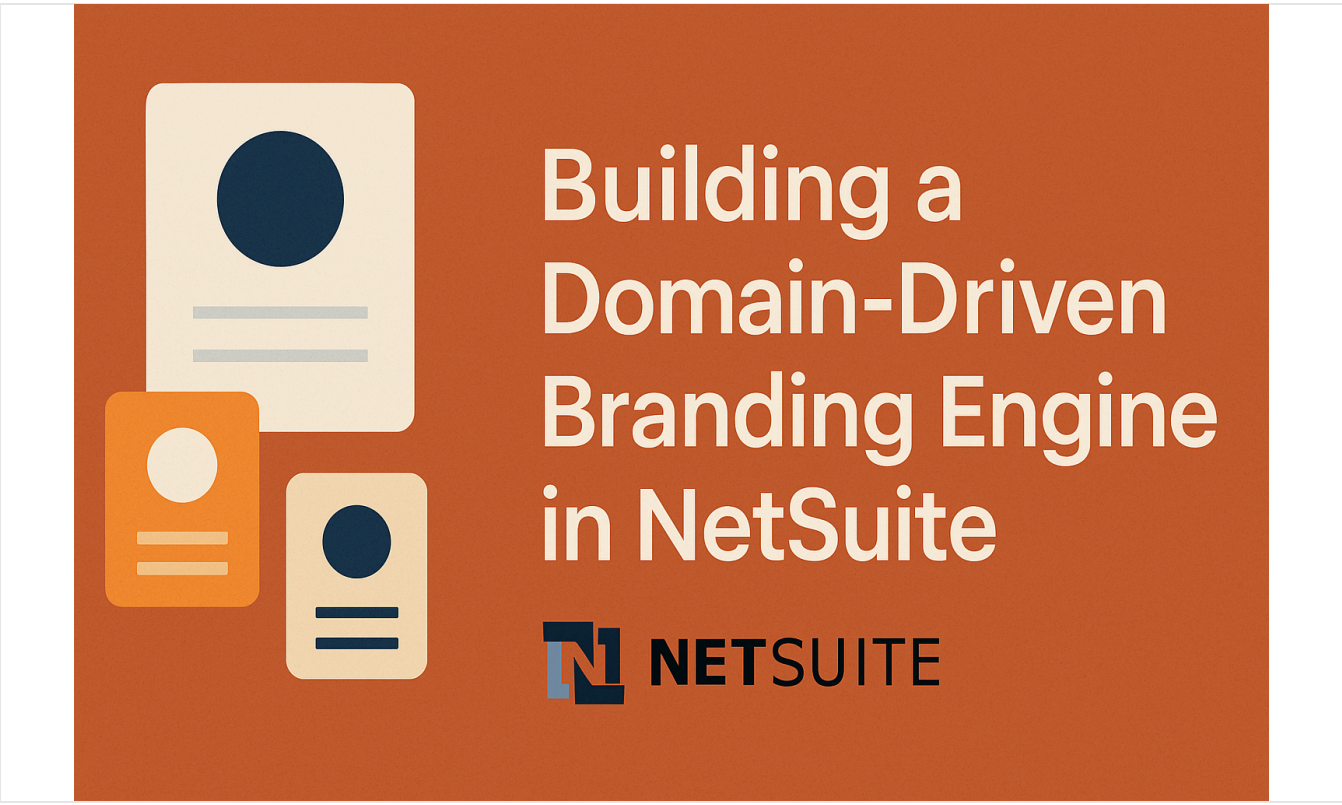
Architecture d'un moteur de marque axé sur le domaine dans NetSuite

Publié le 29 juillet 2025 40 min de lecture



Créer un moteur de personnalisation de marque axé sur le domaine dans les formulaires NetSuite

Introduction à la personnalisation de marque axée sur le domaine dans les ERP (contexte NetSuite)

Dans une organisation multi-filiales, la **personnalisation de marque axée sur le domaine** (Domain-Driven Branding) fait référence à l'adaptation de l'interface utilisateur et des documents pour refléter l'identité unique de chaque marque (logo, couleurs, messages) en fonction du "domaine" ou du contexte (tel que la filiale, la région ou le rôle de l'utilisateur). Ce concept est crucial dans les systèmes ERP comme NetSuite OneWorld, où un seul compte peut servir de nombreuses divisions ou marques. En mettant en œuvre la personnalisation de marque axée sur le domaine, les entreprises s'assurent que les utilisateurs internes et les parties prenantes externes (clients, fournisseurs) voient constamment l'identité de marque correcte, favorisant le professionnalisme et la confiance. La structure de filiales OneWorld de NetSuite prend intrinsèquement en charge une certaine différenciation de marque – par exemple, chaque filiale peut avoir son propre nom, adresse et logos pour les formulaires. Cependant, les capacités prêtes à l'emploi sont limitées aux éléments de base. De nombreuses entreprises souhaitent **un système qui adapte dynamiquement les formulaires et les sorties par marque**. Comme le note Marty Zigman, expert NetSuite, "les clients veulent avoir un seul formulaire qui génère une présentation différente en fonction de l'entreprise... une filiale peut avoir une image de marque, une adresse postale locale, etc., qui doivent figurer sur les communications clients". En d'autres termes, l'ERP doit refléter de manière transparente plusieurs identités de marque sans maintenir des systèmes complètement séparés. Ce rapport explore comment architecturer un **moteur de personnalisation de marque** au sein de NetSuite – combinant des outils techniques (SuiteScript, enregistrements personnalisés, flux de travail, personnalisation de l'interface utilisateur) avec des considérations stratégiques (cohérence de la marque à travers les domaines, personnalisation régionale, expérience utilisateur).

Architecture d'un moteur de personnalisation de marque basé sur le domaine dans NetSuite

La conception d'un moteur de personnalisation de marque implique deux éléments centraux : **(1) Un stockage centralisé des attributs de marque** (par domaine/marque) et **(2) Une logique d'exécution pour appliquer ces attributs** sur les formulaires et les documents. À un niveau élevé, le moteur fonctionne comme suit :

- **Stockage de la configuration de la marque :** Créez un modèle de données pour contenir les paramètres spécifiques à la marque. Cela pourrait utiliser les *enregistrements de filiale NetSuite* (augmentés de champs personnalisés pour des informations de marque supplémentaires) ou un _ enregistrement personnalisé_ dédié (par exemple, "Profil de marque") pour plus de flexibilité. Chaque entrée de marque contiendrait des attributs tels que le fichier image du logo, les codes couleur (valeurs hexadécimales pour les couleurs primaires/secondaires), le nom légal ou le nom de marque "convivial", les slogans ou le texte de pied de page, etc. Pour les utilisateurs de OneWorld, une grande partie de cela commence par l'enregistrement de la filiale – NetSuite vous permet de télécharger un **Logo de filiale (Formulaires)** à utiliser sur les formulaires imprimés et un **Logo de filiale (Pages)** pour les pages d'interface utilisateur de cette filiale. (Ces logos sont utilisés par les formulaires et centres standard ; nous nous appuierons sur cela.) Vous pouvez également ajouter des champs personnalisés à l'enregistrement de la filiale pour stocker des informations supplémentaires comme la couleur de la marque ou le slogan. Dans un exemple réel, une entreprise a ajouté un champ personnalisé "Nom de marque" sur chaque filiale car le nom légal était différent de la marque visible par le client – ce nom de marque était ensuite utilisé sur les modèles d'e-mail et les formulaires au lieu du nom de la SARL. Alternativement, un **enregistrement personnalisé de marque** séparé peut être créé si les besoins de personnalisation de marque ne sont pas un-à-un avec les filiales (par exemple, plusieurs marques sous une seule filiale). La clé est de centraliser ces éléments d'identité afin qu'ils puissent être maintenus en un seul endroit pour chaque domaine.

- **Logique de personnalisation de marque dynamique :** En utilisant la personnalisation SuiteCloud (SuiteScript ou Workflow), le moteur détecte le contexte actuel (quel "domaine de marque" est en jeu) chaque fois qu'un formulaire est chargé ou qu'un document est généré, puis **applique les attributs de marque correspondants**. Le contexte du "domaine" peut être déduit de la filiale de l'enregistrement, du rôle ou de la filiale de l'utilisateur, d'un département/classe (si ceux-ci sont utilisés pour désigner les marques), ou même de l'URL de la requête ou du domaine du site web dans certains cas. Pour les formulaires NetSuite internes, c'est généralement la **filiale de l'enregistrement** ou la **filiale de l'utilisateur actuel** qui est le moteur. L'API d'exécution de SuiteScript fournit `runtime.getCurrentUser().subsidiary` qui donne la filiale de l'utilisateur actuel (OneWorld). Vous pouvez également obtenir la filiale de l'enregistrement via la valeur du champ sur l'enregistrement (par exemple, `scriptContext.newRecord.getValue('subsidiary')` dans un script `beforeLoad` de transaction). Pour les pages externes (comme un _ Centre client_ ou des *Formulaires externes* sur différents domaines), vous pourriez dériver la personnalisation de marque du domaine de l'URL ou du rôle de centre spécifique. Dans tous les cas, la logique du script utilise ce contexte pour rechercher la configuration de marque correspondante (par exemple, charger la filiale ou

l'enregistrement personnalisé de marque) et récupérer les attributs nécessaires (ID/URL du fichier logo, codes couleur, etc.). Ensuite, elle **modifie le formulaire ou la sortie à la volée** – par exemple, en injectant le logo correct, en modifiant les éléments stylistiques (couleurs ou texte de bannière) et en affichant des messages spécifiques à la marque. Nous explorerons plusieurs techniques pour implémenter ce comportement d'exécution dans NetSuite, y compris les scripts d'événements utilisateur SuiteScript, les Suitelets et les flux de travail "point-and-click".

- **Cibles de sortie du moteur :** Le moteur de personnalisation de marque doit affecter **à la fois les formulaires d'interface utilisateur internes et les documents imprimés/envoyés par e-mail**. Les formulaires internes (tels que le formulaire de saisie de commande client ou le formulaire d'enregistrement client dans l'interface utilisateur de NetSuite) peuvent afficher des éléments de marque pour guider les utilisateurs, tandis que les **documents destinés aux clients** (comme les factures PDF, les confirmations de commande) ont absolument besoin des logos et des informations d'entreprise appropriés pour chaque marque. Les modèles PDF/HTML avancés de NetSuite sont un outil principal pour les sorties de marque, car ils permettent une logique conditionnelle et des champs dynamiques dans les formulaires imprimés. Notre moteur se coordonnera avec les modèles avancés (ou sélectionnera dynamiquement des modèles) pour s'assurer, par exemple, qu'une facture PDF utilise le logo de filiale et le libellé de marque corrects. Ainsi, une architecture complète couvre à la fois la *personnalisation de l'interface utilisateur* et le *modélisation des documents*.

- **Approche centralisée vs. distribuée :** Stratégiquement, un moteur axé sur le domaine vous permet d'éviter de dupliquer des dizaines de formulaires ou de modèles pour chaque marque. Sans cela, on pourrait créer des formulaires personnalisés séparés pour chaque filiale (chacun avec des logos codés en dur, des différences de mise en page de formulaire, etc.) et les attribuer par rôle ou par utilisateur – mais cela devient rapidement un casse-tête de maintenance car les modifications doivent être propagées à tous les formulaires. Un moteur scripté, en revanche, peut présenter **un formulaire adaptatif** ou un modèle adaptatif qui s'ajuste en fonction du contexte. Cette logique centrale facilite grandement les mises à jour (comme un nouveau logo ou un nouveau champ) – mettez à jour la configuration ou le code une seule fois, et toutes les marques héritent du changement. Cela réduit également les erreurs de l'utilisateur (garantissant que le bon formulaire est toujours utilisé pour la bonne marque automatiquement). Nous discuterons de la façon d'implémenter un tel changement de formulaire dynamique et l'injection de contenu dans les sections suivantes.

(L'architecture peut être visualisée comme un flux : L'utilisateur/l'enregistrement déclenche le chargement du formulaire → SuiteScript/Workflow détermine le contexte (par exemple, Filiale = "Marque A") → Le moteur récupère les paramètres de la Marque A à partir de l'enregistrement personnalisé ou des champs de la filiale → Le moteur modifie le logo, les couleurs, les messages du formulaire en conséquence avant de le rendre à l'utilisateur. De même pour l'impression : lors de la génération d'un PDF, le modèle ou le script extrait le logo et les détails de la Marque A pour les afficher dans la sortie.)

Stockage et gestion des attributs de marque (enregistrements et champs personnalisés)

Un moteur de personnalisation de marque robuste repose sur des données bien structurées concernant chaque marque. Dans NetSuite, vous avez plusieurs options :

- **Utiliser les enregistrements de filiale (OneWorld) :** Chaque filiale dispose déjà de champs pour le **Logo de filiale (Formulaires)** et le **Logo de filiale (Pages)**, qui peuvent stocker des fichiers image à utiliser respectivement sur les formulaires imprimés et dans les pages de l'interface utilisateur. Dans OneWorld, les utilisateurs ayant un rôle "externe" (comme le Centre client ou le Centre fournisseur) verront en fait le logo de la filiale sur leurs pages si configuré, offrant une certaine personnalisation de l'interface utilisateur intégrée. Par exemple, l'activation de *"Toujours afficher le nom de la filiale"* peut faire en sorte que l'interface utilisateur n'affiche que le nom/logo de la filiale au lieu de celui du parent dans certains contextes. Au-delà des logos, l'enregistrement de la filiale contient le nom légal, l'adresse principale, le téléphone, etc., qui apparaissent souvent sur les documents. Nous pouvons joindre plus d'informations de marque en ajoutant des **champs personnalisés** à la filiale. NetSuite permet d'ajouter des champs à presque tous les enregistrements : pour les filiales, on peut naviguer vers *Personnalisation > Listes, Enregistrements et Champs > Autres champs d'enregistrement > Nouveau*, puis cibler le type d'enregistrement "Filiale". Comme le décrit Steven Hall, *"cela peut être facilement fait en modifiant n'importe quel enregistrement de filiale et en sélectionnant Personnaliser > Nouveau champ... Vous pouvez maintenant ajouter un champ à votre enregistrement de filiale"*. Il peut s'agir de champs de texte pour les codes couleur hexadécimaux, d'un champ pour le "Nom d'affichage de la marque" (s'il est différent du nom de la filiale), d'URL pour une feuille de style spécifique à la filiale, etc. Une fois en place, ces champs permettent au script ou au modèle de personnalisation de marque d'extraire toutes les informations nécessaires via une seule recherche d'enregistrement (la filiale).

- **Enregistrement de configuration de marque dédié :** Si vos besoins en matière de personnalisation de marque sont complexes ou si vous souhaitez dissocier la personnalisation de marque de la filiale légale (par exemple, une filiale gère plusieurs marques, ou vous avez des concepts de marque qui ne sont pas liés à des entités financières), un enregistrement personnalisé pourrait être préférable. Vous pourriez créer un type d'enregistrement personnalisé "Config de marque" avec des champs comme *Nom de domaine/marque (clé)*, *Fichier logo (téléchargement)*, *Couleur principale*, *Couleur secondaire*, *En-tête de modèle d'e-mail*, *Message de facture*, etc. Chaque marque (ou domaine) aurait une entrée d'enregistrement. Cet enregistrement pourrait inclure un champ le liant à une filiale (pour une recherche facile par filiale), ou vous utilisez un identifiant tel que le code de marque. L'avantage d'un enregistrement séparé est la flexibilité : vous pourriez avoir plusieurs configurations de marque par filiale ou même l'utiliser en dehors du contexte OneWorld (par exemple, différents sites web sur une seule filiale). L'inconvénient est que vous devrez vous assurer que cet enregistrement est maintenu et chargé via des scripts, car NetSuite ne l'utilise pas automatiquement n'importe où.

- **Stockage des logos et des médias :** Qu'il s'agisse d'utiliser la filiale ou un enregistrement personnalisé, les logos et autres images doivent résider dans l'armoire de fichiers. Généralement, vous téléchargez le logo de chaque marque (JPG/PNG/GIF) dans *Documents > Fichiers > Images* (ou un dossier), puis vous le sélectionnez dans le formulaire de la filiale ou l'attachez à l'enregistrement de la marque. Les formulaires NetSuite nécessitent des logos de certaines dimensions maximales (200x60 px pour les formulaires standard). Lorsque nous utilisons des modèles PDF avancés ou des scripts personnalisés, nous ferons référence à ces fichiers (par ID de fichier ou URL). Il est recommandé de stocker les logos de marque dans un dossier et une convention de nommage cohérents, et si vous utilisez des enregistrements personnalisés, de stocker peut-être l'**ID de fichier ou l'URL** dans un champ pour un accès direct.

- **Exemple – Champ Nom de marque :** Pour illustrer, considérons l'exemple précédent d'une entreprise multi-filiales : elle voulait afficher un *Nom de marque* sur les enregistrements clients et les transactions au lieu du nom officiel de l'entreprise. Initialement, elle avait un flux de travail avec 26 conditions (pour 26 filiales) pour remplir un champ personnalisé "Nom de marque" sur les clients en fonction de la filiale – un cauchemar de maintenance lors de l'ajout de nouvelles filiales. La solution a été d'ajouter un champ de liste **Nom de marque** sur l'enregistrement de la filiale elle-même, en choisissant parmi une liste personnalisée de noms de marque conviviaux. Ensuite, ils pouvaient récupérer cette valeur dynamiquement avec une seule action de flux de travail ou un script, en utilisant une formule comme `{subsidiary.custrecord_brand_name}`

pour extraire la marque de la filiale liée. Cela a éliminé des dizaines d'entrées de flux de travail et garantit que toute nouvelle filiale n'a besoin de définir ce champ qu'une seule fois, et tous les formulaires/e-mails extraient le nom correct. Cela illustre comment le stockage des données de marque à la source (filiale) et l'utilisation d'une référence dynamique améliorent la maintenabilité.

En résumé, **mettez d'abord en place votre modèle de données de marque**. Si vous avez NetSuite OneWorld, commencez par télécharger les logos pour chaque filiale et ajoutez tous les champs supplémentaires nécessaires (couleurs de marque, etc.) (Source: velosio.com). Si vous n'utilisez pas OneWorld ou si les marques transcendent les filiales, définissez un enregistrement personnalisé pour la personnalisation de marque. Remplissez ces enregistrements avec toutes les informations pertinentes pour chaque domaine. Cela fournit la base sur laquelle la logique du moteur pourra s'appuyer.

Personnalisation dynamique des formulaires avec SuiteScript (niveau UI)

Une fois les données de marque disponibles, nous nous tournons vers **SuiteScript** pour modifier dynamiquement les **formulaires de saisie et l'interface utilisateur des enregistrements** de NetSuite en fonction de ces données. SuiteScript (l'API JavaScript de NetSuite) nous permet d'injecter du contenu ou même de rediriger vers différents formulaires côté serveur. Les techniques clés incluent les **scripts d'événements utilisateur** (en particulier les événements *beforeLoad*) pour modifier les formulaires au fur et à mesure de leur chargement, et les **scripts client** pour toute modification côté client. Voici des stratégies pour utiliser SuiteScript 2.x afin d'implémenter des modifications d'interface utilisateur axées sur le domaine :

- **Événement utilisateur Before Load – Injection de logos et de styles :** Un script d'événement utilisateur sur *beforeLoad* s'exécute sur le serveur chaque fois qu'un formulaire d'enregistrement est chargé (affichage, édition ou impression). À ce stade, le script peut accéder à l'objet formulaire (`scriptContext.form`) et aux données de l'enregistrement. Nous pouvons l'utiliser pour ajouter des champs ou des messages au formulaire. Une astuce courante consiste à ajouter un **champ HTML en ligne** au formulaire qui contient du HTML/CSS/JS personnalisé – permettant ainsi l'insertion d'une bannière ou d'une image stylisée. Par exemple, pour injecter un logo personnalisé ou une bannière de marque en haut d'un formulaire de commande client, vous pourriez faire :

```
Copy

function beforeLoad(context) {
  if (context.type === context.UserEventType.VIEW || context.type === context.U
    var form = context.form;
    // Create an inline HTML field (hidden label)
    var brandField = form.addField({
      id: 'custpage_branding',
      type: 'INLINEHTML',
      label: 'Branding'
    });
    // Determine brand context (e.g. subsidiary)
    var subsId = context.newRecord.getValue({fieldId: 'subsidiary'});
    var brandConfig = loadBrandConfig(subsId); // pseudo-function to get colors
    // Build HTML content with dynamic logo and style
    var logoUrl = brandConfig.logoUrl;
    var color = brandConfig.colorHex;
    // Example HTML snippet: an image and some CSS to change form color
    var html = "<div style='padding:5px; margin-bottom:10px; border-bottom: 2px
             + "<img src='" + logoUrl + "' style='max-height:50px; vertical-ali
             + "<span style='font-size:18px; font-weight:bold; color:"+color+";
             + brandConfig.tagline + "</span></div>";
    brandField.defaultValue = html;
  }
}
```

Le pseudo-code ci-dessus, lorsqu'il est déployé sur l'enregistrement de commande client par exemple, ajouterait une bannière colorée avec le logo et le slogan de la marque chaque fois qu'un utilisateur visualise/modifie une commande client. Nous y parvenons en définissant la `defaultValue` d'un champ `inlineHTML` sur notre HTML personnalisé. L'étiquette du champ peut être masquée (par exemple, mettre l'étiquette comme un espace ou utiliser `form.setDisplayType` sur `hidden`) afin qu'elle n'affiche pas de titre de champ. Cette technique est en fait la façon dont on peut intégrer du HTML arbitraire dans les formulaires NetSuite. Un exemple de Prolecto montre l'injection de JavaScript côté client via un champ HTML en ligne – *"un champ HTML caché qui vous permet d'injecter du JavaScript basé sur le*

navigateur... qui trouvera ensuite des éléments et utilisera du CSS pour les modifier" (Source: blog.prolecto.com). Nous faisons quelque chose de similaire pour le contenu de la marque. **Note :** La position officielle de NetSuite est que la manipulation directe du DOM n'est pas prise en charge – "SuiteScript ne prend pas en charge l'accès direct à l'interface utilisateur de NetSuite via le DOM. Vous ne devez accéder à l'interface utilisateur que via les API SuiteScript". Cependant, l'utilisation de champs HTML en ligne comme ci-dessus est une méthode d'API prise en charge (puisque nous utilisons `form.addField`), et l'injection de CSS ou d'images simples est généralement sûre. Évitez simplement les hypothèses trop fragiles sur la structure de la page.

- **Exemple – Masquage/Stylisation d'éléments :** Pour illustrer davantage la puissance de l'injection de CSS/JS, imaginez que vous vouliez recolorer certains champs standard ou masquer des boutons sur les formulaires par marque. NetSuite charge jQuery par défaut dans l'interface utilisateur, de sorte que votre script injecté peut l'utiliser pour manipuler des éléments. Marty Zigman fournit un extrait pour masquer les boutons de sous-liste via des sélecteurs jQuery – par exemple, `scr += 'jQuery("#print").hide();'` pour masquer le bouton "Imprimer" (Source: blog.prolecto.com). Dans notre cas, nous pourrions de même cibler, par exemple, le titre du formulaire ou des étiquettes de champs spécifiques et appliquer du CSS (par exemple, changer la couleur ou le texte). Une autre approche pour mettre en évidence le contexte de la marque pourrait être d'ajouter une grande bannière de texte. Par exemple, si la Marque X est une filiale spécifique, nous pourrions injecter un `<h2 style="color: red;">SYSTÈME MARQUE X</h2>` HTML en haut pour tout enregistrement dans cette filiale afin de le rendre évident pour les utilisateurs. **Attention :** L'abus de hacks DOM côté client peut rendre la maintenance difficile si NetSuite met à jour son interface utilisateur ; utilisez-les judicieusement pour les améliorations que NetSuite ne prend pas en charge nativement (comme le style dynamique). Testez toujours après les mises à niveau de version de NetSuite.

- **Redirection de formulaire vers des formulaires personnalisés spécifiques :** Une autre approche SuiteScript consiste à **échanger entièrement le formulaire utilisé**, en fonction de critères. NetSuite permet plusieurs formulaires personnalisés par type d'enregistrement, et normalement le formulaire est choisi par les préférences de rôle de l'utilisateur ou les paramètres de type de formulaire. Mais avec un script, vous pouvez forcer un certain formulaire. Pourquoi faire cela ? Peut-être avez-vous conçu des mises en page de formulaire distinctes pour chaque marque (ordre différent des groupes de champs, etc.) et souhaitez-vous que l'utilisateur obtienne automatiquement la bonne sans changer manuellement. Dans un UE `beforeLoad`, si vous détectez que le formulaire actuel n'est pas celui que vous voulez, vous

pouvez utiliser le module `redirect` pour recharger l'enregistrement avec un paramètre `cf` (formulaire personnalisé) spécifié. Marty Zigman le démontre : *"if (custForm != 108) ... redirect.toRecord({ type: record.Type.CUSTOMER, id: ..., parameters: {'cf': '108'} })"*. Ici, le formulaire 108 était le formulaire souhaité. Vous pourriez mapper chaque filiale à un ID de formulaire (peut-être stocké dans une table personnalisée ou des paramètres de script) et rediriger en conséquence. Le résultat est que l'utilisateur ne voit jamais le mauvais formulaire ; le script lui sert immédiatement le bon pour cette marque. Cette méthode échange la complexité de la maintenance de nombreux formulaires (un par marque) contre une garantie d'utilisation correcte du formulaire. Si vos différences de marque sont extrêmes (mises en page de formulaire complètement différentes par marque), cela pourrait être une approche viable en combinaison avec d'autres techniques. Sinon, l'approche de personnalisation en ligne peut souvent éliminer le besoin de formulaires séparés.

- **Bannières de message d'interface utilisateur SuiteScript :** À partir de NetSuite 2018.2, il existe une méthode prise en charge pour afficher des **messages de bannière** sur les formulaires via l'API serveur `form.addPageInitMessage()` ou le module `N/ui/message` dans les scripts client. Ceux-ci apparaissent sous forme de bannières colorées et masquables en haut du formulaire – utiles pour les alertes ou les informations contextuelles. Bien que généralement utilisés pour les notifications (par exemple, « Cette commande est verrouillée car... »), on pourrait les réaffecter à des indicateurs de marque. Par exemple, une bannière d'information verte indiquant « Vous consultez : Environnement **Marque X** » pourrait être affichée sur tous les enregistrements de la Marque X. Il s'agit moins de style que de message. Steven Hall souligne que `pageInitMessage` peut être déclenché dans un UE `beforeLoad`, basé sur n'importe quelle donnée d'enregistrement. Si la logique de marque doit transmettre un message (comme une clause de non-responsabilité spécifique à une région pour les utilisateurs internes), il s'agit d'une méthode intégrée propre. Cependant, pour l'image de marque visuelle (logos/couleurs), le champ HTML intégré est plus flexible.

Illustration : Ci-dessous un exemple de champ HTML intégré utilisé pour afficher un message mis en évidence sur un formulaire (de Sikich). Dans ce cas d'utilisation, un workflow remplit le champ avec une bannière jaune si un enregistrement de facture est en attente. Le même concept peut injecter dynamiquement des bannières ou des images spécifiques à la marque :

Exemple de champ HTML intégré injectant un message coloré personnalisé sur un formulaire. Dans un moteur de branding, des champs similaires peuvent intégrer des logos ou des avis spécifiques à la marque de manière conditionnelle (par exemple, en fonction de la filiale)

L'image ci-dessus démontre la flexibilité du HTML intégré – « le HTML est très flexible. Outre le texte, vous pouvez intégrer des éléments comme des images et des GIF ». Dans notre contexte, nous pourrions intégrer la balise d'image de logo correcte pour la filiale.

- **Scripts client pour un réglage fin :** Bien que le `beforeLoad` côté serveur soit généralement suffisant, il existe des cas pour les scripts client. Par exemple, si vous devez modifier quelque chose après une interaction utilisateur ou en fonction de changements de champs de formulaire, un script client pourrait basculer des éléments de marque. Mais surtout, vous pouvez également utiliser un script client pour appliquer du CSS après le chargement de la page (puisqu'il s'exécute dans le navigateur). Si l'ID d'un élément n'est connu qu'à ce moment-là, un script client pourrait l'ajuster. Une autre astuce consiste à utiliser un **Portlet** (script de portlet de tableau de bord) ou une page **Suitelet** comme conteneur pour charger du CSS/JS personnalisé qui affecte l'interface utilisateur globalement. Certains administrateurs ont créé un « portlet de feuille de style » qui injecte une balise `<style>` dans le tableau de bord, ce qui peut remplacer certains styles globaux (bien que l'utilisation d'iframes par NetSuite puisse limiter la portée). Ce sont des techniques avancées si un thème large est requis (par exemple, changer la couleur de la barre de menu supérieure en fonction du rôle/de la marque – ce qui pourrait être fait en injectant du CSS ciblant l'en-tête). Gardez à l'esprit que NetSuite propose des **Thèmes d'interface utilisateur** dans les préférences utilisateur (Classique, Comptabilité, etc.), mais ceux-ci sont limités et non spécifiques à la marque. Ainsi, l'injection de CSS personnalisé est le seul moyen de véritablement recolorer l'interface utilisateur par marque. À utiliser avec prudence et des tests approfondis.

En résumé, **SuiteScript fournit les outils pour modifier dynamiquement les formulaires au moment du chargement** – en ajoutant du contenu de marque, en sélectionnant le formulaire approprié et en guidant l'expérience utilisateur. En utilisant les événements `beforeLoad` pour insérer des logos, des accents de couleur ou des noms de marque, vous vous assurez que, lorsque les utilisateurs naviguent dans les enregistrements, ils sont toujours conscients du contexte de la marque. Ensuite, nous verrons comment étendre ce concept aux formulaires imprimés et envoyés par e-mail aux clients.

Branding des documents client (Modèles PDF avancés et SuiteScript)

Un branding cohérent doit s'étendre aux **documents de transaction** – factures, commandes clients, bons de commande, relevés de compte client, etc. La fonctionnalité Modèles PDF/HTML avancés de NetSuite est essentielle ici, permettant des impressions hautement personnalisées avec logique. Un moteur de branding axé sur le domaine devrait permettre à un seul modèle de servir plusieurs marques en extrayant les logos et les détails corrects pour chaque domaine. Examinons comment procéder :

- **Aperçu des modèles PDF avancés :** Les modèles PDF/HTML avancés utilisent la syntaxe FreeMarker pour fusionner les données d'enregistrement dans une sortie HTML/PDF. Contrairement aux anciennes mises en page PDF, les modèles avancés sont hautement personnalisables – les développeurs peuvent modifier le HTML/CSS et inclure une logique conditionnelle, des boucles et des références à des enregistrements associés. À des fins multi-marques, un seul modèle avancé peut incorporer des conditions sur la filiale (ou un autre champ) pour modifier les logos, les titres ou même les styles. NetSuite a une nuance : lorsqu'un modèle avancé est utilisé, il **inclut automatiquement le logo de l'entreprise** par défaut (celui défini dans les Informations sur l'entreprise comme Logo de l'entreprise (Formulaires)). Dans OneWorld, cependant, vous souhaitez généralement le logo de la filiale. Heureusement, les objets `companyInformation` et `subsidiary` sont disponibles dans le modèle de données du modèle (avec quelques réserves). Un modèle de facture standard pourrait utiliser `${companyInformation.logoUrl}` pour imprimer le logo principal. Pour utiliser le logo de chaque filiale, vous pouvez le remplacer par la référence spécifique à la filiale. La documentation d'Oracle suggère de créer des modèles séparés par filiale, ou d'utiliser une petite astuce dans le code. Par exemple, le guide de Velosio suggère : *« changer* `${companyInformation.logoUrl}` *en* `${subsidiary.logo@Url}` *»* dans la source du modèle (Source: velosio.com). Cette syntaxe FreeMarker `${subsidiary.logo@Url}` récupère l'URL du fichier « Logo (Formulaires) » de la filiale. En procédant ainsi dans un modèle unifié, chaque fois que la filiale de la transaction a un fichier de logo défini, ce logo apparaîtra sur le PDF au lieu du logo parent (Source: velosio.com). Ce simple changement d'une ligne est souvent le moyen le plus facile d'obtenir des logos multi-filiales sur les documents, à condition d'avoir téléchargé les logos sur chaque filiale (Étape 1 du guide de Velosio) (Source: velosio.com). Après avoir modifié le modèle, vous l'associez à votre formulaire de transaction personnalisé et vous

assurez que le formulaire est configuré pour utiliser le PDF avancé (Source: velosio.com). Désormais, le même fichier de modèle sert toutes les filiales, échangeant les logos en fonction du contexte.

- **Personnalisation d'autres éléments de marque dans les modèles :** Au-delà des logos, vos factures ou formulaires pourraient nécessiter différentes *adresses, noms d'entreprise, numéros d'identification fiscale ou textes juridiques* par filiale/marque. NetSuite fournit l'adresse principale de la filiale et le nom légal via le contexte de l'enregistrement (pour les transactions, les champs de la filiale peuvent être accédés s'ils sont inclus via le contexte du modèle d'impression). Si certaines données ne sont pas directement disponibles, une approche consiste à les **ajouter à l'enregistrement via le sourcing** (comme l'astuce précédente consistant à sourcer les champs de la filiale sur la transaction). Par exemple, si le modèle de facture ne peut pas voir directement le numéro d'enregistrement fiscal de la filiale, vous pourriez créer un champ de transaction personnalisé qui source `{subsidiary.taxid}` sur `afterSubmit` et l'utiliser ensuite dans le modèle. Une autre approche consiste à utiliser les capacités de **rendu de SuiteScript** : `render.Transaction()` de SuiteScript peut produire des PDF et vous permet d'injecter des données personnalisées. Le Content Renderer Engine (CRE) de Marty Zigman est un exemple avancé qui joint plusieurs recherches enregistrées pour fournir des données riches aux modèles. Dans un cas, il a joint l'enregistrement de la filiale pour inclure tous ses champs dans un modèle de facture (car, par défaut, le PDF avancé n'incluait pas les champs de la filiale). L'idée générale est qu'avec la logique FreeMarker ou un script de pré-traitement, vous vous assurez que les données spécifiques de chaque marque sont présentes dans le contexte du modèle. Vous pouvez utiliser des conditions `<#if>` dans le modèle : par exemple, `<#if record.subsidiary.internalId == 3>…HTML pour la marque A…<#elseif ...>…</#if>`. Cela fonctionne mais code en dur les ID dans le modèle. Une méthode plus facile à maintenir consiste à la piloter à partir des données : par exemple, si vous aviez un champ personnalisé « Nom de marque de la filiale » (comme NomDeNotreMarque vs NomLégal), utilisez `${record.custbody_brand_name}` dans le modèle pour imprimer le nom convivial. Tant que ce champ est alimenté avec la valeur correcte (via script ou workflow) sur chaque transaction, le modèle reflétera automatiquement le nom de marque approprié, sans logique `if/else` complexe.

- **Plusieurs modèles vs un seul modèle :** Il existe deux écoles de pensée. **Un modèle dynamique unique** (avec une logique conditionnelle pour toutes les marques) signifie moins de fichiers à maintenir – une seule mise à jour s'applique à tous, assurant la cohérence. Cependant, si les designs sont radicalement différents par marque (mises en page différentes, directives de marque entièrement différentes), la logique conditionnelle peut devenir complexe.

Dans ces cas, vous pourriez opter pour **plusieurs modèles**, un par marque, puis utiliser SuiteScript ou les préférences de formulaire de NetSuite pour sélectionner le bon modèle. Par exemple, vous pourriez créer un formulaire de transaction personnalisé pour les factures de la Marque A qui pointe vers le modèle de la Marque A, un autre pour la Marque B, etc., puis utiliser un script `beforeLoad` pour changer le formulaire (comme discuté) afin que le modèle correct soit utilisé. Cette approche était celle utilisée par beaucoup pour implémenter des logos multiples avant de réaliser l'astuce du code de modèle – ils créaient des modèles de facture séparés pour chaque filiale (Source: velosio.com). Avec la solution `{subsidiary.logo@Url}`, des modèles séparés uniquement pour les différences de logo ne sont pas nécessaires ; vous n'avez besoin de modèles séparés que si la mise en page ou la formulation diffère fondamentalement. **Meilleure pratique** : visez un modèle unique chaque fois que possible, en utilisant des données dynamiques, pour minimiser la duplication. Utilisez des modèles séparés pour les designs véritablement uniques (et gérez-les via des conventions de nommage ou SDF pour le déploiement).

- **Intégration des différences régionales/linguistiques :** Si vous opérez dans différentes régions, le branding pourrait également impliquer des traductions linguistiques ou des différences de formulation juridique. Les modèles avancés de NetSuite prennent en charge l'internationalisation – ils peuvent traduire automatiquement certains champs standard, et vous pouvez créer plusieurs versions linguistiques d'un modèle. Pour un branding axé sur le domaine, envisagez de stocker tout texte spécifique à une région (comme un slogan localisé ou un pied de page réglementaire) dans l'enregistrement de configuration du branding également. Ensuite, au moment du rendu, soit l'alimenter dans l'enregistrement, soit le récupérer via SuiteScript. Par exemple, vous pourriez avoir un champ personnalisé « Message de pied de page de facture » sur la filiale. Dans le modèle, imprimez simplement la valeur de ce champ (qui varierait par filiale). Si le champ n'est pas directement accessible, vous pourriez utiliser SuiteScript dans un `beforeLoad` (ou un événement utilisateur `beforePrint`) pour le copier dans un champ de transaction personnalisé que le modèle pourra utiliser. Certaines organisations utilisent une approche hybride : des modèles séparés pour chaque langue (pour gérer les traductions de texte), mais au sein de chaque modèle, des logos et adresses dynamiques par marque. Planifiez votre approche en fonction du degré de distinction que la sortie doit avoir.

- **Utilisation du rendu SuiteScript (Avancé) :** Le module `N/render` de NetSuite vous permet de scripter la génération de PDF ou d'autres sorties par programme. Ceci est utile si vous devez produire des documents par lots ou les envoyer automatiquement par e-mail. Vous pouvez charger un modèle avancé par script, lui fournir un contexte de données personnalisé (même

des données ne provenant pas de NetSuite) et générer une sortie. Notre moteur de branding pourrait inclure un script qui, lors de l'envoi d'une transaction par e-mail, sélectionne automatiquement un modèle ou modifie même le HTML à la volée pour insérer des éléments de marque. Cependant, cela n'est généralement pas nécessaire si vous configurez les modèles comme décrit ci-dessus. Un cas d'utilisation intéressant pourrait être si les logos sont stockés en externe – par exemple, dans l'article précédent de Prolecto, ils ont récupéré les logos des clients à partir d'un service (Brandfetch) au moment de l'exécution pour les intégrer dans des propositions. Ils ont utilisé un moteur de rendu personnalisé pour effectuer un appel API pour le logo basé sur le domaine du client, puis ont inclus cette URL d'image dans la génération du PDF. Ceci dépasse les fonctionnalités standard de NetSuite mais montre les possibilités : le moteur pourrait récupérer les actifs de marque à la volée s'ils ne sont pas stockés dans NetSuite, garantissant des logos toujours à jour sans téléchargements manuels. Dans notre cadre, en supposant que les logos sont stockés dans NS, nous n'avons pas besoin d'appels externes – mais notez que SuiteScript peut appeler des API REST externes si nécessaire (avec SuiteCloud Plus ou des contextes Map/Reduce pour les processus de longue durée).

- **Test des sorties de documents :** Assurez-vous de tester minutieusement les PDF générés pour chaque marque. Chaque filiale/marque devrait avoir une transaction exemple où vous vérifiez que le logo, le nom de l'entreprise, l'adresse, etc., apparaissent correctement. Vérifiez que tout style (couleurs/polices) est esthétique – parfois le CSS d'un modèle pourrait nécessiter des ajustements si, par exemple, le logo d'une marque a un rapport d'aspect différent (vous pourriez imposer une largeur maximale en CSS pour gérer les logos plus longs). Vérifiez également les modèles d'e-mail s'ils sont utilisés : vous pourriez créer des modèles d'e-mail correspondants qui incluent le branding (comme une image d'en-tête d'e-mail). Les modèles d'e-mail NetSuite (utilisant Freemarker, similaire aux PDF) peuvent également utiliser `${subsidiary.logoUrl}` dans le script du modèle. Si vous préférez les e-mails générés par script, vous pouvez assembler un corps HTML avec l'URL du logo de marque appropriée. Un exemple de Stack Overflow a montré comment sourcer le logo dans un script pour un modèle d'e-mail, une approche similaire consistant à charger la filiale et à obtenir l'URL du fichier de logo. Le *principe est cohérent* : utilisez les données de la filiale ou de l'enregistrement de marque pour alimenter la communication.

Scénario réel : Imaginez une entreprise mondiale « Acme Corp » avec deux marques : *Acme Industrial* et *Acme Retail*, chacune étant une filiale. Les utilisateurs créent des devis et des commandes dans un seul compte NetSuite. Avec notre moteur de branding, lorsqu'un utilisateur ouvre une commande client **Retail**, le formulaire affiche le logo *Acme Retail* et peut-être un thème de couleur verte ; s'il ouvre une commande **Industrial**, il voit le logo *Industrial* et un thème bleu. Lors de l'impression ou de l'envoi par e-mail de la confirmation de commande au client, le PDF

contient automatiquement le logo correct et le texte de pied de page (« Merci d'avoir choisi Acme Retail » vs « Acme Industrial »). Tout cela se produit sans que l'utilisateur n'ait besoin de sélectionner manuellement des formulaires ou des modèles – c'est piloté par la filiale de la transaction. Si une nouvelle marque/filiale est ajoutée, l'administrateur télécharge simplement le nouveau logo et remplit les champs de branding sur l'enregistrement de la filiale ; les scripts et modèles existants l'incluent alors en référençant `${subsidiary.logo@Url}` et d'autres champs dynamiques, à condition qu'ils aient été configurés une fois (Source: velosio.com). Cela garantit la **cohérence de la marque** et fait gagner du temps. Marty Zigman souligne que *« les clients veulent un formulaire unique qui génère des présentations différentes en fonction de l'entreprise »* – notre approche combinée sur l'interface utilisateur et les modèles PDF y parvient en pratique.

Détection du domaine et du contexte utilisateur pour la logique de branding

Déterminer avec précision « quels paramètres de marque appliquer » est une partie cruciale du moteur. Nous avons abordé ce point, mais résumons les méthodes et considérations pour la **détection de contexte** dans NetSuite :

- **Par filiale (pour les transactions/entités) :** Dans OneWorld, la plupart des enregistrements de transaction et d'entité ont un champ **Filiale**. C'est généralement la clé primaire pour le contexte de la marque. Dans les scripts d'événement utilisateur, vous pouvez le récupérer de l'enregistrement (`newRecord.getValue('subsidiary')`). Dans les modèles PDF avancés, vous avez `record.subsidiary` disponible (bien que tous les champs ne soient pas exposés par défaut). Si vous avez besoin d'informations supplémentaires sur la filiale dans les modèles, vous pourriez utiliser une jointure ou un script comme décrit. Pour les scripts d'interface utilisateur internes, l'utilisation de la filiale est simple puisque l'enregistrement est déjà chargé. Une chose à considérer : les **clients ou fournisseurs multi-filiales** – si un client est lié à plusieurs filiales, la filiale sur une transaction de vente dictera le branding, et non la filiale principale du client. Utilisez donc toujours la filiale de la transaction ou celle qui est la plus pertinente pour le contexte actuel.

- **Par rôle/centre utilisateur actuel :** Certaines pages internes (comme les onglets de centre personnalisés ou les portlets de tableau de bord) peuvent ne pas être explicitement liées à une filiale. Dans de tels cas, vous pouvez vous rabattre sur le rôle ou le type de centre de l'utilisateur. Par exemple, vous pourriez avoir des rôles séparés pour l'équipe de chaque marque. Si c'est le cas, le script peut obtenir `runtime.getCurrentUser().role` (renvoie l'ID

du rôle) et décider du branding à partir de là. Alternativement, si chaque rôle est restreint à une filiale, alors la filiale de l'utilisateur (runtime) indiquera effectivement la marque. Notez que `runtime.getCurrentUser().subsidiary` donne l'ID interne de la filiale associée à l'utilisateur (généralement la filiale principale de son employé). Attention : si un administrateur a accès à toutes les filiales, cela pourrait simplement être la société mère sur son enregistrement utilisateur, et non celle avec laquelle il « travaille » actuellement. NetSuite n'a pas de concept de « contexte de filiale actuel » au-delà de l'enregistrement que vous consultez ou des restrictions de rôle. Si nécessaire, vous pourriez présenter un sélecteur de contexte – mais généralement, l'utilisation de la filiale de l'enregistrement ou d'une approche spécifique au rôle suffit.

- **Par domaine URL (Formulaires externes/Commerce) :** Si votre compte NetSuite sert des formulaires clients externes ou un site de commerce sur plusieurs domaines, vous pourriez avoir besoin de détecter le nom de domaine. Par exemple, SuiteCommerce ou la boutique en ligne de NetSuite peuvent avoir plusieurs domaines (un par pays ou par marque). Si vous utilisez les *Formulaires client en ligne* (pour la capture de leads ou de cas), malheureusement NetSuite utilise généralement une URL générique `extforms.netsuite.com` ou un domaine unique. Pour avoir véritablement des formulaires sur des domaines séparés avec un branding unique, une solution courante consiste à intégrer le formulaire dans votre site web ou à utiliser un Suitelet spécifique au domaine. Dans un script Suitelet, vous pouvez inspecter `request.headers['Host']` pour obtenir le nom d'hôte du domaine qui a été atteint, puis décider du branding. Ou passer un paramètre personnalisé dans l'URL (par exemple, `&brand=Retail`) que le script Suitelet ou UE peut lire (`scriptContext.request.parameters.brand`). Ensuite, chargez l'enregistrement de marque correspondant. Bien que cela soit avancé, c'est faisable – **particulièrement utile pour les connexions au centre client ou au centre partenaire sur différents domaines** (bien que généralement chaque centre soit de toute façon lié à une seule filiale). Il y a eu une question de la communauté sur l'utilisation de domaines personnalisés pour les formulaires externes afin de fournir une expérience de marque transparente – l'approche serait similaire : héberger le formulaire sur un domaine de marque et s'assurer que les champs du formulaire sont liés à la filiale ou à la marque correcte. En résumé, pour une utilisation externe, exploitez l'environnement (domaine, paramètres d'URL ou formulaires séparés, chacun codé pour une marque) pour savoir quelles informations de marque afficher.

- **Par classification personnalisée (Département/Classe) :** Dans certaines configurations NetSuite, *Département* ou *Classe* pourrait être utilisé pour indiquer une marque ou une ligne de produits. Si les filiales ne sont pas une option (par exemple, vous n'avez pas OneWorld mais

avez toujours plusieurs marques), vous pourriez vous baser sur un autre champ. Par exemple, un compte non-OneWorld pourrait avoir un champ personnalisé « Marque » sur les transactions. La logique du moteur de branding utilise ensuite cette valeur de champ pour sélectionner un enregistrement de configuration de marque. L'implémentation reste similaire, il suffit de remplacer les références à la filiale par le champ de marque. Assurez-vous que le champ est renseigné sur tous les enregistrements pertinents (éventuellement par défaut à partir du client ou de l'article). C'est un peu plus personnalisé mais nécessaire dans les scénarios multi-marques à entité unique.

- **Persistance du contexte :** L'interface utilisateur de NetSuite est sans état entre les chargements de page, de sorte que nos scripts s'exécutent à chaque fois pour réévaluer le contexte. Si un utilisateur change de contexte (par exemple, modifie une commande et change le champ de la filiale), un *script client* pourrait être nécessaire pour mettre à jour immédiatement l'image de marque sur le formulaire (comme changer le logo à la volée). Alternativement, vous pouvez compter sur l'utilisateur pour enregistrer et recharger l'enregistrement (le prochain beforeLoad détectera la nouvelle filiale et ajustera l'image de marque). Pour une meilleure expérience utilisateur, l'événement fieldChanged d'un script client sur la filiale pourrait effacer/définir un champ ou un message de marque pour indiquer « vous avez changé la marque – le formulaire sera mis à jour après l'enregistrement ». L'échange entièrement dynamique de tous les éléments de marque sur le formulaire sans rechargement est complexe et généralement non requis opérationnellement (car généralement, vous créez une transaction en sachant déjà sous quelle filiale vous vous trouvez).

En résumé, **déterminez le champ clé ou l'indicateur qui identifie le domaine de la marque dans chaque contexte**, et utilisez SuiteScript pour ramifier votre logique en fonction de cela. L'indicateur le plus courant et le plus fiable est l'ID de filiale pour les transactions et le rôle de l'utilisateur (ou la filiale restreinte) pour les pages non transactionnelles. Notre moteur fait essentiellement ceci :

```
let brandId;
if (record.type has subsidiary) {
    brandId = record.getValue('subsidiary');
} else {
    brandId = runtime.getCurrentUser().subsidiary;
}
```

puis utilise `brandId` pour récupérer les données de marque. Cela garantit que les attributs de marque corrects sont extraits à chaque fois.

Maintenir la cohérence entre les marques et les environnements

Lors de la mise en œuvre de plusieurs marques, un objectif majeur est la **cohérence** – à la fois au sein de chaque marque (tous les points de contact ont un aspect uniforme) et à travers le système (le moteur se comporte de manière prévisible pour toutes les marques). Il existe plusieurs bonnes pratiques et considérations pour maintenir le système à long terme :

- **Centraliser les directives de style :** Traitez vos attributs de marque comme un guide de style. Décidez d'un ensemble standard de paramètres (palette de couleurs, polices, directives d'utilisation du logo, etc.) que chaque marque aura dans la configuration. Cela rend votre moteur extensible. Par exemple, si vous ajoutez un « logo secondaire » ou une « image en filigrane » plus tard, ajoutez-le pour toutes les marques dans la structure de configuration (même si certaines ne l'utilisent pas). Cela évite le code personnalisé ponctuel pour une seule marque. Cela aide également aux tests – vous pouvez parcourir systématiquement chaque configuration de marque et vous assurer que chaque champ est rempli. Si, par exemple, une marque n'avait pas de jeu de couleurs, votre script devrait gérer cela avec élégance (peut-être utiliser une couleur d'entreprise par défaut).

- **Expérience utilisateur cohérente :** Assurez-vous que les utilisateurs internes qui travaillent avec plusieurs marques voient une structure de formulaire familière – seuls les visuels de marque diffèrent. Cette cohérence dans la mise en page des formulaires est importante afin que les utilisateurs ne soient pas confus lors du changement de contexte. Il est tentant de personnaliser fortement les formulaires par marque, mais sauf nécessité, gardez la mise en page générale et les champs uniformes ; habillez-les simplement avec l'image de marque. Cela s'aligne également avec la propre recommandation de NetSuite de standardiser les processus entre les filiales tout en permettant la localisation nécessaire. Un représentant commercial devrait suivre les mêmes étapes pour saisir une commande quelle que soit la marque, le moteur de marque confirmant simplement de quelle marque il s'agit via des logos ou des indices de couleur.

- **Gestion des éléments partagés vs distincts :** Identifiez le contenu *global* (partagé entre toutes les marques) par rapport au contenu *spécifique à la marque*. Par exemple, le texte des conditions de paiement sur les factures peut être global (identique pour toutes les filiales), alors gardez-le génériquement dans le modèle. Mais les informations de contact du service client peuvent varier selon la région/marque. Pour de tels éléments, envisagez de les stocker dans la configuration de la marque (par exemple, un champ « E-mail de support » par marque). De

cette façon, le modèle peut insérer l'e-mail de support approprié par marque dans le pied de page. Une erreur serait de coder en dur quelque chose comme support@acme.com dans le modèle alors que chaque marque a son propre e-mail de support – cela briserait la cohérence de la marque.

- **Tests en Sandbox :** Lors du développement de ce système, utilisez un environnement Sandbox NetSuite (ou Release Preview) avant de le déployer en production. La Sandbox devrait avoir une réplique des filiales et idéalement quelques enregistrements de test pour chaque marque. Téléchargez des logos de test (ils peuvent être filigranés « Test » pour éviter toute confusion avec les documents réels). Un défi est que les **ID internes** (des filiales, des champs personnalisés, etc.) peuvent différer entre la sandbox et la production. Si votre script utilise des ID internes (par exemple, si les ID internes des filiales sont différents), envisagez d'utiliser des paramètres de script ou une table de mappage qui peut être ajustée par environnement. Alternativement, utilisez des ID externes ou un champ unique pour rechercher l'enregistrement correct dans le code. Par exemple, si la filiale de la marque A a l'externalId « BrandA », votre script pourrait rechercher la filiale avec l'externalId « BrandA » plutôt que de supposer l'ID interne 3. De cette façon, les différences entre la sandbox et la production sont atténuées, tant que vous définissez les externalIds de manière cohérente. Testez également le déploiement du bundle/SDF de tous les objets personnalisés – formulaires personnalisés, déploiements de scripts, fichiers de modèles, etc., pour vous assurer que rien ne manque.

- **Promotion en Production :** Une fois satisfait, déployez la solution pendant une période de faible impact. Vérifiez les formulaires et les sorties de chaque marque en production avec un petit groupe. Il est judicieux d'informer les utilisateurs des changements (« Nous avons mis à jour le système pour afficher automatiquement les logos et les couleurs spécifiques à la marque – si quelque chose ne semble pas correct, informez le service informatique »). Ayez un plan de retour en arrière : cela pourrait simplement consister à désactiver les scripts d'événements utilisateur et à revenir aux formulaires standard si un problème critique survient. Cependant, si les tests sont approfondis, les problèmes devraient être minimes.

- **Maintenance continue :** Documentez la configuration – par exemple, « Pour une nouvelle filiale/marque, effectuez les étapes X, Y, Z : téléchargez le logo, définissez les champs de couleur, ajoutez à la liste de configuration, etc. ». Le système devrait maintenant être largement basé sur les données. Si un logo doit être mis à jour, vous mettez à jour le fichier de logo de la filiale et le tour est joué – tous les formulaires et modèles utilisant `${subsidiary.logo@Url}` afficheront le nouveau logo (aucune modification de code n'est nécessaire). Si les couleurs de la marque changent, modifiez simplement la valeur du champ personnalisé pour cette marque.

Un domaine à surveiller est celui des **modèles d'e-mail** ou de tout texte codé en dur qui pourrait encore faire référence à d'anciennes informations de marque ; essayez également de les centraliser via la configuration.

- **Surveillance et gestion des erreurs :** Implémentez une journalisation dans vos scripts. Par exemple, si le script ne parvient pas à trouver une configuration de marque pour un contexte, enregistrez une erreur (et revenez peut-être à un comportement sûr par défaut). Les journaux de script de NetSuite peuvent vous alerter de toute mauvaise configuration (comme « Aucune URL de logo trouvée pour la filiale X »). Vous pourriez même créer une recherche enregistrée de tableau de bord listant chaque filiale et si tous les champs de marque sont renseignés, pour détecter proactivement toute pièce manquante.

- **Défis :** Un défi est l'**échelle** – NetSuite OneWorld prend en charge jusqu'à 125 filiales. Si vous aviez réellement 125 marques, le moteur pourrait nécessiter une optimisation (par exemple, la mise en cache des données de configuration plutôt que d'interroger à chaque fois). SuiteScript 2.x dispose d'un module de mise en cache ou vous pouvez stocker des données fréquemment utilisées dans des enregistrements de cache personnalisés. De plus, si plusieurs utilisateurs génèrent des PDF simultanément avec des récupérations de logos externes (comme le scénario Brandfetch), tenez compte des limites de débit et implémentez la mise en cache comme l'a fait Prolecto (ils ont mis en cache l'URL du logo récupéré dans NetSuite après la première récupération). Un autre défi peut être l'**adoption par les utilisateurs** – assurez-vous que les utilisateurs savent que la sélection de formulaire est désormais automatique ; ils ne devraient pas essayer de remplacer les formulaires. Vous pourriez même masquer le sélecteur « Formulaire personnalisé » du formulaire si votre script le contrôle, pour éviter toute confusion.

- **Sécurité et autorisations :** Assurez-vous que les scripts ont les autorisations appropriées. Un script beforeLoad qui charge un enregistrement personnalisé (configuration de marque) ou un fichier peut nécessiter que le déploiement du script ait une audience ou une autorisation appropriée (les scripts s'exécutent généralement avec des privilèges d'administrateur dans les événements utilisateur, mais sinon, assurez-vous que les images du classeur sont accessibles aux rôles qui doivent les voir sur les pages). Si vous utilisez des Suitelets ou des éléments externes, assurez-vous que seuls les utilisateurs prévus peuvent y accéder (ajoutez un jeton ou rendez-les disponibles sans connexion si vraiment publics avec une considération attentive).

Enfin, gardez à l'esprit l'**objectif ultime** : un ERP unifié qui respecte toujours les identités distinctes de chaque domaine d'activité. Cela offre non seulement une image externe soignée (les clients reçoivent des factures avec le bon logo, évitant toute confusion), mais aide également en interne – cela « oriente la marque » les employés vers les processus et les données corrects. La cohérence

de la marque sur tous les canaux est un facteur connu de confiance des clients, et même la cohérence interne peut favoriser l'adoption par les utilisateurs et réduire les erreurs (par exemple, l'envoi du mauvais document de marque à un client). Notre moteur de marque axé sur le domaine dans NetSuite assure cette cohérence par conception.

Conclusion

La mise en œuvre d'un moteur de marque axé sur le domaine dans les formulaires NetSuite est un effort multidisciplinaire qui combine la **personnalisation technique de NetSuite** avec la **gestion stratégique de la marque**. En stockant de manière centralisée les attributs de marque (logos, noms, couleurs, messages) et en tirant parti de SuiteScript, des workflows et des modèles PDF avancés, les organisations peuvent automatiser l'application de l'identité de marque sur toutes les interfaces NetSuite – des formulaires à l'écran aux PDF imprimés et aux e-mails. Nous avons commencé par établir la pertinence : dans les entreprises multi-marques modernes, il est impératif qu'un ERP ne soit pas une coquille unique, mais plutôt une plateforme adaptable qui « parle » la langue et le style de chaque marque. En utilisant la plateforme robuste de NetSuite, nous avons conçu une architecture où la logique de marque est abstraite et centralisée, minimisant la duplication.

Du **côté technique**, nous avons démontré comment utiliser les **événements utilisateur beforeLoad** de SuiteScript pour injecter du contenu dynamique (comme un champ HTML avec le logo et le style d'une marque) et même rediriger vers des formulaires spécifiques à la marque si nécessaire. Nous avons également exploré une alternative sans code utilisant les **Workflows** : en créant un champ personnalisé HTML en ligne et en le définissant via des conditions de workflow, les administrateurs peuvent obtenir de simples messages ou images de marque sans écrire de script (NetSuite permet de définir les valeurs des champs HTML en ligne lors du chargement via un workflow, ce qui est un cas spécial non autorisé pour d'autres types de champs). Cela peut être une bonne option pour les besoins de base ou lorsque les ressources de codage sont limitées. Pour les documents, nous avons exploité les **modèles PDF avancés** pour échanger dynamiquement les logos en utilisant le contexte intégré (`{subsidiary.logo@Url}`) (Source: velosio.com) et pour incorporer des données spécifiques à la filiale. Nous avons fait référence à des informations d'experts montrant qu'un formulaire ou un modèle *peut* gérer plusieurs marques s'il est conçu intelligemment – réduisant considérablement les frais de maintenance.

Du **côté de la stratégie de marque**, nous avons discuté du maintien de la cohérence et de la gouvernance : s'assurer que chaque variante de marque s'aligne toujours sur les normes générales de l'entreprise et offrir une expérience utilisateur fluide. En automatisant la sélection de la marque (basée sur la filiale ou le rôle), nous éliminons les erreurs de l'utilisateur dans le choix des en-têtes ou des formulaires corrects, ce qui protège l'intégrité de la marque. Nous avons également mis en évidence des cas réels : par exemple, le renseignement d'un nom de marque convivial au lieu du nom légal pour améliorer les communications avec les clients, et la résolution du défi de la mise à l'échelle des workflows qui a été résolue en déplaçant les données dans les champs de filiale. Ces exemples soulignent l'importance de modéliser vos données et vos processus de manière évolutive.

À l'avenir, les entreprises mettant en œuvre un tel moteur devraient conserver une documentation pour l'ajout de nouvelles marques et envisager le contrôle de version de leurs scripts et modèles (SuiteCloud Development Framework – SDF – peut gérer les fichiers de script et de modèle dans un projet pour le déploiement). De cette façon, votre moteur de marque devient une fonctionnalité maintenue de votre compte NetSuite, évoluant avec votre entreprise. Il est également prudent de surveiller après la mise en service – par exemple, si un utilisateur trouve un formulaire qui n'a pas reçu l'image de marque (peut-être un type d'enregistrement de cas limite), vous pouvez mettre à jour le script pour le couvrir. Le riche écosystème de **SuiteApps et de communautés** de NetSuite signifie qu'il existe souvent des exemples de code et des solutions à exploiter (comme nous l'avons fait via des sources comme Prolecto et d'autres), mais les besoins en matière de marque de chaque entreprise sont uniques, alors adaptez le moteur à vos exigences.

En conclusion, un moteur de marque axé sur le domaine dans NetSuite est entièrement réalisable avec les outils actuels. Il met en valeur la puissance de la plateforme NetSuite : avec un peu de script et de configuration, votre ERP peut être à la fois **multi-locataire (un seul système)** et **multi-expérience (plusieurs visages de marque)**. Cela conduit à un backend unifié pour l'efficacité, tout en honorant les personas distincts de chaque marque face au client. En suivant les approches décrites – de la modélisation des données à l'injection SuiteScript et à la personnalisation des modèles – les professionnels peuvent construire une solution de marque automatisée et approfondie qui améliore à la fois l'image de l'entreprise et l'efficacité des utilisateurs.

Sources : Les techniques et les meilleures pratiques discutées sont étayées par la documentation officielle de NetSuite et les informations de la communauté d'experts. Les références clés incluent l'aide d'Oracle sur les filiales et les modèles avancés, les blogs d'experts détaillant les modèles de script pour les formulaires dynamiques et les ajustements d'interface utilisateur (Source: blog.prolecto.com), et les études de cas sur les configurations multi-filiales (Source: velosio.com). Ces sources (citées tout au long) offrent des lectures complémentaires et des exemples pour ceux

qui souhaitent approfondir leur mise en œuvre ou explorer des exemples de code spécifiques. En tirant parti de ces ressources et des directives de ce rapport, vous pouvez mettre en œuvre en toute confiance un moteur de marque axé sur le domaine qui fait de NetSuite une plateforme encore plus puissante pour votre entreprise multi-marques.

Étiquettes: netsuite, erp, suitescript, marque-axee-domaine, netsuite-oneworld, personnalisation-ui, moteur-marque, gestion-filiales

---

À propos de Houseblend

HouseBlend.io is a specialist NetSuite™ consultancy built for organizations that want ERP and integration projects to accelerate growth—not slow it down. Founded in Montréal in 2019, the firm has become a trusted partner for venture-backed scale-ups and global mid-market enterprises that rely on mission-critical data flows across commerce, finance and operations. HouseBlend's mandate is simple: blend proven business process design with deep technical execution so that clients unlock the full potential of NetSuite while maintaining the agility that first made them successful.

Much of that momentum comes from founder and Managing Partner **Nicolas Bean**, a former Olympic-level athlete and 15-year NetSuite veteran. Bean holds a bachelor's degree in Industrial Engineering from École Polytechnique de Montréal and is triple-certified as a NetSuite ERP Consultant, Administrator and SuiteAnalytics User. His résumé includes four end-to-end corporate turnarounds—two of them M&A exits—giving him a rare ability to translate boardroom strategy into line-of-business realities. Clients frequently cite his direct, "coach-style" leadership for keeping programs on time, on budget and firmly aligned to ROI.

End-to-end NetSuite delivery. HouseBlend's core practice covers the full ERP life-cycle: readiness assessments, Solution Design Documents, agile implementation sprints, remediation of legacy customisations, data migration, user training and post-go-live hyper-care. Integration work is conducted by in-house developers certified on SuiteScript, SuiteTalk and RESTlets, ensuring that Shopify, Amazon, Salesforce, HubSpot and more than 100 other SaaS endpoints exchange data with NetSuite in real time. The goal is a single source of truth that collapses manual reconciliation and unlocks enterprise-wide analytics.

Managed Application Services (MAS). Once live, clients can outsource day-to-day NetSuite and Celigo® administration to HouseBlend's MAS pod. The service delivers proactive monitoring, release-cycle regression testing, dashboard and report tuning, and 24 × 5 functional support—at a predictable monthly rate. By combining fractional architects with on-demand developers, MAS gives CFOs a scalable alternative to hiring an internal team, while guaranteeing that new NetSuite features (e.g., OAuth 2.0, AI-driven insights) are adopted securely and on schedule.

Vertical focus on digital-first brands. Although HouseBlend is platform-agnostic, the firm has carved out a reputation among e-commerce operators who run omnichannel storefronts on Shopify, BigCommerce or Amazon FBA. For these clients, the team frequently layers Celigo's iPaaS connectors onto NetSuite to

automate fulfilment, 3PL inventory sync and revenue recognition—removing the swivel-chair work that throttles scale. An in-house R&D group also publishes “blend recipes” via the company blog, sharing optimisation playbooks and KPIs that cut time-to-value for repeatable use-cases.

Methodology and culture. Projects follow a “many touch-points, zero surprises” cadence: weekly executive stand-ups, sprint demos every ten business days, and a living RAID log that keeps risk, assumptions, issues and dependencies transparent to all stakeholders. Internally, consultants pursue ongoing certification tracks and pair with senior architects in a deliberate mentorship model that sustains institutional knowledge. The result is a delivery organisation that can flex from tactical quick-wins to multi-year transformation roadmaps without compromising quality.

Why it matters. In a market where ERP initiatives have historically been synonymous with cost overruns, HouseBlend is reframing NetSuite as a growth asset. Whether preparing a VC-backed retailer for its next funding round or rationalising processes after acquisition, the firm delivers the technical depth, operational discipline and business empathy required to make complex integrations invisible—and powerful—for the people who depend on them every day.
