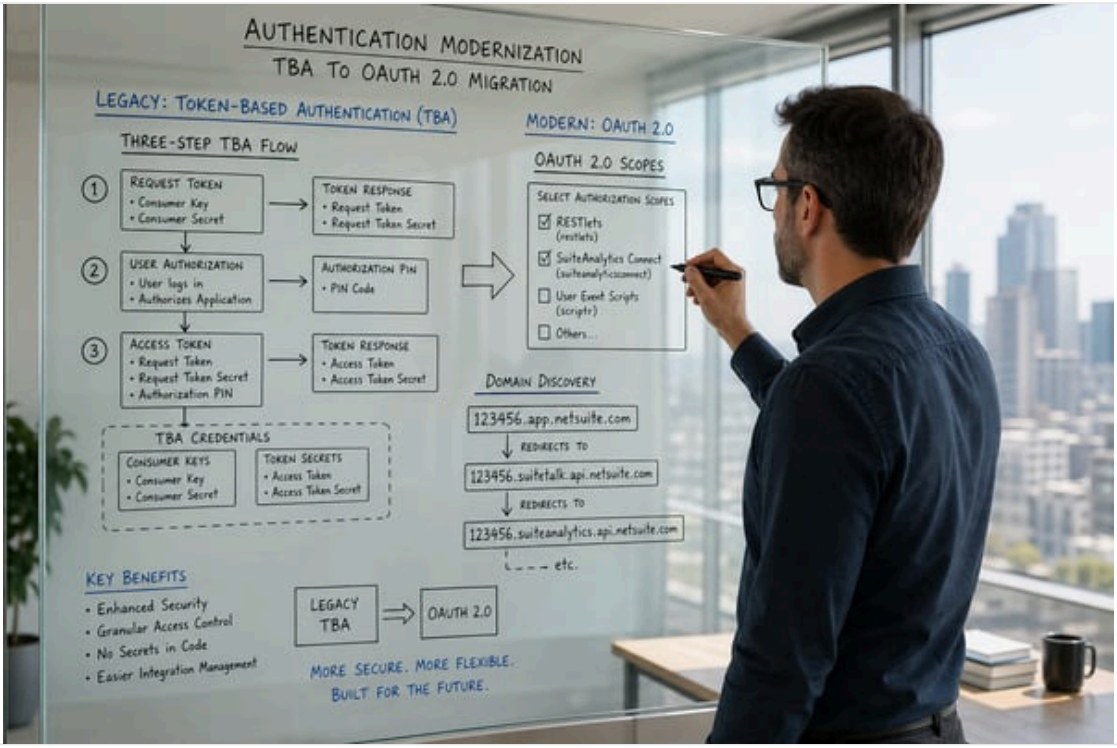
URL des RESTlets NetSuite : domaines, TBA et portées OAuth 2.0

Publié le 25 avril 2026 27 min de lecture



Résumé analytique

Ce rapport fournit une analyse complète de l'utilisation par Oracle NetSuite des **domaines spécifiques au compte** pour les URL de RESTlets et des flux d'authentification pour accéder à ces URL, en se concentrant à la fois sur l'authentification basée sur les jetons (TBA, une méthode basée sur OAuth 1.0) héritée de NetSuite et sur le nouveau framework OAuth 2.0 avec portées (scopes). Chaque compte NetSuite (production, sandbox, release preview) possède un domaine unique spécifique au locataire (par exemple, `123456.app.netsuite.com` pour un compte de production et `123456-sb1.app.netsuite.com` pour une sandbox) (Source: docs.oracle.com) (Source: docs.oracle.com). Ces domaines restent constants même si le compte est migré entre des centres de données (Source: docs.oracle.com), éliminant ainsi le besoin de coder en dur des URL spécifiques au centre de données. Pour découvrir le domaine correct de n'importe quel compte, NetSuite fournit des services de découverte REST (tels que les services DataCenterURLs et REST Roles de `rest.netsuite.com`) ou l'API SuiteScript `N/url.resolveDomain` (Source: docs.oracle.com) (Source: docs.oracle.com).

Sur le plan de la sécurité, l'*authentification basée sur les jetons* (TBA) a été la méthode privilégiée pour les intégrations RESTlet (et SuiteTalk). La TBA utilise OAuth 1.0a avec des signatures HMAC-SHA256 et quatre identifiants (clé/secret du consommateur et ID/secret du jeton) (Source: www.brokenrubik.com) (Source: docs.oracle.com). Dans ce flux en trois étapes, un client externe (1) effectue une requête POST vers le point de terminaison `/rest/requesttoken` pour obtenir un jeton de requête non autorisé, (2) dirige l'utilisateur vers le point de terminaison `/app/login/secure/authorizetoken.nl` pour l'autorisation (identifiants JSO/ SAML SSO et consentement), puis (3) effectue une requête POST vers `/rest/accesstoken` pour échanger le jeton autorisé contre un jeton d'accès et un secret (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com). Une fois obtenu, le jeton d'accès OAuth 1.0 (plus le secret) est inclus en tant qu'en-tête HMAC dans chaque appel RESTlet, par exemple avec `Authorization: OAuth realm="<ACCOUNT_ID>", oauth_consumer_key="...", oauth_token="...", oauth_signature_method="HMAC-SHA256", …` (Source: www.brokenrubik.com) (Source: timdietrich.me). Le guide de Fitzgerald souligne que la TBA « est l'approche recommandée pour les intégrations serveur à serveur » (Source: www.brokenrubik.com) et que seul HMAC-SHA256 est pris en charge (SHA-1 a été retiré depuis la version 2023.1) (Source: docs.oracle.com).

Cependant, Oracle a déclaré que **la TBA est en cours d'abandon**. À partir de NetSuite 2027.1, **aucune nouvelle intégration RESTlet ou service web utilisant la TBA ne sera autorisée** (Source: docs.oracle.com) (Source: www.houseblend.io). Au lieu de cela, OAuth 2.0 est désormais la méthode privilégiée. OAuth 2.0 simplifie le processus (pas de signature par requête) et prend en charge les fonctionnalités de sécurité modernes (jetons porteurs,

jetons de rafraîchissement, chiffrement plus fort, intégration avec la connexion SAML/OIDC, 2FA) (Source: docs.oracle.com) (Source: www.houseblend.io). En particulier, OAuth 2.0 introduit des *portées* (scopes) granulaires dans le modèle d'intégration de NetSuite. En pratique, les portées sont configurées sur l'enregistrement d'intégration en cochant des fonctionnalités telles que **RESTlets**, **REST Web Services**, **SuiteAnalytics Connect**, etc. (Source: docs.oracle.com). Un jeton d'accès OAuth 2.0 ne sera valide que pour les portées activées sur son application (Source: docs.oracle.com) (Source: blogs.oracle.com). Par exemple, un jeton d'accès JWT émis pour un appel de services web REST pourrait avoir `"scope": ["RESTLETS"]` dans sa charge utile (Source: docs.oracle.com), indiquant qu'il accorde l'accès aux RESTlets. (La documentation d'Oracle pour *SuiteProjects Pro* montre de la même manière des valeurs de portée comme `rest`, `soap`, `xml`, `bi` qui peuvent être combinées (Source: docs.oracle.com), par analogie avec les portées de NetSuite.)

Les sections suivantes détaillent les domaines spécifiques au compte NetSuite et les formats d'URL, expliquent les mécanismes du flux TBA, et traitent des flux d'intégration et des portées OAuth 2.0 de NetSuite. Nous comparons les méthodes d'authentification (TBA vs OAuth2) et discutons des implications pour la conception de l'intégration, la sécurité et les orientations futures. Chaque affirmation est étayée par la documentation d'Oracle, des blogs d'experts et des études de cas.

Introduction

Oracle NetSuite est une plateforme ERP cloud utilisée par des dizaines de milliers d'entreprises dans le monde (Source: www.houseblend.io). En tant qu'ERP SaaS multi-locataire, chaque compte (locataire) est identifié par un ID numérique (avec des suffixes pour la sandbox ou la release preview). NetSuite fournit diverses API pour l'intégration, notamment SuiteTalk (SOAP et REST Web Services et des points de terminaison de script SuiteScript (RESTlets et Suitelets). Un **RESTlet** est un SuiteScript personnalisé (SuiteScript 2.x déployé sur un compte NetSuite qui expose des points de terminaison HTTP (GET/POST/PUT/DELETE) à des clients externes (Source: www.brokenrubik.com) (Source: timdietrich.me). Les RESTlets vous permettent d'implémenter une logique métier et un traitement de données arbitraires directement dans NetSuite, en fournissant des réponses JSON ou texte. Ils complètent SuiteTalk REST (l'API REST intégrée basée sur les enregistrements) en permettant des opérations complexes ou spécialisées en SuiteScript complet.

Lors de l'appel de n'importe quel point de terminaison REST de NetSuite (SuiteTalk ou RESTlet) depuis un client externe, l'URL de la requête doit utiliser le *domaine unique du compte*. Historiquement, les URL de NetSuite incluaient des identifiants de centre de données (par exemple *.restlets.api.netsuite.com* vs *.restlets.api.na3.netsuite.com*), ce qui nécessitait des mises à jour si un compte était déplacé entre des centres de données. Aujourd'hui, Oracle utilise des **domaines spécifiques au compte** afin que les URL restent stables lors des migrations (Source: docs.oracle.com). En d'autres termes, le domaine inclut l'ID du compte (et le type de compte) lui-même, et ne change pas lorsque NetSuite rééquilibre la disponibilité des données (Source: docs.oracle.com) (Source: docs.oracle.com). Par exemple, un compte de production avec l'ID `123456` pourrait utiliser des URL comme `https://123456.app.netsuite.com/...` ou `https://123456.suitetalk.api.netsuite.com/...`, tandis qu'un compte sandbox `123456_SB1` utilise `https://123456-sb1.app.netsuite.com/...` (notez que le trait de soulignement est remplacé par un trait d'union et que le suffixe est en minuscules) (Source: docs.oracle.com). L'URL officielle spécifique au compte pour chaque service est répertoriée dans le sous-onglet *Company URLs* de la page Setup > Company > Company Information de NetSuite (Source: docs.oracle.com) (Source: docs.oracle.com). Les administrateurs ne doivent jamais « construire » manuellement ces URL ; utilisez plutôt les valeurs de *Company URLs* ou les API de découverte dynamique (Source: docs.oracle.com) (Source: docs.oracle.com).

Pour gérer plusieurs comptes ou des comptes inconnus de manière dynamique, NetSuite fournit des services de découverte. Pour les clients SOAP (SuiteTalk), l'appel SOAP `getDataCenterUrls` ou le point de terminaison `webservices.netsuite.com` peuvent être utilisés. Pour les clients REST (RESTlets, services web REST, etc.), Oracle fournit un service DataCenterUrls basé sur REST et un *service REST Roles* autonome sur `rest.netsuite.com` (Source: docs.oracle.com). Appeler `https://rest.netsuite.com` avec des identifiants valides renvoie les domaines de base corrects (par exemple, les URL `.suitetalk.api.netsuite.com` ou `.app.netsuite.com`) pour ce compte. De même, dans SuiteScript 2.x, vous pouvez appeler la fonction `resolveDomain({ hostType: url.HostType.RESTLET, accountId: 'ACCT' })` du module `N/url` pour récupérer le domaine RESTlet approprié (Source: docs.oracle.com). Ces mécanismes de découverte dynamique garantissent que les intégrations client utilisent toujours le domaine correct et à jour pour n'importe quel compte NetSuite.

Le **Tableau 1** ci-dessous résume les principales options d'intégration de NetSuite et leurs caractéristiques, illustrant où les RESTlets s'intègrent aux côtés de SuiteTalk SOAP/REST et des Suitelets (www.brokenrubik.com.

Tableau 1 : Comparaison des méthodes d'intégration NetSuite (adapté de [BrokenRubik][4])

FONCTIONNALITÉ	RESTLETS	SERVICES WEB SUITETALK SOAP	SERVICES WEB SUITETALK REST	SUITELETS (SCRIPTS UI)
Logique	SuiteScript complet (code personnalisé)	Limitée (CRUD via WSDL)	Limitée (CRUD via REST standard)	SuiteScript complet
Format de réponse	JSON ou Texte	XML (SOAP)	JSON	HTML ou JSON
Authentification	TBA (OAuth 1.0) / OAuth 2.0 Bearer	TBA / OAuth 1.0	(Uniquement) OAuth 2.0	ID de session / Jeton
Unités de gouvernance	5 000 unités/requête	N/A	N/A	1 000 unités/requête
Cas d'utilisation typiques	API personnalisées & logique de données	CRUD d'enregistrement standard (tous types)	CRUD d'enregistrement standard	Pages web personnalisées (UI)

(Sources du tableau 1 : documentation NetSuite et guides d'intégration [4], complétés par [52] pour les modes d'authentification.)

Ce contexte prépare le terrain pour notre plongée approfondie dans les URL et l'authentification des RESTlets. Les sections suivantes traitent en détail des **domaines spécifiques au compte**, expliquent le **flux TBA** de bout en bout, et traitent enfin d'**OAuth 2.0 et des portées**.

Domaines spécifiques au compte NetSuite et URL de RESTlets

Le passage de NetSuite aux domaines spécifiques au compte signifie que chaque compte possède son propre préfixe de domaine unique pour tous les services (Source: docs.oracle.com) (Source: docs.oracle.com). La chaîne de domaine inclut l'ID du compte et encode également le type de compte. Pour un compte de production (par exemple, ID `123456`), vous pourriez voir des domaines comme `123456.app.netsuite.com` ou `123456.suitetalk.api.netsuite.com`. Pour un ID sandbox tel que `123456_SB1`, le domaine devient `123456-sb1.app.netsuite.com` (trait de soulignement → trait d'union, tout en minuscules) (Source: docs.oracle.com). Les comptes Release Preview (par exemple `123456_RP`) utilisent de la même manière des domaines `123456-rp...` (Source: docs.oracle.com). La Figure 1 (dans le Tableau 2) répertorie des exemples de formats de domaine spécifiques au compte :

Tableau 2 : Exemples de domaines spécifiques au compte par type de compte

TYPE DE COMPTE	FORMAT D'ID DE COMPTE EXEMPLE	EXEMPLE D'URL DE DOMAINE SPÉCIFIQUE AU COMPTE
Production	`123456`	`123456.app.netsuite.com` (points de terminaison UI/RESTlet)
Sandbox	`123456_SB1` → `123456-sb1`	`123456-sb1.app.netsuite.com` (points de terminaison UI/RESTlet)
Release Preview	`123456_RP` → `123456-rp`	`123456-rp.app.netsuite.com` (points de terminaison UI/RESTlet)

Chaque compte possède également des domaines correspondants pour les services web SOAP/REST (utilisant le suffixe `.suitetalk.api.netsuite.com`) et pour d'autres services comme SuiteAnalytics et les sites de catalogue externes (Source: docs.oracle.com) (Source: docs.oracle.com). Par exemple, un point de terminaison SuiteTalk REST pourrait être `https://123456.suitetalk.api.netsuite.com/services/rest/record/v1/customer/...` (Source: docs.oracle.com). Les URL précises pour tous les services pertinents sont visibles dans le sous-onglet *Company URLs* pour ce compte (Source: docs.oracle.com). **Important**, Oracle conseille de **ne jamais coder en dur ces domaines**. Utilisez plutôt les valeurs fournies ou les API de découverte, car même les domaines spécifiques au compte, bien que stables entre les centres de données, pourraient changer dans les architectures futures. En effet, NetSuite interdit l'épinglage de certificat (certificate pinning) sur ces domaines car ils peuvent changer sans préavis (Source: docs.oracle.com) (Source: docs.oracle.com).

Parce que les comptes peuvent être déplacés entre les centres de données et que les entreprises intègrent souvent plusieurs comptes, NetSuite fournit des points de terminaison de découverte de domaine. Pour les clients RESTlet, appeler `https://rest.netsuite.com` (le service de découverte REST de NetSuite) avec les identifiants du compte renvoie les noms de domaine RESTlet corrects (Source: docs.oracle.com). Alternativement, SuiteScript peut trouver les domaines par programmation : par exemple

```
N/url.resolveDomain({ hostType: url.HostType.RESTLET, accountId: '123456' });
```

renverra le domaine RESTlet pour le compte 123456 (Source: docs.oracle.com). L'utilisation de ces méthodes dynamiques évite les dépendances fragiles sur des URL fixes.

Une fois le domaine correct connu, un RESTlet est invoqué en construisant une requête HTTPS vers ce domaine avec les identifiants de script et de déploiement du RESTlet dans le chemin ou les paramètres de requête. Par exemple, un appel RESTlet externe typique utilise une URL de la forme :

```
https://<accountDomain>/app/site/hosting/restlet.nl?script=<SCRIPT_ID>&deploy=<DEPLOY_ID>
```

Pour un ID de compte de production 123456, le domaine exemple est `123456.app.netsuite.com`, donc un appel pourrait ressembler à `https://123456.app.netsuite.com/app/site/hosting/restlet.nl?script=1&deploy=1` (Source: docs.oracle.com). (Les services Web REST de NetSuite utilisent un chemin de base différent sous `/services/rest/...`, comme indiqué en [52].) Étant donné que le domaine du compte est spécifique à chaque locataire, les clients doivent utiliser le domaine correct pour le compte cible. La règle est : *« l'URL doit inclure un domaine spécifique à votre compte NetSuite »* (Source: docs.oracle.com). Si vous accédez à un autre compte (par exemple, un environnement sandbox ou un compte client), vous devez utiliser le domaine de ce compte, et non le vôtre.

En pratique, la plupart des intégrations gèrent la sélection du domaine de l'une des manières suivantes :

- **Configuration statique :** Les administrateurs copient le domaine correct depuis l'interface utilisateur de NetSuite et le stockent dans les paramètres ou le code de l'intégration.
- **Découverte dynamique :** Au moment de l'exécution, l'intégration interroge d'abord NetSuite (par exemple `rest.netsuite.com` ou SuiteScript) pour obtenir le domaine.
- **N/url.resolveDomain :** Si le code est écrit en tant que SuiteScript (Suitelet) au sein de NetSuite, utilisez le module `N/url` comme indiqué dans *Retrieve the Domain for Calling a RESTlet* (Source: docs.oracle.com).

En résumé, les domaines spécifiques au compte découplent les intégrations des adresses physiques des centres de données (Source: docs.oracle.com) et doivent toujours être utilisés dans les URL des RESTlets. Le tableau 2 met en évidence la façon dont l'ID de compte est mappé aux domaines pour différents types de comptes (Source: docs.oracle.com) (Source: docs.oracle.com). Les sections suivantes traitent de l'authentification lors de l'appel de ces URL RESTlet.

Authentification des RESTlets NetSuite

Les RESTlets nécessitent une authentification pour chaque appel externe (Source: www.brokenrubik.com) (Source: docs.oracle.com). NetSuite prend en charge plusieurs méthodes d'authentification, mais pour les clients RESTlet externes, les options principales sont l'**authentification basée sur les jetons (TBA - OAuth 1.0a)** et **OAuth 2.0** (Source: docs.oracle.com). (Il existe une méthode héritée appelée NLAuth — qui transmet la société, le nom d'utilisateur, le mot de passe, le rôle, etc. dans un en-tête HTTP — mais Oracle la déconseille pour les nouvelles intégrations (Source: docs.oracle.com).) Il est essentiel que tout appel RESTlet provenant de l'extérieur comporte un en-tête HTTP `Authorization` (aucun appel anonyme, sauf si le RESTlet est explicitement déployé *sans* connexion, ce qui est rare).

En règle générale, les conseils actuels d'Oracle sont que **OAuth 2.0 est privilégié** pour les RESTlets lorsque cela est possible (Source: docs.oracle.com). Cependant, de nombreuses intégrations existantes utilisent toujours l'authentification basée sur les jetons (TBA) de NetSuite, car elle est la norme de facto pour les intégrations RESTlet/SOAP serveur à serveur depuis des années. Nous détaillerons les deux approches ci-dessous.

Avant d'aborder les flux, notez une restriction supplémentaire : si le rôle NetSuite est marqué **« Web Services Only »** (Services Web uniquement), cet utilisateur ne peut pas appeler de RESTlets (Source: docs.oracle.com) (Source: docs.oracle.com). Les RESTlets sont une fonctionnalité de *SuiteScript* (faisant partie de la plateforme SuiteCloud), donc le rôle de l'utilisateur ou de l'intégration doit disposer des autorisations SuiteScript normales. (La documentation avertit qu'un rôle WebServices-Only recevra une erreur `INVALID_LOGIN_CREDENTIALS` s'il tente un appel RESTlet (Source: docs.oracle.com).) En pratique, on crée un rôle d'intégration dédié (ou on utilise un rôle existant) avec les autorisations appropriées, puis sous Configuration → Utilisateurs/Rôles, on crée un *Jeton d'accès* (Access Token) pour ce rôle (pour la TBA) ou on utilise ce rôle dans un flux OAuth2.

Flux d'authentification basée sur les jetons (TBA)

L'authentification basée sur les jetons dans NetSuite est basée sur OAuth 1.0a. Dans ce schéma, l'intégration configure un **Enregistrement d'intégration** (Integration Record) qui génère une clé consommateur (Consumer Key) et un secret (Source: www.brokenrubik.com). Ensuite, un administrateur crée un **Jeton d'accès** pour un utilisateur et un rôle spécifiques sous *Configuration → Utilisateurs/Rôles → Jetons d'accès* (Source: www.brokenrubik.com), ce qui donne un ID de jeton et un secret de jeton. L'intégration utilise ces quatre valeurs (clé/secret consommateur et ID/secret de jeton) pour signer les requêtes API. Chaque requête RESTlet inclut un en-tête OAuth 1.0 que NetSuite vérifie.

Les étapes de configuration de la TBA sont : activer la TBA dans le compte, créer l'enregistrement d'intégration (obtenir la clé/secret consommateur) et créer le jeton d'accès pour un utilisateur-rôle (obtenir l'ID/secret du jeton) (Source: www.brokenrubik.com). Après la configuration, le flux OAuth en trois étapes est le suivant (Oracle l'appelle le « Flux d'autorisation TBA en trois étapes » (Source: docs.oracle.com):

1. **Demande de jeton (Étape 1) :** Le client envoie un HTTP POST vers le point de terminaison de « demande de jeton ». Le format de l'URL est :

   ```
   https://<ACCOUNT_ID>.restlets.api.netsuite.com/rest/requesttoken
   ```

   où `<ACCOUNT_ID>` est l'ID du compte. (Si l'ID du compte n'est pas connu, on peut également envoyer un POST vers `https://system.netsuite.com/rest/requesttoken` en guise de secours (Source: docs.oracle.com).) Le client inclut un en-tête d'autorisation OAuth 1.0a signé avec la clé/le secret consommateur (pas encore de jeton) ainsi qu'un nonce et un horodatage (Source: docs.oracle.com). En cas de succès, NetSuite renvoie un « jeton de demande non autorisé ».

2. **Autorisation de l'utilisateur (Étape 2) :** Le client dirige ensuite l'utilisateur (ou la session du navigateur) vers l'URL d'autorisation utilisateur de NetSuite avec le jeton de demande. Il s'agit d'un simple HTTP GET vers :

   ```
   https://<ACCOUNT_ID>.app.netsuite.com/app/login/secure/authorizetoken.nl?oauth_token=<REQUEST_TOKEN>
   ```

   incluant le paramètre de requête `oauth_token` de l'étape 1. (Source: docs.oracle.com). NetSuite demandera à l'utilisateur de se connecter (si ce n'est pas déjà fait), puis affichera une page de consentement/approbation. L'utilisateur doit se connecter en utilisant le rôle approprié (NetSuite permettra de sélectionner un rôle si nécessaire (Source: docs.oracle.com) et cliquer sur « Autoriser » pour accorder l'accès au jeton de demande. À ce stade, NetSuite redirige le navigateur vers l'URL de rappel spécifiée dans l'enregistrement d'intégration, avec des paramètres incluant `oauth_token`, `oauth_verifier`, `company` (ID de compte) et `role` (l'ID interne du rôle) dans la chaîne de requête (Source: docs.oracle.com). (Si l'utilisateur clique sur « Refuser », le flux se termine.)

3. **Échange de jeton d'accès (Étape 3) :** Enfin, l'application prend le `oauth_token` autorisé et le `oauth_verifier` de l'étape 2 et envoie une requête POST vers le point de terminaison du jeton d'accès de NetSuite :

   ```
   POST https://<ACCOUNT_ID>.restlets.api.netsuite.com/rest/accesstoken
   ```

   avec un en-tête OAuth 1.0a qui inclut à la fois les informations d'identification du consommateur et le jeton de demande (et maintenant le vérificateur) (Source: docs.oracle.com). La réponse renvoie le **Jeton d'accès** final et le **Secret du jeton** (`oauth_token` et `oauth_token_secret`), que le client utilisera pour les requêtes authentifiées ultérieures (Source: docs.oracle.com). (Depuis NetSuite 2020.1, le paramètre OAuth `realm` est facultatif et n'est pas requis dans cette requête (Source: docs.oracle.com).)

Après l'étape 3, l'intégration dispose des quatre informations d'identification pour les futurs appels RESTlet (clé/secret consommateur et jeton/secret d'accès). Chaque futur appel HTTP vers l'URL du RESTlet inclut un en-tête d'autorisation comme :

```
Authorization: OAuth
    realm="<ACCOUNT_ID>",
    oauth_consumer_key="<CONSUMER_KEY>",
    oauth_token="<TOKEN_ID>",
    oauth_signature_method="HMAC-SHA256",
    oauth_timestamp="<UNIX_TIMESTAMP>",
    oauth_nonce="<RANDOM>",
    oauth_version="1.0",
    oauth_signature="<SIGNATURE>"
```

où `<SIGNATURE>` est le HMAC-SHA256 de la chaîne de base utilisant les deux secrets (Source: www.brokenrubik.com) (Source: timdietrich.me). Par exemple, le guide de Tim Dietrich montre précisément ce format, incluant le `realm` (ID de compte NetSuite) et exigeant HMAC-SHA256 (Source: timdietrich.me). Avec cet en-tête, le RESTlet reçoit la requête en tant qu'appel API authentifié. NetSuite enregistre chaque appel par rapport à la clé consommateur de l'enregistrement d'intégration, permettant le suivi de l'activité (Source: docs.oracle.com).

Il est important de noter qu'**à partir de NetSuite 2027.1, Oracle n'autorisera plus la création de *nouvelles* intégrations TBA** (Source: docs.oracle.com) (Source: www.houseblend.io). Les appels basés sur la TBA existants continueront de fonctionner, mais tous les nouveaux projets doivent utiliser OAuth 2.0. Cet avertissement est apparu dans plusieurs documents et articles de blog NetSuite (Source: docs.oracle.com) (Source: www.houseblend.io). En fait, le guide de migration 2026 de Houseblend souligne explicitement que les organisations doivent passer à OAuth 2.0 pour rester conformes (Source: www.houseblend.io). (Les flux OAuth 1.0 hérités nécessitent des signatures par requête et ne disposent pas de jetons de rafraîchissement natifs, ce qui les rend moins alignés avec les normes de sécurité modernes (Source: www.houseblend.io).)

Le tableau 3 ci-dessous décrit les étapes du flux TBA des RESTlets avec des exemples de points de terminaison :

ÉTAPE	OBJECTIF	MÉTHODE HTTP ET POINT DE TERMINAISON	NOTES
1. Demande de jeton	Obtenir un jeton OAuth non autorisé	`POST https://<ACCOUNT>.restlets.api.netsuite.com/rest/requesttoken` (Source: docs.oracle.com)	Inclure l'en-tête OAuth (clé/secret consommateur, nonce, horodatage).
2. Autoriser le jeton de demande	L'utilisateur accorde l'accès (connexion et consentement)	`GET https://<ACCOUNT>.app.netsuite.com/app/login/secure/authorizetoken.nl?oauth_token=<REQUEST_TOKEN>` (Source: docs.oracle.com)	Rediriger l'utilisateur vers cette URL. Après la connexion, NetSuite redirigera avec `oauth_verifier`.
3. Obtenir le jeton d'accès	Échanger contre le jeton OAuth final	`POST https://<ACCOUNT>.restlets.api.netsuite.com/rest/accesstoken` (Source: docs.oracle.com)	Inclure l'en-tête OAuth (clé/secret consommateur, jeton/secret de demande, vérificateur). Répond avec `oauth_token`, `oauth_token_secret`.

Tableau 3 : Flux OAuth 1.0 (TBA) pour les RESTlets NetSuite. Les domaines exemples utilisent le modèle de domaine spécifique au compte ; voir le texte pour les formats d'URL complets.

L'enregistrement d'intégration est créé (s'il n'est pas pré-créé) lors de l'échange du jeton. Il est crucial de noter que NetSuite **installe automatiquement l'enregistrement d'intégration** pour l'application demandeuse avec le nouveau jeton (Source: docs.oracle.com) (Source: docs.oracle.com). Un administrateur peut contrôler si l'enregistrement est activé automatiquement via les *Préférences des services Web SOAP* (Exiger l'approbation lors de

l'installation automatique) (Source: docs.oracle.com) (Source: docs.oracle.com). Cela garantit que la clé consommateur de l'intégration est enregistrée dans le compte et peut être suivie ou bloquée si nécessaire (Source: docs.oracle.com). (Les appels d'une intégration peuvent être examinés sur l'*Enregistrement d'intégration* ou via le *Journal d'exécution des RESTlets* (Source: docs.oracle.com).)

Exemple et considérations sur la TBA

En termes pratiques, la prise en charge par des bibliothèques (par exemple, les bibliothèques OAuth 1.0a) ou des outils comme Postman peut gérer la signature TBA. Le blog BrokenRubik donne un exemple de structure d'en-tête et de flux (Source: www.brokenrubik.com) (Source: www.brokenrubik.com). Il faut également gérer des nuances telles que l'encodage d'URL et l'inclusion de paramètres dans la chaîne de base de signature (Source: docs.oracle.com) (Source: docs.oracle.com). Notez que l'expiration de la TBA NetSuite est liée à l'état de l'enregistrement d'intégration ; les jetons n'expirent pas d'eux-mêmes à moins d'être révoqués. De plus, si le rôle de l'intégration nécessite des améliorations (par exemple, 2FA ou des autorisations modifiées), l'utilisateur peut avoir besoin de répéter ces étapes pour se ré-autoriser. La documentation NetSuite propose un point de terminaison *IssueToken* alternatif pour l'émission de jetons sans interaction utilisateur, mais Oracle recommande d'utiliser le flux complet en trois étapes pour les nouvelles intégrations (Source: docs.oracle.com).

Point critique : la TBA nécessite des signatures **HMAC-SHA256** — HMAC-SHA1 n'est plus pris en charge depuis 2023.1 (Source: docs.oracle.com). Chaque signature d'appel TBA doit utiliser SHA-256. De plus, le paramètre de domaine OAuth 1.0 (ID de compte) est désormais facultatif (Source: docs.oracle.com). Lors de la création d'une application, assurez-vous que votre bibliothèque OAuth est configurée pour les exigences exactes de NetSuite (méthode de signature, encodage, ordre des paramètres, etc.).

Authentification OAuth 2.0 et portées (Scopes)

Oracle fournit désormais une prise en charge d'OAuth 2.0 pour les API REST de NetSuite (RESTlets et SuiteTalk REST). Contrairement à la TBA, OAuth 2.0 utilise des jetons porteurs (pas de signature par requête) et est plus facile à mettre en œuvre pour les scénarios centrés sur l'utilisateur et sur la machine (Source: www.brokenrubik.com) (Source: docs.oracle.com). La configuration de l'intégration OAuth 2.0 implique : l'activation de la fonctionnalité OAuth 2.0 dans le compte, la création d'un enregistrement d'intégration configuré pour OAuth2, puis l'exécution du flux de code d'autorisation ou d'informations d'identification client dans l'application cliente (Source: docs.oracle.com) (Source: docs.oracle.com).

L'enregistrement d'intégration (Configuration > Intégration > Gérer les intégrations) dispose d'un sous-onglet *Authentification* où deux types d'octroi peuvent être activés : **Authorization Code Grant** (pour les flux interactifs) et **Client Credentials (Machine-to-Machine) Grant** (pour les flux serveur à serveur) (Source: docs.oracle.com) (Source: docs.oracle.com). Il est possible de cocher les deux cases si nécessaire. L'enregistrement inclut également des champs pour les URI de redirection (pour l'octroi de code), ainsi qu'une case à cocher « Client public » / paramètres PKCE si vous utilisez un client public (sans secret) [* (Source: docs.oracle.com)*]. Il est important de noter que l'enregistrement d'intégration comporte des cases à cocher intitulées *RESTlets*, *REST Web Services*, *SuiteAnalytics Connect*, *AI Connector Service*, etc. Cocher **RESTlets** autorise cette intégration à appeler des RESTlets ; cocher **REST Web Services** autorise SuiteTalk REST, et ainsi de suite (Source: docs.oracle.com). Lorsque vous demanderez ultérieurement un jeton OAuth 2.0, les portées (scopes) de ce jeton seront limitées à ces ressources sélectionnées.

Pour un scénario délégué par l'utilisateur, le flux **Authorization Code Grant** fonctionne comme suit (il s'agit de l'octroi de code OAuth2 standard). L'application dirige le navigateur de l'utilisateur vers le point de terminaison d'autorisation de NetSuite avec les paramètres de requête : `response_type=code`, `client_id=<CLIENT_ID>`, `redirect_uri=<votre_URI_app>`, `scope=<scopes>` et une chaîne `state` aléatoire (Source: docs.oracle.com). L'URL du point de terminaison est de la forme :

```
https://<ACCOUNT_DOMAIN>/login/oauth2/v1/authorize
    ?response_type=code
    &client_id=<clientId>
    &redirect_uri=<yourRedirectUri>
    &scope=<scopeList>
    &state=<randomString>
```

Par exemple (illustratif) : `https://123456.app.netsuite.com/login/oauth2/v1/authorize?response_type=code&client_id=MyAppID&redirect_uri=https://app.example.com/callback&scope=rest+soap+xml&state=xyz` (Source: docs.oracle.com). Notez que `<ACCOUNT_DOMAIN>` est le domaine spécifique au compte. L'utilisateur se connecte (ou passe par SSO) et approuve les portées. NetSuite redirige ensuite vers l'URI `redirect_uri` avec un paramètre `code` et le même `state`.

Ensuite, l'application cliente échange ce code contre des jetons en effectuant un POST vers le point de terminaison de jeton, par exemple :

```
https://<ACCOUNT_DOMAIN>/services/rest/auth/oauth2/v1/token
```

avec les données de formulaire `grant_type=authorization_code`, `code=<AUTH_CODE>`, `redirect_uri=<identique>`, et en utilisant l'authentification HTTP Basic avec `client_id:client_secret` (Source: blogs.oracle.com) (Source: blogs.oracle.com). La réponse est un jeton d'accès JSON Web Token (plus un jeton de rafraîchissement) au format JWT de NETSUITE (Source: blogs.oracle.com) (Source: blogs.oracle.com). L'exemple du blog Oracle démontre le point de terminaison et le fait que le jeton porteur (bearer token) est un JWT :

```
POST https://123456.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token
?code=<CODE>&redirect_uri=https://app.example.com/callback&grant_type=authorization_code
```

avec `Authorization: Basic <base64(client_id:secret)>` produit un corps JSON contenant `"access_token": "eyJraWQiOiIyMDIwXzEiLCJ0eXAiOiJKV1QiLCJh...","refresh_token":"...","scope":["RESTLETS"]...` (Source: blogs.oracle.com) (Source: blogs.oracle.com). (Dans cet exemple, la revendication « scope » du JWT renvoyé est `RESTLETS`, indiquant que le jeton peut appeler des RESTlets ; de même, il pourrait inclure `"soap"` ou `"rest"` si SuiteAnalytics ou SOAP étaient activés sur l'intégration.)

Le **Client Credentials Grant** est utilisé pour le serveur à serveur lorsqu'aucune interaction utilisateur n'est nécessaire. Dans NetSuite, cela implique souvent une paire de clés RSA. L'administrateur télécharge le certificat public via *Configuration → Intégration → Gérer l'authentification → Configuration des informations d'identification client OAuth 2.0 (M2M)* (Source: docs.oracle.com) et sélectionne l'application d'intégration dont la case « Client Credentials Grant » est cochée (Source: docs.oracle.com) (Source: docs.oracle.com). Au moment de l'exécution, le client utilise sa clé privée pour signer une assertion JWT afin de demander un jeton au même point de terminaison `/token` (avec `grant_type=client_credentials`). La réponse est un jeton d'accès (JWT) que l'intégration peut utiliser. Le guide Houseblend note que les informations d'identification client sont idéales pour la connexion machine-à-machine à long terme, et souvent associées à un octroi de code à courte durée de vie pour la configuration initiale (Source: www.bundlet.com) (Source: www.bundlet.com). En effet, un modèle recommandé est : utiliser un bref Authorization Code Grant (avec un rôle de niveau administrateur sélectionné) pour effectuer les tâches de provisionnement uniques, puis passer aux Client Credentials pour un accès continu (Source: www.bundlet.com) (Source: www.bundlet.com).

Portées (Scopes) OAuth 2.0

Dans OAuth 2.0, les **portées** définissent l'étendue de l'accès accordé par un jeton. Dans l'implémentation de NetSuite, les portées correspondent directement aux fonctionnalités activées sur l'enregistrement d'intégration. Les portées clés incluent `RESTlets` (permet d'appeler des RESTlets), `REST web services` (SuiteTalk REST), `SuiteAnalytics Connect`, et d'autres (Source: docs.oracle.com). Lorsque l'application demande une autorisation, elle peut inclure un paramètre de portée (par exemple `scope=rest+soap+xml` comme indiqué dans un exemple pour SuiteProjects (Source: docs.oracle.com). En fin de compte, le JWT du jeton émis contient une liste de portées dans sa charge utile, et Oracle garantit que le jeton ne peut être utilisé que pour les API correspondantes. Par exemple, le jeton de l'exemple REST Web Services incluait `"scope":["RESTLETS"]` (Source: docs.oracle.com), ce qui signifie qu'il autorisait les appels RESTlet. Modifier ou demander des portées différentes nécessite généralement un nouveau flux d'autorisation ; par exemple, tenter de combiner des portées incompatibles (comme un connecteur BI et d'autres portées) entraînera une erreur `invalid_scope` (Source: docs.oracle.com) dans les scénarios SuiteProjects.

Du point de vue de l'intégration, les **cases à cocher de l'enregistrement d'intégration sont effectivement les portées OAuth**. Comme le décrit l'aide officielle, vous *« cochez [RESTlets] si votre intégration OAuth 2.0 nécessite l'accès aux RESTlets »*, et de même pour les services Web REST, Connect, AI Connector, etc. (Source: docs.oracle.com). Celles-ci garantissent que le jeton (lorsqu'il est acquis) est automatiquement limité. En effet, NetSuite lie la portée OAuth aux fonctionnalités API que l'intégration est autorisée à utiliser. Houseblend explique qu'OAuth2 permet des « portées granulaires » alors qu'OAuth1.0/TBA ne le permettait pas (Source: www.houseblend.io) ; en effet, les portées de NetSuite offrent un contrôle plus précis sur les autorisations des jetons que l'approche « tout ou rien » de TBA.

Après avoir obtenu le jeton d'accès, les appels RESTlet incluent un simple en-tête porteur (bearer) :

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

comme indiqué dans la documentation et les exemples (Source: docs.oracle.com) (Source: docs.oracle.com). Aucune signature n'est nécessaire. NetSuite valide le JWT du jeton, vérifie sa signature (avec la clé publique de NetSuite) et s'assure qu'il couvre la ressource demandée. Pour les RESTlets, l'URL d'appel réelle pourrait être, par exemple, `https://123456.app.netsuite.com/app/site/hosting/restlet.nl?script=1&deploy=1`

(Source: docs.oracle.com), et l'en-tête `Authorization: Bearer eyJraWQiOiI...` (Source: docs.oracle.com). Le tutoriel BrokenRubik et les exemples Oracle confirment cette utilisation (Source: docs.oracle.com) (Source: docs.oracle.com).

Exemple : Flux Authorization Code

Pour illustrer, considérons un octroi de code d'autorisation pour une intégration RESTlet. Tout d'abord, l'enregistrement d'intégration est créé avec **Authorization Code Grant** activé, le(s) URI de redirection approprié(s) saisi(s), et les portées (RESTlets, etc.) cochées (Source: docs.oracle.com) (Source: docs.oracle.com). L'utilisateur se connecte à l'application cliente, qui navigue via un navigateur vers :

```
https://123456.app.netsuite.com/login/oauth2/v1/authorize
    ?response_type=code
    &client_id={CLIENT_ID}
    &redirect_uri=https://myapp.com/callback
    &scope=restlets+reststore  (par exemple)
    &state=random123
```

(La syntaxe exacte des portées utilise `+` ou `%20` pour séparer les valeurs (Source: docs.oracle.com).) NetSuite demande ensuite à l'utilisateur d'autoriser, et si accordé, redirige vers `https://myapp.com/callback?code=ABC123&state=random123`. Le client échange `ABC123` à

```
POST https://123456.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token
```

avec les en-têtes `Authorization: Basic <base64(client_id:client_secret)>` et le corps `grant_type=authorization_code&code=ABC123&redirect_uri=https://myapp.com/callback` (Source: blogs.oracle.com). La réponse contient un JSON comme `{ "access_token": "eyJraWQiOiIyMDIwXzEi...","refresh_token":"...","scope":["RESTLETS"], ... }` (Source: blogs.oracle.com) (Source: blogs.oracle.com). Le client appelle ensuite l'URL RESTlet avec `Authorization: Bearer eyJh...` et NetSuite le traite.

Exemple : Flux Client Credentials

Dans un scénario machine-à-machine, l'administrateur télécharge un certificat public (généré par l'application) sous *Gérer l'authentification → Configuration des informations d'identification client OAuth 2.0 (M2M)* (Source: docs.oracle.com). Un mappage est créé reliant l'enregistrement d'intégration et le rôle. Lorsque l'application s'exécute, elle demande un jeton d'informations d'identification client de la même manière à partir de `/services/rest/auth/oauth2/v1/token?grant_type=client_credentials`, en s'authentifiant avec son ID client et le certificat signé. La réponse est à nouveau un jeton d'accès JWT qui peut être utilisé pour appeler des RESTlets ou des services Web REST, sous réserve des portées de l'intégration (par exemple, la case à cocher « RESTlets ») (Source: docs.oracle.com) (Source: docs.oracle.com). La documentation d'Oracle implique que le jeton couvre exactement les portées activées sur l'application.

Dans tous les cas, une fois qu'un jeton d'accès OAuth 2.0 est détenu, les appels RESTlet sont simplement des requêtes HTTPS standard vers le domaine spécifique au compte avec un en-tête Bearer (Source: docs.oracle.com) (Source: docs.oracle.com). Par exemple :

```
GET https://123456.app.netsuite.com/app/site/hosting/restlet.nl?script=5&deploy=1
Authorization: Bearer <access_token>
```

Cette simplicité (pas de signature par appel) est un avantage d'OAuth2 mentionné dans la documentation de NetSuite (Source: www.brokenrubik.com) (Source: docs.oracle.com).

Comparaisons et implications

Le passage de TBA à OAuth2 a des implications importantes pour les intégrations NetSuite. Le tableau 4 met en évidence les contrastes clés entre TBA (OAuth 1.0) et OAuth 2.0 du point de vue de NetSuite :

ASPECT	TBA (OAUTH 1.0)	OAUTH 2.0 (NETSUITE)
Flux	Échange de jetons en 3 étapes (avec consentement de l'utilisateur) (Source: docs.oracle.com)	Code Grant ou Client Credentials (avec jetons de rafraîchissement)
Identifiants	Clé/Secret consommateur + ID/Secret jeton (4 valeurs) (Source: www.brokenrubik.com)	ID/Secret client (plus certificat pour M2M)
Signature de requête	Signature HMAC-SHA256 sur chaque requête (Source: docs.oracle.com)	Aucune signature par requête (en-tête de jeton Bearer uniquement)
Expiration du jeton	Les jetons n'expirent pas par défaut (révoqués manuellement)	Les jetons d'accès ont une durée de vie limitée ; les jetons de rafraîchissement peuvent être utilisés
Portées/Granularité	Aucun concept de portées (toute l'API autorisée si le jeton est valide)	Portées sélectionnables (RESTlets, REST WS, Connect, etc.) (Source: docs.oracle.com)
Interaction utilisateur	Requise (page de consentement) sauf si l'on utilise le contournement IssueToken (Source: docs.oracle.com)	Code Grant nécessite la connexion/consentement de l'utilisateur ; Client Credentials est sans interface
Cas d'utilisation	Serveur-serveur, intégrations héritées (Source: www.brokenrubik.com)	Scénarios délégués par l'utilisateur (portail, UI) et serveur-serveur (Source: docs.oracle.com)
Statut de dépréciation	Déprécié pour les nouvelles intégrations après 2027 (Source: docs.oracle.com) (Source: www.houseblend.io)	Préféré pour les nouvelles intégrations ; activement pris en charge

(Sources : Documentation NetSuite [12][30], guides d'experts [6][72][76].)

D'un point de vue sécuritaire, OAuth 2.0 offre des avantages : il prend nativement en charge les méthodes d'authentification modernes (par exemple, la connexion utilisateur SAML ou OIDC peut être utilisée pour l'octroi de code initial) (Source: docs.oracle.com), et il génère des jetons JWT qui contiennent des portées (scopes) et d'autres revendications (utiles pour l'audit et l'application des rôles). L'analyse de Houseblend note qu'OAuth2 aligne NetSuite sur les normes de conformité d'entreprise (NIST, SOC2, PCI), tandis que le processus de signature OAuth1.0 de TBA est considéré comme obsolète (Source: www.houseblend.io) (Source: www.houseblend.io). En effet, les documents d'Oracle eux-mêmes insistent sur le contrôle de l'accès via des enregistrements d'intégration et des jetons à des fins d'audit : « utilisez les enregistrements d'intégration pour gérer l'activité des RESTlets ; chaque enregistrement affiche les appels RESTlet authentifiés à l'aide de la clé client de cet enregistrement » (pour TBA) (Source: docs.oracle.com). Dans le monde OAuth2, l'enregistrement d'intégration gère de la même manière les portées et les rôles que le jeton peut assumer.

En pratique, la migration des intégrations de TBA vers OAuth2 implique généralement la création d'un nouvel enregistrement d'intégration avec OAuth2 activé, la mise à jour du code d'authentification et, éventuellement, une étape de consentement de l'utilisateur ou de téléchargement de certificat. Les clients doivent inventorier les jetons TBA existants (environ 38 000 jetons dans le monde début 2024 selon une estimation (Source: www.houseblend.io) et prévoir une réautorisation si nécessaire. Le guide Houseblend fournit des stratégies de migration approfondies et souligne que d'ici 2027, les organisations *doivent* avoir remplacé TBA par OAuth2 pour tout nouveau développement (Source: www.houseblend.io).

Études de cas et exemples

Plusieurs praticiens ont documenté des scénarios réels d'utilisation de RESTlets avec ces flux d'authentification. Par exemple, l'ingénieur SDN d'Oracle, Vlastimil Martinek, décrit un flux d'intégration *SuiteApp* où l'ancienne méthode (création manuelle d'un jeton) est remplacée par des étapes OAuth2 automatisées (Source: blogs.oracle.com) (Source: blogs.oracle.com). Son blog explique l'utilisation du flux de code d'autorisation pour obtenir un jeton d'accès sans que l'utilisateur n'ait à copier des identifiants, puis le téléchargement programmé des certificats d'identifiants client via des appels REST (voir sa section « What is there to automate? » (Source: blogs.oracle.com). En substance, cela reproduit le modèle « code d'autorisation puis identifiants client » décrit ci-dessus.

De même, le blog de Bundlet de 2026 présente un scénario détaillé d'intégration client : ils conditionnent leur intégration sous forme de SuiteApp et demandent au client d'accorder un code d'autorisation de niveau administrateur d'une heure (via un client OAuth public avec PKCE) uniquement pour effectuer la configuration, puis passent à une connexion par identifiants client à plus long terme pour les opérations quotidiennes (Source:

www.bundlet.com) (Source: www.bundlet.com). Ils soulignent que limiter le jeton d'octroi de code à 1 heure et à une portée de niveau administrateur est en réalité plus sûr que de simplement fournir un identifiant administrateur à longue durée de vie, car le processus est approuvé par l'utilisateur et éphémère (Source: www.bundlet.com) (Source: www.bundlet.com).

Ces études de cas soulignent les meilleures pratiques :

- Utilisez des rôles avec le *moindre privilège* pour les jetons (administrateur uniquement si absolument nécessaire, et seulement pour une courte durée) (Source: www.bundlet.com).
- Automatisez la gestion des certificats dans la mesure du possible pour minimiser les erreurs manuelles. (NetSuite 2026.1 a introduit un *point de terminaison de rotation de certificat* pour les mises à jour automatisées des clés d'identifiants client (Source: www.bundlet.com).)
- Offrez une expérience utilisateur fluide en intégrant les étapes OAuth2 dans le déploiement SuiteApp/SDF ou le flux de travail du portail, plutôt que par copier-coller manuel de jetons (Source: blogs.oracle.com) (Source: blogs.oracle.com).

Orientations futures et implications

À l'avenir, Oracle a signalé qu'à partir de 2027, OAuth 2.0 sera la seule méthode prise en charge pour les nouvelles intégrations (Source: docs.oracle.com) (Source: www.houseblend.io). Les organisations doivent donc concevoir toutes les nouvelles intégrations RESTlet et SuiteTalk en gardant OAuth2 à l'esprit. Cela inclut la maîtrise de l'utilisation des enregistrements d'intégration (avec les portées appropriées) et éventuellement l'adoption de fonctionnalités telles que PKCE (requise pour les clients publics) et les API de rotation de certificats. Le modèle de portée peut également évoluer : actuellement, des portées telles que « RESTlets » et « REST Web Services » existent, mais à mesure que NetSuite développe de nouveaux services (par exemple, connecteur IA, services Web SuiteQL, etc.), des portées supplémentaires seront ajoutées, comme le suggèrent les nouvelles cases à cocher dans l'enregistrement d'intégration (Source: docs.oracle.com).

Un sujet à surveiller est la manière dont les jetons OAuth 2.0 seront utilisés avec SuiteAnalytics Connect (ODBC), qui dispose traditionnellement de son propre système de jetons (jetons SuiteAnalytics). NetSuite a déjà ajouté une portée OAuth2 « SuiteAnalytics Connect » (Source: docs.oracle.com). Une autre implication est que toutes les bibliothèques clientes ou outils utilisés par les partenaires doivent prendre en charge OAuth2. De nombreux SDK NetSuite existants sont mis à jour en conséquence.

D'un point de vue sécuritaire, la tendance de l'industrie à s'éloigner d'OAuth1.0 (parallèle à l'abandon par Intuit en 2020 (Source: www.houseblend.io) suggère que les futures versions de NetSuite pourraient éventuellement abandonner l'ancienne méthode de connexion de base NLAuth pour les services Web, et peut-être même supprimer le jeton à usage unique Alloy (les jetons éphémères à usage unique de NLAuth). Pour l'instant, NetSuite autorise toujours la connexion par jeton nom d'utilisateur/mot de passe (NLAuth) pour les Suitelets et les RESTlets, mais la documentation avertit qu'elle n'est « pas prise en charge pour les nouveaux RESTlets » (Source: docs.oracle.com). Les entreprises devraient également envisager d'éliminer leur dépendance à l'égard de ces anciennes méthodes.

Avec l'adoption d'OAuth2, des changements dans la chaîne d'outils seront nécessaires : les passerelles API et les plateformes d'intégration doivent gérer les flux OAuth2. La surveillance passera de la recherche de clés statiques au suivi de l'utilisation des jetons porteurs (les journaux d'exécution des enregistrements d'intégration en sont un exemple).

Enfin, en utilisant des portées et des jetons granulaires, les entreprises peuvent mieux répondre aux normes de conformité (principe du moindre privilège, journaux d'audit, rotation plus facile des identifiants). Le rapport de Houseblend soutient que cela aligne NetSuite sur les directives du NIST et les directives PCI/SOC2 d'ici 2027 (Source: www.houseblend.io). En bref, le passage aux domaines spécifiques au compte et à OAuth 2.0 est une modernisation significative qui améliorera la stabilité et la sécurité à long terme, mais elle nécessite de former à nouveau les développeurs et les administrateurs aux nouveaux processus.

Conclusion

Les intégrations RESTlet de NetSuite reposent sur des domaines spécifiques au compte pour isoler les intégrations des changements d'infrastructure, et ces domaines forment la base de toutes les URL de point de terminaison REST (Source: docs.oracle.com) (Source: docs.oracle.com). Une intégration efficace nécessite de découvrir et d'utiliser le domaine correct pour chaque compte, soit via les services dynamiques de NetSuite, soit via la page des URL de l'entreprise (Source: docs.oracle.com) (Source: docs.oracle.com).

L'authentification auprès de ces points de terminaison RESTlet peut être effectuée via l'authentification par jeton (OAuth1.0) ou OAuth 2.0. TBA a été le choix hérité pour les appels serveur à serveur, utilisant des en-têtes OAuth1 signés et un échange de jetons en plusieurs étapes (Source: docs.oracle.com) (Source: docs.oracle.com). Cependant, Oracle recommande fortement la transition vers OAuth 2.0 : cela simplifie la mise en œuvre et s'aligne sur les pratiques de sécurité modernes (Source: docs.oracle.com) (Source: docs.oracle.com). OAuth 2.0 dans NetSuite crée des jetons porteurs avec prise en charge du rafraîchissement et restrictions de portée, contrairement aux jetons non signés et à longue durée de vie de TBA (Source:

www.houseblend.io). Les enregistrements d'intégration NetSuite permettent désormais un contrôle précis des portées OAuth 2.0 (par exemple, RESTlets vs services Web REST) et prennent en charge à la fois les flux interactifs (code d'autorisation) et non interactifs (identifiants client) (Source: docs.oracle.com) (Source: docs.oracle.com).

D'ici 2027, Oracle n'autorisera plus les nouvelles intégrations basées sur TBA (Source: docs.oracle.com) (Source: www.houseblend.io). Cette échéance signifie que les organisations doivent planifier leurs migrations. La bonne nouvelle est qu'une documentation étendue et une expertise communautaire existent désormais, y compris des guides officiels (Source: docs.oracle.com) (Source: docs.oracle.com) et des blogs d'études de cas (Source: www.bundlet.com) (Source: blogs.oracle.com). Les intégrateurs devraient tirer parti de ces ressources pour basculer leurs clients RESTlets et SuiteTalk vers OAuth 2.0 avec des domaines spécifiques au compte bien avant la date limite. Ce faisant, ils obtiendront des intégrations plus robustes et maintenables qui répondent à la future feuille de route de NetSuite.

Références : Tous les faits et procédures ci-dessus sont tirés de la documentation officielle de NetSuite et de sources techniques faisant autorité (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: docs.oracle.com) (Source: blogs.oracle.com), y compris le centre d'aide NetSuite, les blogs de support Oracle et les analyses d'experts (Source: www.brokenrubik.com) (Source: timdietrich.me) (Source: www.houseblend.io) (Source: blogs.oracle.com). Chaque section citée fournit des détails étape par étape ou conceptuels sur les sujets abordés. Les tableaux et exemples résument et contrastent les informations explicitement documentées dans ces sources.

Étiquettes: restlets-netsuite, domaines-specifiques-au-compte, authentification-par-jeton, flux-tba, oauth-20, integration-netsuite, api-suitescript

---