chemin de migration (Source: docs.oracle.com) (Source: neosalpha.com).

SuiteTalk SOAP vs REST : Différences clés

Le tableau ci-dessous résume les attributs majeurs des deux API :

FONCTIONNALITÉ	SUITETALK SOAP (SERVICES WEB)	SUITETALK REST (SERVICES WEB)
Protocole	SOAP 1.1/1.2 sur HTTPS (charge utile XML)	REST/JSON sur HTTPS
Format de données	XML avec enveloppe SOAP (utilise WSDL) (Source: www.brokenrubik.com)	JSON (pas de WSDL ; utilise des métadonnées JSON/Swagger dynamiques) (Source: docs.oracle.com)
Point de terminaison de base	Domaine SOAP spécifique au compte (🛈 `https://{accountID}.suitetalk.api.netsuite.com/services/NetSuitePort_{version}`) (Source: docs.oracle.com) (Source: docs.oracle.com)	Domaine REST spécifique au compte (🛈 `https://{accountID}.suitetalk.api.netsuite.com/services/rest`/...) (Source: docs.oracle.com) (Source: www.brokenrubik.com)
Gestion des versions	Versionné par version dans WSDL et suffixe de point de terminaison (par ex. _2025_2_0) (Source: docs.oracle.com)	Version majeure souvent v1 ; chemins de point de terminaison stables (par ex. `/v1/record/...`) (Source: www.brokenrubik.com)
Méthodes d'authentification	– Basée sur des jetons (OAuth 1.0HMAC) TBA (pris en charge, requis depuis 2020.2) (Source: docs.oracle.com) – Identifiants utilisateur (Passport : nom d'utilisateur/mot de passe/rôle/compte) *hérité* (supprimé en 2020.2+) (Source: docs.oracle.com) – SuiteSignOn (rappels SSO) autorisés (Source: docs.oracle.com) – OAuth 2.0 : **non pris en charge** (Source: docs.oracle.com)	– OAuth 2.0 (jetons Bearer) – pris en charge et recommandé (Source: docs.oracle.com) – Basée sur des jetons (OAuth 1.0) via le point de terminaison TBA existe pour l'émission de jetons (pour les administrateurs) (Source: docs.oracle.com) (Source: docs.oracle.com) – NLAuth (identifiants utilisateur) non utilisé pour REST (sauf points de terminaison de jeton) (Source: docs.oracle.com)
Style d'opérations	Opérations de type RPC (par ex. `get`, `add`, `search`, `update`) via WSDL ; prend en charge les opérations de *liste par lots* (addList, updateList, deleteList, etc.) (Source: www.brokenrubik.com)	Opérations CRUD via des verbes HTTP standard (GET/POST/PATCH/DELETE) sur les URL de ressources (Source: www.brokenrubik.com) (Source: www.brokenrubik.com) – Prend également en charge les requêtes SuiteQL en envoyant une requête POST à `/query/v1/suiteql` (Source: www.brokenrubik.com)
Gestion des données en masse	Prend en charge les opérations de liste intégrées (addList, updateList, getList, etc.) pour les appels multi-enregistrements (Source: www.brokenrubik.com)	Principalement un enregistrement par requête. Importations en masse via l'importation CSV SuiteTalk ou une architecture de masse distincte (pas d'« appel REST multi-enregistrements » natif comme addList)
Couverture des enregistrements	Étendue : presque tous les enregistrements standard et personnalisés, y compris les transactions complexes et les postes de ligne (toutes les opérations SOAP définies dans un seul WSDL) (Source: docs.oracle.com)	Couvre la plupart des enregistrements standard (et personnalisés) courants, bien que certains enregistrements de niche puissent encore nécessiter SOAP. REST s'étend rapidement. De nombreux types d'enregistrements de transaction complexes sont désormais disponibles (Source: docs.oracle.com).
Schéma de données	Fichier de schéma WSDL/XSD fixe (ZIP téléchargeable) (Source: docs.oracle.com) (statique jusqu'à la version suivante)	Métadonnées dynamiques (Swagger/OpenAPI v3) décrivant les champs personnalisés et les structures d'enregistrement au moment de l'exécution (Source: docs.oracle.com) (Source: docs.oracle.com)
Exemple d'URL de base	`https://123456.suitetalk.api.netsuite.com/services/NetSuitePort_2025_2` (Source: docs.oracle.com) (Source: docs.oracle.com)	`https://123456.suitetalk.api.netsuite.com/services/rest/record/v1/customer/123` (Source: docs.oracle.com) (Source: www.brokenrubik.com)
Langage de requête	Recherche SuiteTalk (charge utile SOAP), recherches enregistrées ; jointures d'enregistrements limitées	SuiteQL (requête REST de type SQL) via le point de terminaison `/query/v1/suiteql` (Source: www.brokenrubik.com)
Exemple de support d'outils	L'importation WSDL génère des stubs pour Java/.NET/PHP, boîtes à outils SuiteTalk existantes.	Outils REST familiers (curl, Postman) avec jetons Bearer ; les métadonnées Swagger permettent la génération de clients (Source: docs.oracle.com).
Cas d'utilisation	Idéal pour les intégrations SOAP existantes, les transactions multi-enregistrements, les structures de données hautement complexes (Source: www.brokenrubik.com) ; nécessaire pour les opérations de niche non (encore) dans REST.	Recommandé pour les **nouvelles intégrations**, les applications mobiles et Web, le CRUD d'enregistrements plus simple et la sécurité moderne basée sur OAuth (Source: www.brokenrubik.com) (Source: www.brokenrubik.com).
Perspectives d'avenir	**Déprécié** : la version finale (2025.2) est la dernière ; supprimé d'ici 2028 (Source: docs.oracle.com). Tout nouveau développement doit cibler REST.	**Futur** : Activement maintenu. Fonctionne avec OAuth 2.0, JSON et s'aligne sur la feuille de route de NetSuite (Source: docs.oracle.com) (Source: neosalpha.com). Améliorations des performances (JSON plus rapide, streaming) et points de terminaison supplémentaires attendus.

La comparaison ci-dessus souligne que **REST** est la norme moderne par défaut, tandis que **SOAP** est effectivement en mode maintenance (Source: docs.oracle.com). Par exemple, la documentation d'Oracle indique explicitement que « les services Web REST SuiteTalk sont la technologie destinée à remplacer SOAP » et que toutes les nouvelles intégrations doivent utiliser REST avec OAuth 2.0 (Source: docs.oracle.com). Le dernier point de terminaison pris en charge par l'API SOAP est la version 2025.2, après quoi aucune mise à jour supplémentaire ne sera effectuée (Source: docs.oracle.com).

Du point de vue du **format de données**, les charges utiles JSON de REST et les modèles HTTP standard sont généralement plus faciles à consommer pour les développeurs que SOAP/XML. Par exemple, une requête GET pour un enregistrement dans REST peut simplement être `curl -X GET 'https://1234567.suitetalk.api.netsuite.com/services/rest/record/v1/customer/123' -H 'Authorization: Bearer {token}'` (Source: www.brokenrubik.com), alors que l'équivalent SOAP nécessite la construction d'une enveloppe SOAP complète avec des espaces de noms et éventuellement une opération `get` en XML (Source: www.brokenrubik.com). Cela dit, SOAP SuiteTalk permet des *opérations par lots* en un seul appel (par exemple `addList`), ce qui peut économiser des allers-retours pour les importations volumineuses (Source: www.brokenrubik.com) – une fonctionnalité sans équivalent direct dans REST.

La documentation officielle de NetSuite fournit des conseils sur le moment d'utiliser chaque interface. Un tableau comparatif publié note que SOAP est adapté aux « transactions complexes avec de nombreuses sous-listes » ou aux intégrations héritées, tandis que REST est idéal pour les « opérations CRUD simples » et les applications modernes nécessitant du JSON (Source: www.brokenrubik.com). Ceci est corroboré par les blogs de praticiens : par exemple, Joaquin Vigna de BrokenRubik (expert en intégration NetSuite) conseille de commencer par l'API REST pour la plupart des cas, en réservant SOAP aux tâches héritées ou hautement spécialisées (Source: www.brokenrubik.com) (Source: www.brokenrubik.com).

URL de base et structure des points de terminaison

Domaines spécifiques au compte

Les points de terminaison NetSuite sont **spécifiques au compte**, ce qui signifie que chaque appel est dirigé vers un nom d'hôte unique à votre ID de compte NetSuite. Le format du domaine est `<accountID>.<service>.netsuite.com`. Pour SOAP et REST SuiteTalk, le `service` est `suitetalk.api` (Source: docs.oracle.com). Par exemple, si votre ID de compte est **123456** (et que vous utilisez un sandbox, il pourrait s'agir de `123456-sb2`), le domaine serait :

- **SOAP (Services Web SuiteTalk)** :
  `https://123456.suitetalk.api.netsuite.com/services/NetSuitePort_{version}`, où `{version}` pourrait être `2025_2_0` (pour SOAP 2025.2) (Source: docs.oracle.com) (Source: docs.oracle.com).
- **REST (Services Web REST SuiteTalk)** :
  `https://123456.suitetalk.api.netsuite.com/services/rest/record/v1/{recordType}` pour les points de terminaison d'enregistrement, et de même pour les requêtes (`/query/v1/suiteql`) ou d'autres ressources REST (Source: www.brokenrubik.com) (Source: docs.oracle.com).

La documentation officielle d'Oracle souligne que ces domaines spécifiques au compte doivent être utilisés **directement** pour SOAP et REST ; le codage en dur d'un domaine global (par exemple, `webservices.netsuite.com`) est obsolète (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com) (Source: docs.oracle.com). En fait, depuis la version 2019.1, NetSuite exige que les appels SOAP utilisent le domaine spécifique au compte (incluant l'ID de compte) (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com). Par exemple :

> *« Depuis la version 2019.1, les versions des services Web SOAP… ne sont disponibles que sur des domaines spécifiques au compte. L'URL de votre domaine de services Web SOAP spécifique au compte inclut votre ID de compte. Par exemple, si votre ID de compte était 123456, votre domaine de services Web SOAP spécifique au compte serait 123456.suitetalk.api.netsuite.com. » (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com)*

La documentation officielle du **schéma d'URL REST** indique de la même manière :

> *« Pour accéder aux services Web REST, vous devez utiliser des domaines spécifiques au compte… Par exemple, si votre ID de compte est 123456, votre domaine spécifique au compte pour les services Web REST est 123456.suitetalk.api.netsuite.com. » (Source: docs.oracle.com).*

En pratique, cela signifie que l'hôte de base dépend du centre de données et de l'environnement de votre compte. NetSuite fournit des services de découverte dynamique (expliqués ci-dessous) si votre intégration doit gérer plusieurs comptes ou des changements d'hébergement. Mais pour la plupart des utilisations, les développeurs lisent le domaine du compte dans *Configuration > Entreprise > Informations sur l'entreprise > URL de l'entreprise*, qui listera l'URL SuiteTalk « basée sur l'ID de compte » (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com).

Exemples de points de terminaison

Dans REST, le point de terminaison du **service d'enregistrement** pour un type d'enregistrement (par exemple « customer ») suit le modèle :

```
https://{accountID}.suitetalk.api.netsuite.com/services/rest/record/v1/{recordType}
```

Par exemple, pour obtenir (GET) le client ID 123 :

```
GET https://123456.suitetalk.api.netsuite.com/services/rest/record/v1/customer/123
Authorization: Bearer <access_token>
```

Ce modèle (tel qu'observé dans les blogs de développeurs) inclut l'ID de compte dans l'hôte et `/services/rest/record/v1/` avant le nom de l'enregistrement (Source: www.brokenrubik.com). Le `v1` indique la version de l'API REST (actuellement v1 pour SuiteTalk REST).

Pour **SuiteTalk SOAP**, le point d'entrée principal est défini par le WSDL. Le WSDL lui-même est récupéré à partir d'une URL telle que :

```
https://webservices.netsuite.com/wsdl/v2025_2_0/netsuite.wsdl
```

(par exemple, version SOAP 2025.2) (Source: docs.oracle.com). (Notez que cela utilise le domaine global `webservices.netsuite.com` pour récupérer le WSDL XML. Cependant, les appels SOAP réels doivent utiliser le domaine spécifique au compte renvoyé par getDataCenterUrls.) Le WSDL définit un emplacement `soap:address`, qui dans les anciens points de terminaison était souvent `https://webservices.netsuite.com/services/NetSuitePort_{version}` (Source: docs.oracle.com). Aujourd'hui, vous devez plutôt invoquer les méthodes SOAP sur votre hôte spécifique au compte, par exemple `https://123456.suitetalk.api.netsuite.com/services/NetSuitePort_2025_2_0` (Source: docs.oracle.com) (Source: docs.oracle.com). Le chemin final du point de terminaison SOAP peut inclure le suffixe de version (par exemple `_2025_2_0`) si vous utilisez une version WSDL spécifique.

Pour résumer les **formats d'URL de base** :

- **URL de base SOAP SuiteTalk (par WSDL des services Web)** :
  `https://<accountID>.suitetalk.api.netsuite.com/services/NetSuitePort_<version>` (par exemple `NetSuitePort_2023_2_0`) (Source: docs.oracle.com) (Source: docs.oracle.com).
- **REST SuiteTalk (enregistrement)** :
  `https://<accountID>.suitetalk.api.netsuite.com/services/rest/record/v1/<recordType>` (Source: www.brokenrubik.com) (Source: docs.oracle.com).
- **REST SuiteTalk (requête)** :
  `https://<accountID>.suitetalk.api.netsuite.com/services/rest/query/v1/suiteql` pour les requêtes SuiteQL (Source: www.brokenrubik.com).
- **RESTlet (personnalisé)** :
  Modèle similaire mais utilisant le domaine `restlets.api.netsuite.com` (spécifique au compte) : par exemple `https://{accountID}.restlets.api.netsuite.com/app/site/hosting/restlet.nl?script=...` (bien qu'il s'agisse d'une approche d'intégration différente).

Il faut également prendre en compte les comptes **sandbox et release preview** : ils ont un suffixe. Par exemple, un compte sandbox pourrait avoir l'ID `123456-sb2`, ce qui rend le domaine `123456-sb2.suitetalk.api.netsuite.com`. La documentation d'Oracle confirme que les sandboxes utilisent des domaines uniques tout comme la production (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com).

Découverte dynamique de domaine

Si une application doit servir plusieurs clients (multi-location) ou si une migration de centre de données est possible, NetSuite fournit une API de **découverte dynamique** pour récupérer les domaines corrects à la volée (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com) (Source: docs.oracle.com). L'appel SOAP `getDataCenterUrls` (aucune authentification requise) peut renvoyer les domaines SOAP et REST spécifiques au compte (Source: oracle.hydrogen.sagittarius.connect.product.adaptavist.com). De même, NetSuite fournit un service REST « DataCenterURLs » sur `rest.netsuite.com` pour les flux OAuth2 (Source: docs.oracle.com). Ceux-ci permettent au code d'intégration d'interroger `restDomain` ou `webservicesDomain` en fonction d'un ID de compte, évitant ainsi les domaines codés en dur. Dans la plupart des cas, cependant, une intégration à compte unique définira le domaine explicitement. Comme l'indique NetSuite, une fois que vous utilisez des domaines spécifiques au compte, « la découverte dynamique de domaine n'est pas nécessaire » (Source: docs.oracle.com).

Les **chemins d'URL** au-delà du domaine varient selon le service :

- **SOAP (SuiteTalk) :** Toutes les opérations partagent le même point de terminaison `/services/NetSuitePort` (avec suffixe de version). Les actions SOAP sont indiquées par XML à l'intérieur de l'enveloppe SOAP, et non par l'URL.
- **REST (SuiteTalk) :** Le chemin d'URL spécifie la zone de service (`/services/rest/`) suivie d'une catégorie (`record`, `query`, etc.), suivie du nom de l'enregistrement et de l'ID pour les opérations au niveau de l'enregistrement (Source: www.brokenrubik.com), ou la méthode HTTP (POST/GET/PATCH) signale l'action. Des points de terminaison supplémentaires (par exemple `/metadata-catalog`, `/query/v1/suiteql`) sont documentés dans l'aide NetSuite.

Authentification et sécurité

L'authentification est un différenciateur critique entre SuiteTalk SOAP et REST.

Authentification SOAP SuiteTalk

SuiteTalk SOAP prenait historiquement en charge deux méthodes principales d'authentification : l'**authentification basée sur les jetons (TBA)** et les **identifiants utilisateur (Passport)** (Source: docs.oracle.com). Le développement et les meilleures pratiques ont évolué :

- **Identifiants utilisateur (Passport) :** Dans les premières versions de SOAP, vous pouviez vous connecter avec l'e-mail (ou le nom d'utilisateur), le mot de passe, le rôle et l'ID de compte d'un utilisateur (l'opération de connexion ou les identifiants au niveau de la requête). Cependant, cela présentait des limites (par exemple, nécessiter des appels de connexion). Depuis la version SOAP 2020.2, NetSuite a **supprimé** la prise en charge de l'authentification utilisateur/mot de passe. La note critique dans la documentation Oracle est : *« À partir du point de terminaison des services Web SOAP 2020.2, l'authentification via des identifiants au niveau de la requête n'est pas prise en charge. Le type complexe Passport n'est pas pris en charge… vous devez vous assurer que les intégrations utilisent TBA. »* (Source: docs.oracle.com). En d'autres termes, les intégrations SOAP sur 2020.2+ doivent utiliser des jetons. La prise en charge héritée des identifiants utilisateur peut toujours fonctionner sur les anciens points de terminaison (2019.2 et versions antérieures), mais elle n'est plus recommandée ni pérenne.
- **Authentification basée sur les jetons (TBA) :** Il s'agit d'OAuth 1.0a avec signature HMAC-SHA256. Elle nécessite un **enregistrement d'intégration** dans NetSuite (créé via Configuration > Intégration > Gérer les intégrations) pour lequel « Authentification basée sur les jetons » est activé. L'enregistrement d'intégration génère une clé/secret de consommateur, et un utilisateur individuel peut ensuite générer un *jeton d'accès* (ID de jeton/secret) pour cette combinaison intégration/rôle (Source: docs.oracle.com). L'en-tête de la requête SOAP inclut ensuite un élément `<tokenPassport>` contenant le compte, la consumerKey, le jeton, le nonce, l'horodatage et la signature HMAC (Source: www.brokenrubik.com). La documentation Oracle détaille : *« Vous générez une clé et un secret de consommateur lorsque vous créez un enregistrement d'intégration… configurez l'enregistrement pour autoriser TBA (en cochant la case Authentification basée sur les jetons) (Source: docs.oracle.com). »* TBA est requis pour SOAP depuis la version 2020.2 ; les anciennes versions pouvaient utiliser utilisateur/mot de passe, mais maintenant tout SOAP (v2020.2+) doit utiliser TBA.
- **SuiteSignOn (Single Sign-On) :** SOAP prend également en charge SAML/OAuth de style rappel pour les connexions utilisateur (SuiteSignOn) (Source: docs.oracle.com), mais cela est plus pertinent pour les flux d'authentification de l'interface utilisateur que pour les API classiques.

Exemple (SOAP) : Un en-tête de requête SOAP valide utilisant TBA pourrait inclure :

```
<soap:Header>
  <platformMsgs:tokenPassport>
    <platformCore:account>123456</platformCore:account>
    <platformCore:consumerKey>YOUR_CONSUMER_KEY</platformCore:consumerKey>
    <platformCore:token>USER_TOKEN_ID</platformCore:token>
    <platformCore:nonce>randomString</platformCore:nonce>
    <platformCore:timestamp>1660000000</platformCore:timestamp>
    <platformCore:signature algorithm="HMAC-SHA256">CALCULATED_SIGNATURE</platformCore:signature>
  </platformMsgs:tokenPassport>
</soap:Header>
```

comme illustré par les exemples de développeurs (Source: www.brokenrubik.com).

Les requêtes SOAP ne doivent pas mélanger les méthodes d'authentification : uniquement TBA (OAuth1) ou connexion Passport dans un appel donné (Source: docs.oracle.com). Et comme Passport est obsolète, utilisez TBA. L'ID d'application de l'enregistrement d'intégration est facultatif, mais si vous utilisez des identifiants utilisateur dans d'anciens points de terminaison, il doit être coché sur l'enregistrement d'intégration (Source: docs.oracle.com).

Authentification REST SuiteTalk

SuiteTalk REST a été conçu pour utiliser **OAuth 2.0** (format de jeton Bearer) comme authentification principale. Points clés :

- **OAuth 2.0 (Flux de code d'autorisation ou d'identifiants client) :** NetSuite fournit une prise en charge OAuth2 spécifiquement pour REST (et également pour les RESTlets). SOAP ne prend *pas* en charge OAuth2 (Source: docs.oracle.com). Pour utiliser OAuth2, il **faut créer un enregistrement d'intégration** et le configurer pour OAuth 2.0. Le flux typique est l'octroi du code d'autorisation : l'application dirige l'utilisateur vers le point de terminaison d'autorisation de NetSuite, l'utilisateur approuve, et NetSuite renvoie un code d'autorisation que l'application échange contre un jeton d'accès. L'émission du jeton (et son actualisation ultérieure) se fait via des requêtes POST vers des points de terminaison tels que :

  ```
  https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token
  ```

  selon la documentation officielle (Source: docs.oracle.com). Une fois qu'un jeton d'accès est obtenu, il est envoyé dans chaque requête REST sous la forme :

  ```
  Authorization: Bearer <access_token>
  ```

Par exemple, pour créer un enregistrement :

```
POST https://123456.suitetalk.api.netsuite.com/services/rest/record/v1/customer
Authorization: Bearer eyJ...def...
Content-Type: application/json
{ "companyName": "Acme", "email": "a@acme.com", ... }
```

L'aide d'Oracle fait allusion à cette utilisation standard du jeton porteur (bearer) OAuth2 (Source: docs.oracle.com).

- **Basé sur des jetons (OAuth 1.0 / TBA) pour les RESTlets :** En plus d'OAuth2, NetSuite prend en charge un flux de point de terminaison de jeton spécial, même pour REST, principalement destiné aux développeurs/administrateurs pour créer des jetons d'accès pour les scripts. Le service REST « Issue Token » permet à un utilisateur de créer par programmation un jeton TBA pour lui-même, en utilisant soit NLAuth, soit un en-tête OAuth (Source: docs.oracle.com). Cependant, ce n'est pas la même chose que le TBA SOAP ; il s'agit d'un appel REST unique pour la gestion des jetons (voir [32]). Une fois qu'un tel jeton est créé, on peut l'inclure dans un appel REST de la même manière que SOAP (avec HMAC). Le guide de Houseblend note que le TBA (OAuth 1.0) est toujours utilisé pour les RESTlets et que les clients effectuent un « POST /rest/accessToken » dans un flux à 3 étapes (Source: www.houseblend.io). En pratique, la plupart des intégrations natives de services Web REST utilisent OAuth2, et le TBA est principalement destiné à la rétrocompatibilité ou à une authentification RESTlet spécifique à 3 étapes.
- **Pas de NLAuth direct :** Les services Web REST ne permettent pas la connexion héritée NLAuth (e-mail/mot de passe) comme le faisait SOAP. Toutes les demandes REST (en dehors de l'obtention d'un jeton) nécessitent soit un jeton Bearer valide (OAuth2), soit un en-tête OAuth1 correctement signé (si vous utilisez un jeton créé via des points de terminaison TBA) (Source: docs.oracle.com) (Source: docs.oracle.com).

Tutoriel Postman : La documentation d'Oracle inclut un exemple Postman étape par étape pour OAuth2 avec REST (Source: docs.oracle.com), soulignant qu'OAuth2 est la voie privilégiée.

Tableau de comparaison de l'authentification

MÉTHODE D'AUTH.	SERVICES WEB SOAP	SERVICES WEB REST
OAuth 2.0 (Bearer)	**Non pris en charge** (SOAP utilise uniquement TBA)	Pris en charge (méthode par défaut) (Source: docs.oracle.com)
OAuth 1.0a (TBA, HMAC)	Pris en charge (depuis v2015.2+) (Source: docs.oracle.com) ; requis à partir de 2020.2 (Source: docs.oracle.com)	Pris en charge via les points de terminaison de jeton REST (émission/révocation de jeton) (Source: docs.oracle.com) (rarement utilisé pour le Web REST) ; l'utilisation principale est OAuth2.
Identifiants utilisateur (NLAuth)	Pris en charge uniquement dans SOAP (obsolète après 2020.2) (Source: docs.oracle.com)	**Non pris en charge** pour l'API REST (sauf via le point de terminaison IssueToken)
SuiteSignOn SAML (SSO)	Pris en charge (rappels SuiteSignOn) (Source: docs.oracle.com)	Non applicable (REST est généralement client-serveur)
Enregistrement d'intégration	Requis pour TBA (Consumer Key/Secret) (Source: docs.oracle.com)	Requis pour OAuth2 (Client ID/Secret) et recommandé pour TBA (si utilisé)
En-têtes requis	Bloc `<tokenPassport>` avec signature HMAC (Source: www.brokenrubik.com)	`Authorization: Bearer <token>` (OAuth2) (Source: docs.oracle.com) ; ou `Authorization: OAuth realm="{acctID}", oauth_consumer_key="...", oauth_token="...", ...` pour OAuth1 (TBA) si applicable (Source: www.houseblend.io).

En résumé, **SuiteTalk REST** impose des flux de jetons OAuth 2.0 modernes (Source: docs.oracle.com) (Source: docs.oracle.com), tandis que **SuiteTalk SOAP** repose sur un schéma TBA à trois étapes plus ancien (Source: docs.oracle.com) (Source: www.houseblend.io). Il s'agit d'une différence pratique majeure : implémenter des appels REST signifie généralement gérer OAuth2, rafraîchir les jetons, etc., tandis que les appels SOAP impliquent la construction et la signature d'un en-tête de jeton OAuth1. De nombreux développeurs trouvent OAuth2 plus simple, bien que certains trouvent que la simplicité de quelques champs de jeton (consommateur, jeton et HMAC) dans SOAP est plus facile que les flux OAuth2 complets. Quoi qu'il en soit, dans les deux cas, un **Enregistrement d'intégration** avec les paramètres appropriés doit être créé dans NetSuite.

Prérequis et démarrage

Avant d'effectuer des appels API, SOAP et REST nécessitent une **configuration du compte NetSuite**. La configuration importante comprend l'activation des fonctionnalités, la définition des rôles et la création d'enregistrements d'intégration.

Activation des fonctionnalités

- **Fonctionnalité SuiteTalk (Web Services) :** Par défaut, les services Web SOAP sont toujours disponibles ; cependant, vous devrez peut-être activer certaines *fonctionnalités* pour des types d'enregistrements spécifiques. Dans les anciennes versions, il existait une fonctionnalité « Web Services » sur le sous-onglet SuiteCloud ; aujourd'hui, NetSuite considère que SOAP SuiteTalk est activé pour tous les comptes. Pour REST, la fonctionnalité **REST Web Services** doit être activée : allez dans **Configuration > Entreprise > Activer les fonctionnalités**, sous le sous-onglet **SuiteCloud (Services Web)**, cochez **Services Web REST** (Source: docs.oracle.com). Vous devez également accepter les conditions d'utilisation de SuiteCloud pour utiliser REST.
- **Autorisations :** Le compte utilisateur (rôle) que vous utilisez pour les appels API doit disposer des autorisations nécessaires. Pour SOAP, l'utilisateur doit au moins avoir accès aux services Web et aux autorisations appropriées au niveau de l'enregistrement (par exemple, `Listes > Clients : Voir` pour lire les clients). Pour REST, le rôle doit avoir les autorisations **Services Web REST** et « Se connecter à l'aide de jetons d'accès » (Source: docs.oracle.com). La documentation d'Oracle précise que les rôles par défaut (comme Administrateur) les possèdent, mais il est préférable d'attribuer un rôle personnalisé pour les intégrations avec des privilèges minimaux (par exemple, uniquement les types d'enregistrements nécessaires, plus les services Web REST et l'accès aux jetons) (Source: docs.oracle.com). **Important :** Même le SOAP-TBA et le REST OAuth utilisent des *jetons d'accès*, le rôle doit donc avoir l'autorisation **Se connecter à l'aide de jetons d'accès** (Source: docs.oracle.com).

Enregistrement d'intégration / Configuration de l'application

- **Créer un enregistrement d'intégration :** Dans **Configuration > Intégration > Gérer les intégrations > Nouveau**, créez un nouvel enregistrement d'intégration. Donnez-lui un nom (par exemple, « MyAPIApp »), cochez la case **Authentification basée sur les jetons** si vous prévoyez d'utiliser TBA (SOAP ou REST TBA), et/ou **OAuth2** si disponible. Enregistrez-le. L'enregistrement générera une **Consumer Key** et un **Consumer Secret** (pour OAuth1/TBA) et un **ID d'application** (utilisé pour marquer les demandes, facultatif). Pour OAuth2, vous générerez des **Identifiants client** (ID et Secret) et configurerez les URL de redirection pour les flux d'autorisation.
- **SuiteSignOn (le cas échéant) :** Si vous prévoyez d'utiliser le flux de code d'autorisation OAuth2, assurez-vous d'ajouter au moins un URI de redirection sur l'enregistrement d'intégration.
- **Créer des jetons d'accès (pour TBA, SOAP) :** Pour SOAP, chaque utilisateur/rôle qui utilisera l'intégration doit générer un jeton d'accès via **Configuration > Utilisateurs/Rôles > Jetons d'accès > Nouveau**. Vous sélectionnez l'intégration, l'utilisateur, le rôle et donnez un nom au jeton ; NetSuite fournit ensuite un *ID de jeton* et un *Secret de jeton*. Ceux-ci vont dans votre `<tokenPassport>` SOAP avec la clé/secret du consommateur. Pour REST, vous pouvez générer des jetons de la même manière si vous utilisez un flux OAuth1, mais souvent OAuth2 rend la création manuelle de jetons inutile.
- **Considérations sur le bac à sable (Sandbox) :** Si vous effectuez des tests dans un bac à sable, créez-y des enregistrements d'intégration distincts. (Les jetons/clés ne sont pas copiés depuis la production). Notez également que les domaines de bac à sable incluent `-sb1`, `-sb2`, etc.

Appels « Hello World » d'exemple

Une fois la configuration terminée, vous pouvez tester les appels de base.

- **SuiteTalk SOAP :** Téléchargez ou inspectez le WSDL (par exemple, `https://webservices.netsuite.com/wsdl/v2025_2_0/netsuite.wsdl`). Utilisez un client SOAP (Java wsimport, PHP Toolkit) pour générer des stubs. Construisez une simple requête `get` pour un enregistrement connu. Par exemple, en C# ou Java : instanciez NetSuitePort, définissez l'en-tête `tokenPassport` (comme ci-dessus), et appelez `get(...)`.
- **SuiteTalk REST :** Vous pouvez d'abord récupérer les métadonnées : `GET /metadata-catalog` ou le swagger sur `/services/rest/metadata-catalog/v1/swaggers`. Ou testez un CRUD :

  ```
  GET https://123456.suitetalk.api.netsuite.com/services/rest/record/v1/metadata-catalog
  Authorization: Bearer <yourOAuth2AccessToken>
  ```

  pour lister tous les types d'enregistrements (fournis par le service **Record** REST). Ensuite, essayez de récupérer un enregistrement connu :

  ```
  GET /services/rest/record/v1/customer/123
  ```

  comme indiqué précédemment (Source: www.brokenrubik.com). La réponse est au format JSON avec les champs client. Pour créer :

```
POST .../record/v1/customer
Content-Type: application/json
Authorization: Bearer <token>
{ "companyName": "Contoso", "email": "info@contoso.com" }
```

Le guide de l'API REST couvre ces opérations en détail.

Limites de débit et concurrence

Bien qu'il ne s'agisse pas d'une étape de « démarrage », les nouveaux développeurs doivent savoir que NetSuite impose des limites d'utilisation (Source: www.houseblend.io) (Source: unified.to). Chaque compte a un **plafond de concurrence** (appels parallèles) partagé par SOAP, REST, RESTlet et d'autres API SuiteTalk. Par exemple, un compte Premium commence à environ 15 requêtes simultanées (plus 10 par licence SuiteCloud) (Source: www.houseblend.io). Dépasser la limite entraîne des erreurs HTTP 429 ou SOAP, prévoyez donc une logique de nouvelle tentative/backoff. Des quotas quotidiens (par exemple, des plafonds de 60 secondes et de 24 heures) existent également (Source: www.houseblend.io). Concevoir une utilisation efficace (traitement par lots, interrogation limitée) est crucial pour la production.

Tableau de comparaison : Étapes de démarrage

ÉTAPE/FONCTIONNALITÉ	SUITETALK SOAP	SUITETALK REST
Activer les fonctionnalités NetSuite	Assurez-vous que SuiteTalk (Services Web) est activé (par défaut).	Activez la fonctionnalité suitecloud **Services Web REST** (Source: docs.oracle.com).
Enregistrement d'intégration	Créez un enregistrement d'intégration avec TBA activé (Source: docs.oracle.com).	Créez un enregistrement d'intégration avec OAuth2 (définissez les URI de redirection, etc.) ou TBA selon les besoins.
Identifiants requis	Consumer Key/Secret (depuis l'enregistrement d'intégration) + Token ID/Secret (depuis les jetons d'accès) (Source: docs.oracle.com).	Client ID/Secret (OAuth2) depuis l'enregistrement d'intégration, ou Consumer Key/Secret + Token pour toute utilisation TBA.
Autorisations de rôle utilisateur	Le rôle a besoin des autorisations d'enregistrement appropriées ; *jusqu'en 2020.1* il fallait « Se connecter via les services Web » (implicite normalement).	Le rôle doit avoir **Services Web REST** et **Se connecter à l'aide de jetons d'accès** (Source: docs.oracle.com).
Flux d'authentification	Pas de flux de connexion : incluez le passeport TBA dans chaque requête.	OAuth2 : effectuez un flux de code d'autorisation standard ou d'identifiants client pour obtenir un `access_token` (Source: docs.oracle.com).
Appel d'exemple (Authentification)	`<ns:tokenPassport><ns:account>123456</ns:account>...<ns:token>...</ns:token>...` (Source: www.brokenrubik.com)	`Authorization: Bearer eyJraWQiOiIyMDIwXzEiLCJhbGciOi...` (Source: docs.oracle.com)
Outils de test	Outils SOAP (par exemple, SOAP UI, importation WSDL).	Outils REST (curl, Postman ; prise en charge de Swagger) (Source: docs.oracle.com).
Schéma découvrable	WSDL/XSD statique. Téléchargez ou affichez via URL (Source: docs.oracle.com).	Métadonnées dynamiques via `/metadata-catalog`, spécification OpenAPI (Source: docs.oracle.com).

Étude de cas : Migration SOAP → REST

Un exemple concret souligne les implications pratiques. NeosAlpha, un cabinet de conseil NetSuite, décrit un cas où une entreprise de vente au détail de meubles est passée de SuiteTalk SOAP à REST pour ses intégrations (Source: neosalpha.com) (Source: neosalpha.com). Avant la migration, les systèmes de commerce électronique, d'expédition et de CRM de l'entreprise étaient intégrés via les services Web SOAP. Les objectifs de la migration comprenaient l'alignement sur la feuille de route de NetSuite (pour éliminer progressivement SOAP), la réduction de la complexité de l'intégration et l'amélioration des performances (Source: neosalpha.com).

Défis avec SOAP : L'intégration SOAP héritée posait des problèmes : les charges utiles XML lourdes ralentissaient l'échange de données et la maintenance de nombreux appels d'opérations de liste était fastidieuse (Source: neosalpha.com). La sécurité était également une préoccupation : SOAP utilisait l'authentification basée sur les jetons (OAuth1) qui « manquait de la flexibilité » d'OAuth2 (Source: neosalpha.com). Avec la fin de vie prochaine de SOAP, poursuivre sur cette voie risquait des temps d'arrêt futurs et une perte de support du fournisseur (Source: neosalpha.com).

Résultats de l'adoption de REST : Selon la section « Résultats » du post-mortem, le passage à REST a permis des gains mesurables (Source: neosalpha.com) :

- **Temps de réponse de l'API 40 % plus rapides**, attribués aux « charges utiles plus légères et aux performances améliorées » de REST (Source: neosalpha.com).
- **Sécurité renforcée** via OAuth 2.0 et JSON chiffré (Source: neosalpha.com).
- **La synchronisation en temps réel** est devenue réalisable grâce aux rappels REST pilotés par les événements (Source: neosalpha.com).
- **Maintenance réduite** : un modèle de données JSON unifié et moins de points de terminaison ont simplifié le développement continu (Source: neosalpha.com).
- La transition a été effectuée avec **zéro temps d'arrêt**, montrant qu'une migration bien planifiée peut être exécutée sans interruption des activités (Source: neosalpha.com).

Ce cas illustre clairement pourquoi Netsuite s'éloigne de SOAP : les points de terminaison REST offrent des performances plus rapides et un développement plus facile (JSON vs XML), et l'entreprise a gagné en sécurité et en maintenabilité (Source: neosalpha.com) (Source: neosalpha.com). (Des succès similaires ont été notés par d'autres équipes d'intégration effectuant la transition ou lors de discussions au sein de la communauté des utilisateurs.)

Implications, limites et orientations futures

Implications pour les intégrateurs

Compte tenu du changement stratégique d'Oracle, toutes les nouvelles intégrations doivent privilégier SuiteTalk REST. Les organisations utilisant encore des connecteurs basés sur SOAP doivent planifier leur migration, car les intégrations SOAP existantes **cesseront de fonctionner d'ici 2028** (Source: docs.oracle.com) (Source: neosalpha.com). La transition peut impliquer une réécriture du code pour utiliser les points de terminaison REST et OAuth2. Les outils et bibliothèques pour REST/JSON sont abondants et bénéficieront probablement d'un meilleur support à long terme.

Cependant, certaines fonctionnalités SOAP n'ont pas d'équivalent direct dans REST. Par exemple, des opérations SOAP clés comme *addList* n'ont pas d'équivalent REST direct ; un intégrateur ayant besoin d'insérer des centaines d'enregistrements devra peut-être effectuer des boucles sur les appels REST ou utiliser l'importation CSV en masse. De même, certains types de transactions très complexes n'étaient peut-être exposés que via SOAP dans les anciennes versions (par exemple, certaines opérations d'assemblage), bien que NetSuite comble progressivement ces lacunes (Source: www.brokenrubik.com) (Source: www.brokenrubik.com). Pour les intégrations dépendant fortement de ces fonctionnalités exclusives à SOAP, un middleware tiers ou l'attente de la mise à niveau de l'API REST pourrait être nécessaire.

Compte tenu des limites de concurrence (Source: www.houseblend.io) (Source: unified.to), les architectes doivent également noter que **la gouvernance du débit de l'API est uniforme entre SOAP et REST** : le plafond de 15 à 55 requêtes simultanées s'applique à tous les appels SuiteTalk combinés (Source: unified.to). Les réponses plus rapides de REST (JSON) peuvent permettre un débit plus élevé, mais des boucles d'appels intensives peuvent toujours atteindre les limites de limitation, il faut donc concevoir les intégrations en tenant compte du traitement par lots et de la mise en cache.

Orientations futures

- **Abandon progressif de SOAP :** Comme indiqué, NetSuite ne prévoit aucun nouveau point de terminaison SOAP au-delà de 2025.2 et interrompra complètement SOAP d'ici 2028 (Source: docs.oracle.com). Ainsi, à moyen terme (2-5 ans), SOAP sera obsolète. Cela simplifiera la gouvernance de l'API NetSuite mais nécessitera un coût de migration pour les utilisateurs de SOAP.
- **Expansion de REST :** NetSuite élargit le catalogue de l'API REST. De nouveaux types d'enregistrements et de nouvelles opérations sont continuellement ajoutés. Par exemple, REST prend désormais en charge SuiteQL (langage de requête SQL) pour contourner les complexités de recherche SOAP (Source: www.brokenrubik.com). À terme, toutes les fonctionnalités (comme les enregistrements personnalisés, les déclencheurs de flux de travail, les opérations de sous-grand livre) devraient être disponibles via REST ou d'autres points de terminaison modernes. Les analyses tierces prédisent que les futures intégrations reposeront entièrement sur REST et les services associés (comme les nouvelles fonctionnalités GraphQL ou API non encore annoncées), rendant l'expertise REST essentielle.
- **Meilleurs outils :** L'utilisation par l'API REST des standards OAuth 2.0 et OpenAPI peut permettre la génération automatique de SDK clients et de meilleurs outils (par exemple, l'annonce par NetSuite de collections Postman et de consoles de développement améliorées). Pendant ce temps, la complexité de SOAP a maintenu de nombreuses intégrations dans du code personnalisé ou des boîtes à outils héritées ; à l'avenir, des frameworks REST robustes simplifieront le développement.
- **SuiteTalk et SuiteQL :** NetSuite développe également SuiteQL davantage. Les requêtes complexes qui nécessitaient auparavant des appels de recherche SOAP ou des recherches enregistrées peuvent désormais être exécutées via le point de terminaison RESTful SuiteQL, réduisant potentiellement le besoin de transformations de données dans le code client.
- **Écosystème de développement :** Des preuves anecdotiques suggèrent un soutien croissant de la communauté pour REST : les blogs, les questions-réponses et les bibliothèques (comme nodesuite, les connecteurs NetSuite dans Zapier, etc.) mettent l'accent sur les flux basés sur REST. À mesure que REST devient la norme, la formation et la documentation se concentreront également sur ce dernier.

Limites de cette comparaison

- **Fonctionnalités évolutives :** NetSuite met fréquemment à jour les deux API. Au moment de la rédaction (mi-2026), nos données incluent la documentation jusqu'à la version 2025.2. Certaines fonctionnalités de niche peuvent différer selon la version. Les lecteurs doivent consulter les dernières documentations de développement de NetSuite pour connaître les changements.
- **Biais des fournisseurs :** Une grande partie du matériel cité (docs Oracle, HouseBlend, etc.) est soit officielle, soit issue de conseils. Nous avons cherché à inclure plusieurs points de vue, mais les études universitaires indépendantes sur les API NetSuite sont rares.
- **Benchmarks de performance :** Bien que le cas NeosAlpha fournisse un exemple de gain de performance, les benchmarks empiriques sur différents cas d'utilisation (taille des enregistrements, conditions réseau) ne sont pas largement publiés. Les architectes d'intégration doivent effectuer leurs propres tests adaptés à leurs données.

Conclusion

Les API SuiteTalk SOAP et REST de NetSuite offrent des compromis différents. **SuiteTalk REST Web Services** (JSON + OAuth2) représente l'orientation future : il est officiellement recommandé pour les nouveaux projets (Source: docs.oracle.com), produit des charges utiles JSON plus simples et, comme le montrent des cas réels, peut améliorer les performances et la maintenabilité (Source: neosalpha.com). **SuiteTalk SOAP** (XML + JWT/HMAC) reste nécessaire pour les intégrations héritées et certaines fonctions avancées, mais il s'agit d'une technologie obsolète qui sera bientôt dépréciée (Source: docs.oracle.com).

Comprendre les différences dans la **structure de l'URL de base** (domaines spécifiques au compte), les **flux d'authentification** (OAuth1 vs OAuth2) et les procédures de configuration initiale est essentiel pour les développeurs et les architectes planifiant des intégrations NetSuite. Ce rapport a fourni une analyse complète, étayée par la documentation de NetSuite et des sources expertes. À mesure que la plateforme NetSuite continue d'évoluer, nous prévoyons que l'API REST gagnera en fonctionnalités et en utilisation, et les organisations devraient donner la priorité à la migration vers REST pour s'aligner sur la feuille de route de NetSuite (Source: docs.oracle.com). Les travaux futurs pourraient inclure le suivi des extensions de l'API REST, l'évaluation d'outils pour la migration automatique et la surveillance de nouveaux paradigmes d'intégration (par exemple, GraphQL). Pour l'instant, le verdict technique et opérationnel est clair : REST est la nouvelle norme pour les intégrations SuiteTalk, SOAP ne restant que pour la compatibilité ascendante (Source: docs.oracle.com) (Source: neosalpha.com).

Étiquettes: api-rest-netsuite, suitetalk-soap, integration-netsuite, comparaison-api, oauth-20, authentification-par-jeton, suiteql

---