IBM Unica Interact
Version 8.6
25 mai 2012

Guide d'administration

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 265.

juillet 2012

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.can.ibm.com (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

© Copyright IBM Corporation 2001, 2012.

Table des matières

© Copyright IBM Corp. 2001, 2012

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	IBM Canada
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

© Copyright IBM Corp. 2001, 2012

France	Canada	Etats-Unis
(Pos1)		Home
Fin	Fin	End
(PgAr)		PgUp
(PgAv)		PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

Chapitre 1. Administration d'IBM Unica Interact

L'administration de Interact comporte plusieurs tâches. Ces tâches sont notamment les suivantes (liste non limitative) :

- Maintenance des utilisateurs et des rôles
- Maintenance des sources de données
- Configuration des fonctions facultatives de proposition d'offre dans Interact
- Contrôle et gestion des performances de l'environnement d'exécution

Avant de commencer à administrer Interact, vous devez maîtriser plusieurs grands concepts concernant le fonctionnement d'Interact pour pouvoir travailler plus facilement. Les sections suivantes décrivent les tâches d'administration associées à Interact.

La seconde partie de ce guide décrit les API (Application Programming Interfaces) disponibles avec Interact : API Interact, API ExternalCallout et API d'apprentissage.

Principaux concepts liés à Interact

Cette section décrit certains concepts clés que vous devez comprendre avant de commencer à utiliser Interact.

Référentiels

Un référentiel correspond à un ensemble d'identifiants pouvant être ciblés par une campagne. Par exemple, les référentiels d'un ensemble de campagnes peuvent être « Foyer », « Prospect », « Client » et « Compte ». Chacun de ces niveaux représente une vue particulière des données marketing disponibles pour une campagne.

Les référentiels sont en général organisés hiérarchiquement. Illustration avec les exemples précédents :

- Foyer se trouve au sommet de la hiérarchie. Chaque foyer peut inclure plusieurs clients et un ou plusieurs prospects.
- Client vient immédiatement après Foyer dans la hiérarchie. Chaque client peut disposer de plusieurs comptes.
- Compte est situé tout en bas de la hiérarchie.

Il existe d'autres types plus complexes de hiérarchies de référentiels, notamment dans les environnements business-to-business, dans lesquels des référentiels peuvent être nécessaires notamment pour les entreprises, les sociétés, les divisions, les groupes, les personnes, les comptes, etc.

Ces référentiels peuvent être liés par différents types de relations, par exemple de un à un, de un à plusieurs ou de plusieurs à plusieurs. En définissant des référentiels, vous permettez à ces concepts d'être représentés dans Campaign et aux utilisateurs de gérer les relations entre les différents référentiels à des fins de ciblage. Par exemple, bien que chaque foyer puisse compter plusieurs prospects, il peut être préférable de limiter le publipostage à un seul prospect par foyer.

© Copyright IBM Corp. 2001, 2012

Environnement de conception

L'environnement de conception est l'emplacement où vous effectuez la plus grande partie de la configuration Interact. Dans l'environnement de conception, vous définissez les événements, les points d'interaction, les segments dynamiques et les règles de traitement. Après avoir configuré ces composants, vous les déployez dans l'environnement d'exécution.

L'environnement de conception est installé avec l'application Web Campaign.

Evénements

Un événement est une action exécutée par un visiteur et qui déclenche une action dans l'environnement d'exécution : par exemple placement d'un visiteur dans un segment, présentation d'une offre ou journalisation de données.

Les événements sont d'abord créés dans un canal interactif et ensuite déclenché par un appel à l'API Interact à l'aide de la méthode `postEvent`. Un événement peut conduire à une ou plusieurs des actions suivantes définies dans l'environnement de conception Interact :

- Déclencher la resegmentation
- Journaliser le contact de l'offre
- Journaliser l'acceptation de l'offre
- Journaliser le refus de l'offre

Vous pouvez également utiliser des événements pour déclencher des actions définies par la méthode `postEvent`, incluant la journalisation des données dans une table, l'inclusion de données dans l'apprentissage ou le déclenchement de diagrammes individuels.

Les événements peuvent être organisés en catégories pour une raison de commodité dans l'environnement de conception. Ces catégories n'ont aucun rôle fonctionnel dans l'environnement d'exécution.

Canaux interactifs

Un canal interactif est une représentation dans Campaign d'un point de contact lorsque la méthode de l'interface est une boîte de dialogue interactive. Cette représentation logicielle permet de coordonner tous les objets, données et ressources de serveur nécessaires au marketing interactif.

Un canal interactif est un outil que vous utilisez pour définir des événements et des points d'interaction. Vous pouvez également accéder aux rapports d'un canal interactif à partir de l'onglet Analyse de ce canal interactif.

Les canaux interactifs contiennent également des affectations de serveurs d'exécution de production et de transfert. Vous pouvez créer plusieurs canaux interactifs pour organiser vos événements et points d'interaction si vous ne disposez que d'un jeu de serveurs d'exécution de production et de transfert, ou pour répartir vos événements et points d'interaction en fonction du système en relation avec les clients.

Diagrammes interactifs

Un diagramme interactif est associé à diagramme par lots Campaign mais présente de légères différences. Les diagrammes interactifs exécutent la même fonction principale que les diagrammes par lots, et divisent vos clients en groupes appelés

segments. Toutefois, dans le cas des diagrammes interactifs, ces groupes sont des segments dynamiques. Interact utilise ces diagrammes interactifs pour affecter un profil à un segment lorsqu'un comportemental ou système indique qu'une resegmentation du visiteur est nécessaire.

Les diagrammes interactifs contiennent un sous-ensemble des processus de diagrammes par lots ainsi que plusieurs processus qui leur sont propres.

Remarque : Les diagrammes interactifs peuvent être créés uniquement dans une session Campaign.

Points d'interaction

Un point d'interaction correspond à un emplacement de votre point de contact où vous souhaitez présenter une offre. Les points d'interaction contiennent du contenu composé d'éléments de remplissage par défaut lorsque l'environnement d'exécution ne possède pas d'autre contenu éligible à présenter.

Les points d'interaction peuvent être organisés en zones.

Offres

Une offre représente un message marketing unique pouvant être distribué de différentes façons.

Dans Campaign, vous pouvez créer des offres qui seront utilisées dans une ou plusieurs campagnes.

Une offre peut être réutilisée :

- dans plusieurs campagnes ;
- à différents moments ;
- pour différents groupes de personnes (cibles) ;
- sous forme de « versions » différentes, en modifiant simplement les zones paramétrées de l'offre.

Dans les diagrammes, vous affectez les offres aux cellules cibles à l'aide de l'un des processus de contact et vous suivez ensuite les résultats de votre campagne en capturant les données relatives aux consommateurs qui ont reçu l'offre et à ceux qui ont répondu.

Profils

Un profil correspond à un ensemble de données client utilisées par l'environnement d'exécution. Ces données peuvent être un sous-ensemble des données client disponibles dans votre base de données client, de données collectées en temps réel ou une combinaison des deux. Ces utilisation sont utilisées aux fins suivantes :

- Affectation d'un client à un ou plusieurs segments dynamiques dans des scénarios d'interaction en temps réel.

  Vous devez disposer d'un ensemble de données de profil pour chaque référentiel selon lequel vous souhaitez segmenter. Par exemple, si vous segmentez en fonction du lieu, vous pouvez choisir de n'inclure que le code postal du client parmi les informations d'adresse dont vous disposez.
- Personnalisation d'offres
- Comme attributs à suivre pour l'apprentissage

Par exemple, vous pouvez configurer Interact pour qu'il surveille l'état-civil d'un visiteur et le nombre de visiteurs correspondant à chaque état-civil qui acceptent une offre spécifique. L'environnement d'exécution peut alors exploiter ces informations pour affiner la sélection des offres.

Ces données sont en lecture seule pour l'environnement d'exécution.

Environnement d'exécution

L'environnement d'exécution se connecte à votre point de contact et effectue des interactions. Il peut être constitué d'un ou plusieurs serveurs d'exécution connectés à un point de contact.

L'environnement d'exécution utilise les informations déployés à partir de l'environnement de conception en combinaison avec l'API Interact pour présenter les offres à votre point de contact.

Sessions d'exécution

Il existe une session d'exécution sur le serveur d'exécution pour chaque visiteur de votre point de contact. Cette session contient toutes les données du visiteur que l'environnement d'exécution utilise pour affecter les visiteurs à des segments et pour recommander des offres.

Vous créez une session d'exécution à l'aide de l'appel `startSession`.

Points de contact

Un point de contact est une application ou un emplacement à partir desquels vous pouvez interagir avec un client. Un point de contact peut correspondre à un canal dans lequel le client est à l'origine du contact (interaction « entrante ») ou dans lequel vous contactez le client (interaction « sortante »). Les sites Web et les applications de centre d'appels en sont des exemples courants. Avec l'API Interact, vous pouvez intégrer Interact à vos points de contact pour présenter des offres aux clients en fonction de leur action dans le point de contact. Les points de contact sont également appelés des systèmes en relation directe avec le client (CFS).

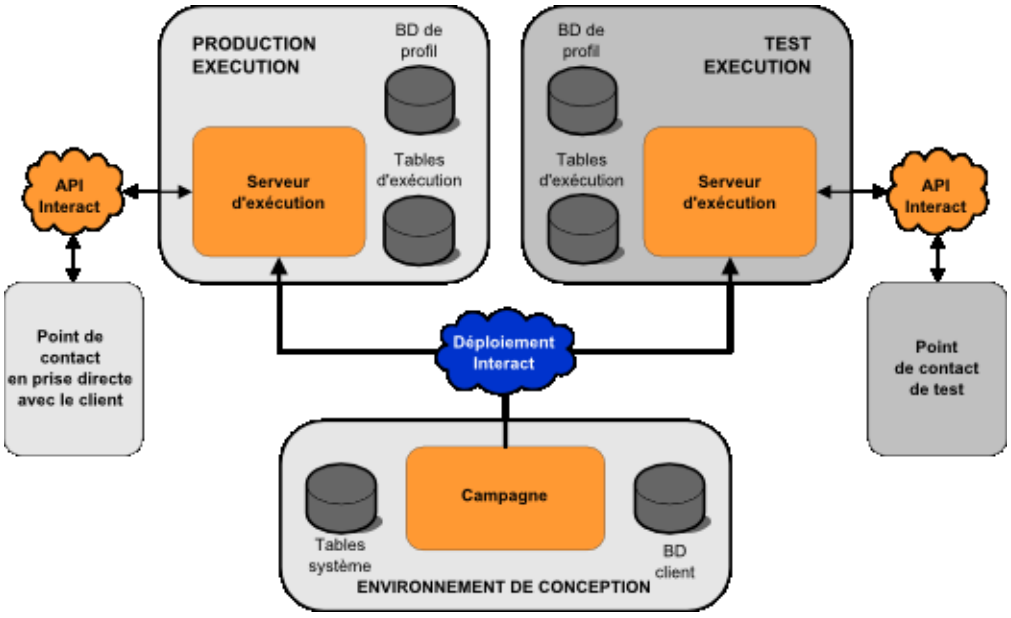
Règles de traitement

Les règles de traitement permettent d'affecter une offre à un segment dynamique. Ces affectations sont soumises à des contraintes supplémentaires imposées par la zone personnalisée que vous associez à l'offre dans la règle de traitement. Par exemple, vous pouvez affecter un ensemble d'offres à un segment dynamique dans la zone de « connexion » et un ensemble d'offres différent pour le même segment dans la zone de « l'après-vente ». Les règles de traitement sont définies dans l'onglet Stratégie d'interaction d'une campagne.

Chaque règle de traitement possède également un score marketing. Si un client est associé à plusieurs segments, et qu'il est donc concerné par plusieurs offres, les scores marketing permettent de définir l'offre qui soit être suggérée par Interact. Les offres que l'environnement d'exécution peut suggérer peuvent être influencées par un module d'apprentissage, une liste de suppression d'offres et des affectations d'offres individuelles ou globales.

Architecture Interact

L'environnement Interact comprend au moins deux composants principaux, l'environnement de conception et l'environnement d'exécution. Vous pouvez aussi avoir des serveurs d'exécution de test en option. La figure ci-après représente une vue globale de l'architecture.



L'environnement de conception est l'emplacement où vous effectuez la plupart de la configuration Interact. L'environnement de conception est installé avec l'application Web Campaign et fait référence aux tables système Campaign et vos bases de données client. Vous utilisez l'environnement de conception pour définir les points d'interaction et les événements que vous utilisez avec l'API.

Après avoir conçu et configuré la façon dont vous voulez que l'environnement d'exécution gère les interactions client, vous pouvez déployer ces données sur un groupe de serveurs de transfert à des fins de test, ou un groupe de serveurs d'exécution de production pour l'interaction client en temps réel.

L'API Interact fournit la connexion entre votre point de contact et le serveur d'exécution. Vous référencez les objets (points d'interaction et événements) créés dans l'environnement de conception à l'aide de l'API Interact et les utilisez pour demander des informations au serveur d'exécution.

Considérations réseau Interact

Une installation de production de Interact concerne au moins deux machines. Dans un environnement de production traitant de gros volumes, avec plusieurs serveurs d'exécution Interact et des bases de données distribuées, votre installation peut englober plusieurs dizaines de machines. Pour obtenir des performances optimales, vous devez prendre en compte plusieurs exigences de topologie de réseau.

- Si votre implémentation de l'API Interact démarre et termine les sessions dans le même appel, par exemple :

  ```
  executeBatch(startSession, getOffers, postEvent, endSession)
  ```

vous n'avez pas besoin d'activer la persistance des sessions (sticky sessions) entre l'équilibreur de charge et les serveurs d'exécution Interact. Vous pouvez configurer la gestion de session des serveurs d'exécution Interact pour le type de cache local.

- Si votre implémentation de l'API Interact utilise plusieurs appels pour démarrer et terminer les sessions, par exemple :

  ```
  startSession
  . . .
  executeBatch(getOffers, postEvent)
  . . .
  endSession
  ```

  et que vous utilisez un équilibreur de charge pour vos serveurs d'exécution Interact, vous devez activer une persistance quelconque pour l'équilibreur de charge (les "sticky sessions"). Si cela n'est pas possible, ou si vous n'utilisez pas d'équilibreur de charge, configurez la gestion de session des serveurs Interact pour un `cacheType` distribué. Si vous utilisez un cache distribué, tous les serveurs d'exécution Interact doivent pourvoir communiquer via la multidiffusion. Vous devrez peut-être régler votre réseau de sorte que la communication entre les serveurs Interact utilisant les mêmes adresse IP et port de multidiffusion n'entravent pas les performances du système. Un équilibreur de charge avec des sessions persistantes ("sticky sessions") donne de meilleures performances qu'un cache distribué.
- Si vous avez plusieurs groupes de serveurs utilisant une distribution `cacheType`, chacun doit utiliser un port de multidiffusion unique. L'utilisation d'un port et d'une adresse de multidiffusion uniques pour chaque groupe de serveurs est recommandée.
- Placez les serveurs Interact de votre votre environnement d'exécution, Marketing Platform, les équilibreurs de charge, et le point de contact dans le même emplacement géographique pour obtenir des performances optimales. La phase de conception et celle d'exécution peuvent se situer dans des emplacements géographiques différents, mais vous devez dans ce cas vous attendre à un déploiement lent.
- Vous devez disposer d'une connexion réseau haut débit (au moins 1 Go) entre le groupe de serveurs de production Interact et son point de contact associé.
- La phase de conception nécessite un accès http ou https pour que l'exécution termine les tâches de déploiement. Tous les pare-feux ou d'autres applications réseau doivent être configurés pour permettre le déploiement. Il peut être nécessaire d'étendre la durée du délai d'attente HTTP entre l'environnement de conception et l'environnement d'exécution si vous avez des déploiements volumineux.
- Le module de l'historique des contacts et des réponses nécessite d'accéder à la base de données de la phase de conception (tables système Campaign) ainsi que d'accéder à la base de données d'exécution (tables d'exécution Interact). Vous devez configurer vos bases de données et votre réseau de façon appropriée pour que ce transfert de données puisse se produire.

Dans une d'installation de test ou de transfert, vous pouvez installer les phases de conception et d'exécution Interact sur la même machine. Ce scénario n'est pas recommandé dans un environnement de production.

Chapitre 2. Configuration des utilisateurs IBM Unica Interact

Dans Interact, vous devez configurer deux ensembles d'utilisateurs : les utilisateurs de l'environnement d'exécution et les utilisateurs de l'environnement de conception.

- **Les utilisateurs de l'environnement d'exécution** sont créés dans le Marketing Platform qui est configuré pour fonctionner avec les serveurs d'exécution.
- **Les utilisateurs de l'environnement de conception** sont les utilisateurs de Campaign. Configurez la sécurité pour les différents membres de votre équipe de conception comme pour Campaign.

Configuration de l'utilisateur de l'environnement d'exécution

Après avoir installé Interact, vous devez configurer au moins un utilisateur Interact , l'utilisateur de l'environnement d'exécution.

L'utilisateur de l'environnement d'exécution permet d'accéder aux tables d'exécution. Il s'agit du nom d'utilisateur et du mot de passe que vous utilisez lorsque vous déployez des canaux interactifs. Le serveur d'exécution utilise l'authentification JDBC du serveur d'applications Web pour les donnée d'identification de la base de données. Vous n'avez par conséquent pas besoin d'ajouter de sources de données de l'environnement d'exécution à l'utilisateur de l'environnement d'exécution.

Important : Tous les serveurs d'exécution appartenant au même groupe de serveurs doivent partager les mêmes données d'identification utilisateur. Si vous avez des instances distinctes de Marketing Platform pour chaque serveur d'exécution, vous devez créer le même utilisateur et le même mot de passe sur chacune des instances.

Si vous utilisez un utilitaire de chargement de base de données, vous devez définir les tables d'exécution en tant que source de données avec les données d'identification de connexion pour l'utilisateur de l'environnement d'exécution. Le nom de la source de données doit être `systemTablesDataSource`.

Si vous activez la sécurité pour la surveillance JMX avec le protocole JMXMP, vous devrez peut-être faire appel à un utilisateur distinct pour la sécurité de la surveillance JMX.

Configuration des utilisateurs de l'environnement de conception

Les utilisateurs de l'environnement de conception sont les utilisateurs Campaign. Vous configurez les utilisateurs de l'environnement de conception de la même manière que vous configurez les autorisations de rôle dans Campaign.

Vous devez accorder aux utilisateurs Campaign autorisés à modifier les diagrammes interactifs le droit d'accès à la source de données des tables d'exécution de test.

Si Interact est installé et configuré, les options supplémentaires suivantes sont disponibles pour la Stratégie globale et les nouvelles stratégies. Gardez à l'esprit le

© Copyright IBM Corp. 2001, 2012

fait que certains utilisateurs de l'environnement de conception ont également besoin de certaines autorisations Campaign telles que les macros personnalisées.

Catégorie	Autorisations
Campagnes	• Afficher les stratégies d'interaction des campagnes : possibilité d'afficher, mais pas de modifier, les onglets de stratégie d'interaction dans une campagne. • Modifier des stratégies d'interaction de campagnes : possibilité de modifier les onglets de stratégie d'interaction, et notamment les règles de traitement. • Supprimer des stratégies d'interaction de campagnes : possibilité de supprimer des onglets de stratégie d'interaction des campagnes. La suppression d'un onglet de stratégie d'interaction est limitée si la stratégie d'interaction a été inclue dans un déploiement de canal interactif. • Ajouter des stratégies d'interaction de campagnes : possibilité de créer des onglets de stratégie d'interaction dans une campagne. • Lancer Lancer des déploiements de stratégie d'interaction de campagnes : possibilité de marquer un onglet de stratégie d'interaction pour le déploiement ou l'annulation du déploiement.
Canaux interactifs	• Déploiement de canaux interactifs : possibilité de déployer un canal interactif dans les environnements d'exécution Interact • Editer les canaux interactifs — possibilité de modifier l'onglet Synthèse du des canaux interactifs. • Supprimer des canaux interactifs : possibilité de supprimer des canaux interactifs. La suppression des canaux interactifs est limitée si le canal interactif a été déployé. • Afficher des canaux interactifs : possibilité d'afficher, mais pas de modifier les canaux interactifs. • Ajouter des canaux interactifs : possibilité de créer des canaux interactifs. • Afficher des rapports sur les canaux interactifs : possibilité d'afficher l'onglet d'analyse du canal interactif. • Ajouter des objets enfants aux canaux interactifs : possibilité d'ajouter des points d'interaction, des zones, des événements et des catégories.

Catégorie	Autorisations
Sessions	• Afficher des diagrammes interactifs : possibilité d'afficher un diagramme interactif dans une session. • Ajouter des diagrammes interactifs : possibilité de créer des diagrammes interactifs dans une session. • Modifier des diagrammes interactifs : possibilité de modifier des diagrammes interactifs. • Supprimer des diagrammes interactifs : possibilité de supprimer des diagrammes interactifs. La suppression des diagrammes interactifs est limitée si le canal interactif auquel le diagramme interactif a affecté a été déployé. • Copier des diagrammes interactifs : possibilité de copier des diagrammes interactifs. • Tester des diagrammes interactifs : possibilité de lancer l'exécution de test pour un diagramme interactif. • Réviser des diagrammes interactifs : possibilité d'afficher un diagramme interactif et d'ouvrir les processus pour voir les paramètres, mais pas d'effectuer des modifications. • Déployer des diagrammes interactifs : possibilité de marquer des diagrammes interactifs pour le déploiement ou l'annulation du déploiement.

Exemple de permissions dans un environnement de conception

Par exemple, vous pouvez créer deux rôles, un pour les personnes créant des diagrammes interactifs et un pour les personnes définissant les stratégies d'interaction. Chaque section liste les permissions accordées au rôle.

Rôle du diagramme interactif

Macro personnalisée

- Ajouter des macros personnalisées
- Modifier des macros personnalisées
- Utiliser des macros personnalisées

Zone dérivée

- Ajouter des champs dérivés
- Modifier des champs dérivés
- Utiliser des champs dérivés

Modèle de diagramme

- Coller des modèles

Modèle de segment

- Ajouter des segments
- Modifier des segments

Session

- Afficher la synthèse des sessions
- Afficher des diagrammes interactifs

- Ajouter des diagrammes interactifs
- Modifier des diagrammes interactifs
- Copier des diagrammes interactifs
- Tester des diagrammes interactifs
- Déployer des diagrammes interactifs

Rôle de stratégie d'interaction

Campagne
- Afficher synthèse campagne
- Mettre à jour les populations ciblées pour une campagne
- Afficher les stratégies d'interaction des campagnes
- Modifier des stratégies d'interaction de campagnes
- Ajouter des stratégies d'interaction de campagnes
- Lancer des déploiements de stratégie d'interaction de campagne

Offre
- Afficher le résumé des offres

Modèle de segment
- Afficher la synthèse des segments

Session
- Réviser des diagrammes interactifs

Chapitre 3. Gestion des sources de données Interact

Interact nécessite plusieurs sources de données pour fonctionner correctement. Certaines sources de données contiennent les informations dontInteract a besoin pour fonctionner, d'autres sources de données contiennent vos données.

Les sections suivantes décrivent les sources de données Interact, y compris les informations dont vous avez besoin pour les configurer correctement, et des suggestions pour leur maintenance.

Gestion des sources de données Interact

Interact nécessite plusieurs ensembles de données pour fonctionner.

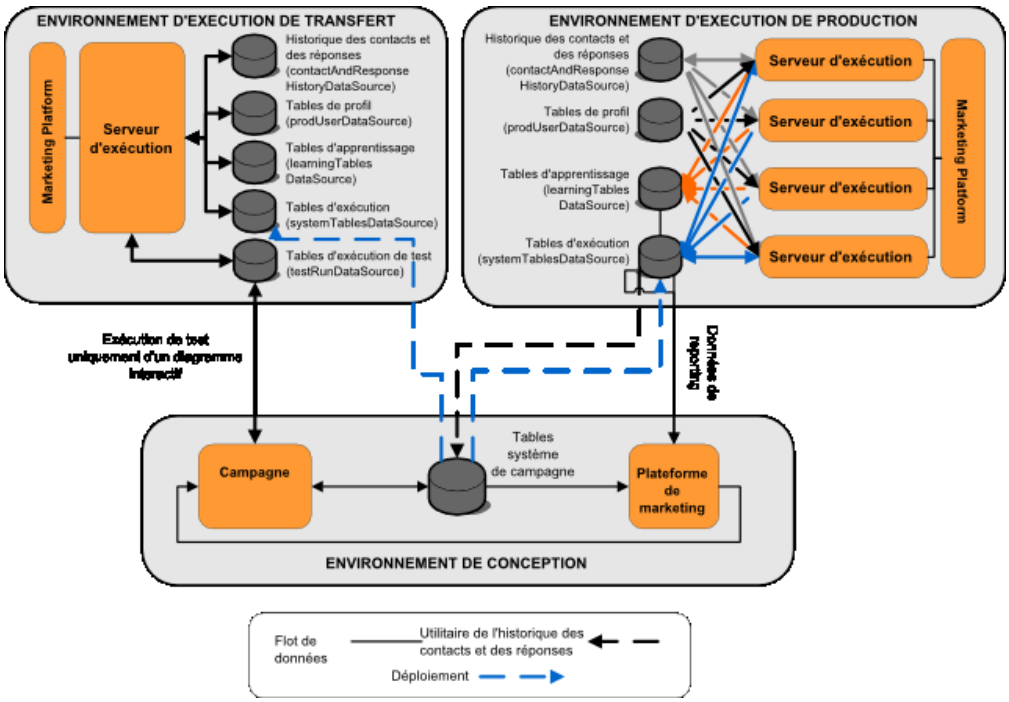
- **Tables système Campaign** — En plus de toutes les données de Campaign, les tables système Campaign contiennent des données des composants Interact que vous créez dans l'environnement de conception, tels que les règles de traitement et les canaux interactifs. L'environnement de conception et les tables système Campaign partagent la même base de données physique et le même schéma.
- **Tables d'exécution** — (`systemTablesDataSource`) Contiennent les données de déploiement de l'environnement de conception, les tables de transfert de l'historique des contacts et des réponses, et les statistiques d'exécution.
- **Tables de profil** — (`prodUserDataSource`) contiennent des informations, en plus des données client collectées en temps réel, requises par les diagrammes interactifs pour placer correctement les visiteurs dans des segments intelligents. Si vous faites appel entièrement aux données en temps réel, vous n'avez pas besoin de tables de profil. Si vous utilisez des tables de profil, vous devez disposer d'au moins une table de profil public par référentiel utilisé par le canal interactif.

  Les tables de profil peuvent également contenir les tables utilisées pour enrichir la proposition d'offres, y compris les tables de suppression d'offre, la substitution de score, et affectation d'offre globale et individuelle.
- **Tables d'exécution de test** — (`testRunDataSource`) Contiennent un exemple de toutes les données requises par les diagrammes interactifs pour placer les visiteurs dans des segments, y compris les données imitant les éléments collectés en temps réel pendant une interaction. Ces tables sont nécessaires pour le groupe de serveurs désigné en tant que groupe de serveurs d'exécution de test pour l'environnement de conception uniquement.
- **Tables d'apprentissage** — (`learningTablesDataSource`) Contiennent toutes les données collectées par l'utilitaire d'apprentissage intégré. Ces tables peuvent inclure une table qui définit les attributs dynamiques. Si vous n'utilisez pas l'apprentissage ou utilisez un utilitaire d'apprentissage externe que vous créez, vous n'avez pas besoin de tables d'apprentissage.
- **Historique des contacts et des réponses pour les réponses inter-sessions** — (`contactAndResponseHistoryDataSource`) Soit les tables de l'historique Campaign, soit une copie de ces dernières. Si vous n'utilisez pas la fonction de réponse inter-sessions, vous n'avez pas besoin de configurer ces tables d'historique des contacts.

© Copyright IBM Corp. 2001, 2012

Bases de données et applications

Le diagramme suivant illustre les sources de données Interact possibles et leurs liens aux applications IBM® Unica.



- Campaign et le groupe de serveur d'exécution de test accèdent aux tables d'exécution de test.
- Les tables d'exécution de test sont utilisées pour tester les exécutions de diagramme interactif uniquement.
- Lorsque vous utilisez un serveur d'exécution pour tester un déploiement, y compris les API Interact de, le serveur d'exécution utilise les tables de profil pour les données.
- Si vous configurez le module d'historique des réponses et des contacts, le module utilise un processus ETL (extraction, transformation et chargement) en arrière-plan pour transférer les données des tables de transfert d'exécution vers les tables de l'historique des réponses et des contacts Campaign.
- La fonction de génération de rapports interroge les données à partir des tables d'apprentissage, des tables d'exécution et des tables système Campaign pour afficher des rapports dans Campaign.

Vous devez configurer les environnements d'exécution de test à utiliser un autre ensemble de tables que vos environnement d'exécution de production. Avec les tables distinctes pour le transfert et la production, vous pouvez conserver vos résultats de test séparément de vos résultats réels. Le module de l'historique des contacts et des réponses insère toujours ces données dans les tables de l'historique des contacts et des réponses de Campaign (Campaign ne comporte pas de tables de l'historique des contacts et des réponses pour le test). Si vous avez des tables d'apprentissage distinctes pour l'environnement d'exécution de test, et vous souhaitez voir les résultats dans des rapports, vous avez besoin d'une instance distincte d'IBM Cognos BI pour exécuter les rapports d'apprentissage de l'environnement de test.

Tables système Campaign

Lorsque vous installez l'environnement de conception, vous pouvez également créer de nouvelles tables spécifiques à Interact dans les tables système Campaign.

Si vous activez le module d'historique des réponses et des contacts, le module copie l'historique des réponses et des contacts depuis les tables de transfert des tables d'exécution vers les tables de l'historique des réponses et des contacts dans les tables système Campaign. Les tables par défaut sont `UA_ContactHistory`, `UA_DtlContactHist` et `UA_ResponseHistory`, mais le module d'historique des réponses et des contacts utilise n'importe lesquelles des tables qui sont mappées dans Campaign pour les tables de l'historique des réponses et des contacts

Si vous utilisez les tables d'offres globales et les tables de substitution de score pour affecter des offres, vous devrez peut-être remplir la table `UACI_ICBatchOffers` dans les tables système Campaign si vous utilisez des offres non contenues dans les règles de traitement du canal interactif.

Tables d'exécution

Si plusieurs référentiels sont définis, vous devez créer des tables de transfert pour les données de l'historique des contacts et des réponses pour chaque référentiel.

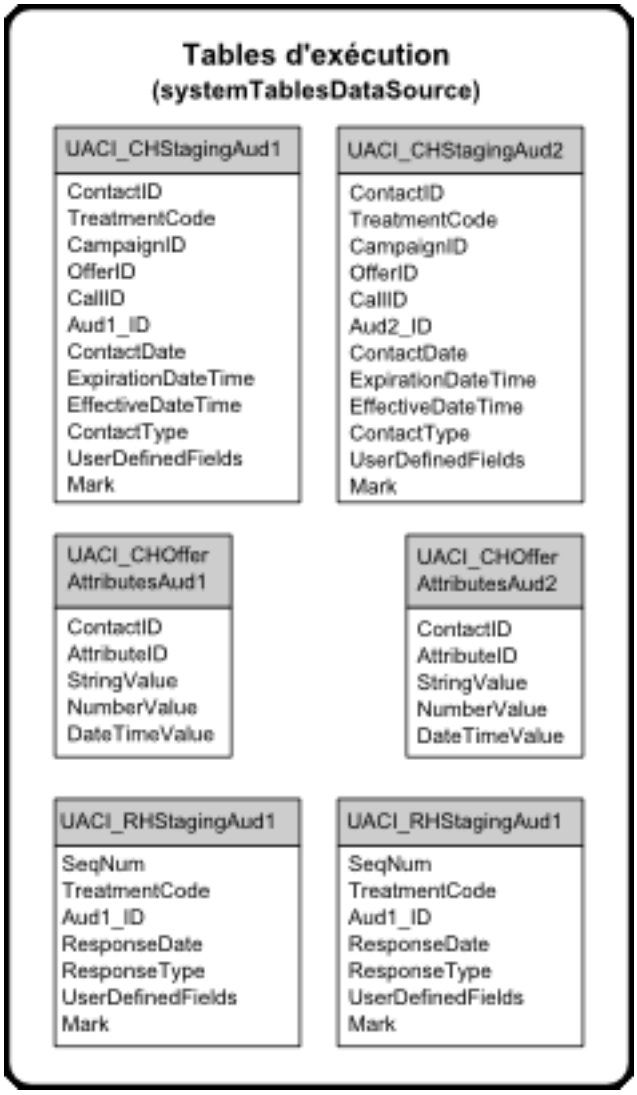
Les scripts SQL créent les tables suivantes pour le référentiel par défaut :

- `UACI_CHStaging`
- `UACI_CHOfferAttrib`
- `UACI_RHStaging`

Vous devez créer des copies de ces trois tables pour chacun de vos référentiels dans les tables d'exécution.

Si vos tables d'historique des contacts et de réponse Campaign comportent des zones définies par l'utilisateur, vous devez créer les mêmes noms et types de zones dans les tables `UACI_CHStaging` et `UACI_RHStaging`. Vous pouvez remplir ces zones lors de l'exécution en créant des paires nom/valeur du même nom dans les données de session. Par exemple, vos tables de l'historique des contacts et des réponses contiennent la zone `catalogID`. Vous devez ajouter la zone `catalogID` aux tables `UACI_CHStaging` et `UACI_RHStaging`. Ensuite, l'API Interact renseigne cette zone en définissant un paramètre d'événement en tant que paire valeur-nom nommée `catalogID`. Les données de session peuvent être fournies par la table de profil, les données temporelles, l'apprentissage, ou l'API Interact.

Le diagramme suivant illustre les tables d'exemple pour les référentiels Aud1 et Aud2.Ce diagramme ne contient pas toutes les tables de la base de données d'exécution.



Toutes les zones des tables sont obligatoires. Vous pouvez modifier `IDConsommateur` et `UserDefinedFields` en fonction de vos tables de l'historique des contacts et des réponses Campaign.

Tables d'exécution de test

Les tables d'exécution de test sont utilisés pour tester des diagrammes interactifs uniquement. Les exécutions de test des diagrammes interactifs doivent tester votre logique de segmentation. Il vous suffit de configurer une seule base de données d'exécution de test pour votre installation Interact. Les tables d'exécution de test n'ont pas besoin de se trouver dans une base de données autonome. Vous pourriez, par exemple, utiliser vos tables de données client pour Campaign.

L'utilisateur de la base de données associé aux tables d'exécution de test doivent disposer des privilèges CREATE pour ajouter les tables de résultats d'exécution de test.

La base de données d'exécution de test doit contenir toutes les tables mappées dans le canal interactif.

Ces tables doivent contenir des données permettant d'exécuter les scénarios que vous voulez tester dans vos diagrammes interactifs. Par exemple, si votre logique de tri des diagrammes interactifs permet de trier les personnes en segments basés sur le choix effectué dans un système de messagerie vocale, vous devez avoir au moins une ligne pour chaque sélection possible. Si vous créez une interaction qui fonctionne avec un formulaire sur votre site Web, vous devez inclure des lignes représentant les données manquantes ou syntaxiquement incorrectes, par exemple `nom@domainecom` pour la valeur d'une adresse électronique.

Chaque table d'exécution de test doit contenir au moins la liste des ID du référentiel approprié, et une colonne représentant les données en temps réel que vous prévoyez d'utiliser. Comme les exécution de test n'ont pas accès aux données en temps réel, vous devez fournir des données exemple pour chaque élément de données en temps réel attendu. Par exemple, si vous souhaitez utiliser des données pouvant être collectées en temps réel, telles que le nom de la dernière page Web visitée, qui est stocké dans l'attribut `lastPageVisited`, ou le nombre d'articles d'un panier d'achat, qui est stocké dans l'attribut `shoppingCartItemCount`, vous devez créer des colonnes portant les mêmes noms, et remplir les colonnes avec des données exemple. Ceci vous permet de tester l'exécution des branches de votre logique de diagramme dont la nature est contextuelle ou comportementale.

Les exécutions de test des diagrammes interactifs ne sont pas optimisées pour traiter des ensembles de données volumineux. Vous pouvez limiter le nombre de lignes utilisées pour l'exécution de test dans le processus d'interaction. Toutefois, le résultat est que le premier ensemble de lignes toujours sélectionné. Pour vous assurer que différents ensembles de lignes sont sélectionnés, utilisez différentes vues des tables d'exécution de test.

Pour tester les performances de débit des diagrammes interactifs dans l'environnement d'exécution, vous devez créer un environnement d'exécution de test, y compris une table de profil pour l'environnement de test.

Dans la pratique, vous pouvez avoir besoin de trois ensembles de tables pour les tests : une table d'exécution de test pour tester les diagrammes interactifs, des tables de profil de test pour le groupe de serveurs de test, et un ensemble de tables de profil de production.

Substitution des types de données par défaut utilisés pour les tables créées dynamiquement

L'environnement d'exécution Interact crée de manière dynamique les tables dans deux scénarios distincts : lors de l'exécution d'un test d'un diagramme et lors de l'exécution d'un processus Extraction qui écrit dans une table n'existant pas déjà. Pour créer ces tables, Interact s'appuie sur des types de données codés en dur pour chaque type de base de données pris en charge.

Vous pouvez substituer les types de données par défaut en créant une table de types de données de remplacement, nommée `uaci_column_types`, dans `testRunDataSource` ou `prodUserDataSource.` Cette table supplémentaire permet à Interact de gérer les cas rares non pris en charge par les types de données codés en dur.

Lorsque la table `uaci_column_types` est définie, Interact utilise les métadonnées pour les colonnes en tant que types de données à utiliser pour toute génération de

table. Si la table uaci_column_types n'est pas définie, ou s'il existe des exceptions rencontrées lors de la tentative de lecture de la table, les types de données par défaut sont utilisés.

Au démarrage, le système d'exécution vérifie d'abord le testRunDataSource pour la table uaci_column_types. Si la tableuaci_column_types n'existe pas dans testDataSource, ou si prodUserDataSource est d'un autre type de base de données, Interact vérifie alors le prodUserDataSource de la table.

Remplacement des types de données par défaut

Procédez comme suit pour remplacer les types de données par défaut des tables créées dynamiquement.

1. Créez une table dans TestRunDataSource ou ProdUserDataSource avec les propriétés suivantes :

   Nom de table : uaci_column_types

   Noms de colonnes :

   - uaci_float
   - uaci_number
   - uaci_datetime
   - uaci_string

   Définissez chaque colonne à l'aide du type de données approprié pris en charge par votre base de données.
2. Redémarrez le serveur d'exécution pour permettre à Interact de reconnaître la nouvelle table.

   Important : Le serveur d'exécution doit être redémarré à chaque fois que des modifications sont apportées à la table uaci_column_types.

Types de données par défaut pour les tables créées dynamiquement

Le tableau suivant répertorie les types de données codées en dur que le système d'exécution Interact utilise par défaut pour chaque base de données prise en charge pour les colonnes à virgule flottante, numériques, de date/heure et de chaîne.

Tableau 1. Types de données par défaut pour les tables créées dynamiquement

Base de données	Types de données par défaut
DB2	• float • bigint • timestamp • varchar
Informix	• float • int8 • DATETIME YEAR TO FRACTION • char2

Tableau 1. Types de données par défaut pour les tables créées dynamiquement (suite)

Base de données	Types de données par défaut
Oracle	• float • number(19) • timestamp • varchar2
SQL Server	• float • bigint • datetime • nvarchar

Base de données de profil

Le contenu de la base de données de profil dépend entièrement des données dont vous avez besoin pour configurer vos diagrammes interactifs et l'API Interact. Interact exige ou recommande que chaque base de données contienne certaines tables ou données.

La base de données de profil doit contenir ce qui suit :

- Toutes les tables mappées dans le canal interactif.

  Ces tables doivent contenir toutes les données requises pour exécuter vos diagrammes interactifs en production. Ces tables doivent être à plat, rationalisées et correctement indexées. Comme il existe un coût en termes de performances pour accéder aux données dimensionnelles, vous devez utiliser un schéma non normalisé chaque fois que possible. Au minimum, vous devez indexer la table de profil dans les zones d'ID de référentiel. S'il existe d'autres zones extraites des tables dimensionnelles, elles doivent être indexées de manière à réduire les temps d'extraction de la base de données. Les ID de référentiel des tables de profil doivent correspondre aux ID de référentiel définis dans Campaign.
- Si vous définissez la propriété de configuration `enableScoreOverrideLookup` sur true, vous devez inclure une table de substitution de score pour au moins un référentiel. Vous définissez les noms de table de substitution de score avec la propriété `scoreOverrideTable`.

  La table de substitution de score peut contenir des associations client-offre individuelles. Vous pouvez créer une table de substitution de score exemple, `UACI_ScoreOverride` en exécutant le script SQL `aci_usertab` sur votre base de données de profil. Vous devez également indexer cette table sur la colonne ID de référentiel.

  Si vous définissez propriété `enableScoreOverrideLookup` sur la valeur false, vous n'avez pas besoin d'inclure une table de substitution de score.
- Si vous définissez la propriété de configuration `enableDefaultOfferLookup` sur true, vous devez inclure la table des offres globales (`UACI_DefaultOffers`). Vous pouvez créer la table des offres globales en exécutant le script SQL `aci_usertab` sur votre base de données de profil.

  La table des offres globales peut contenir des associations client-offre.
- Si vous définissez la propriété de configuration `enableOfferSuppressionLookup` sur true, vous devez inclure une table de suppression d'offre pour au moins un référentiel. Vous pouvez définir les noms de table de suppression d'offre avec la propriété `offerSuppressionTable`.

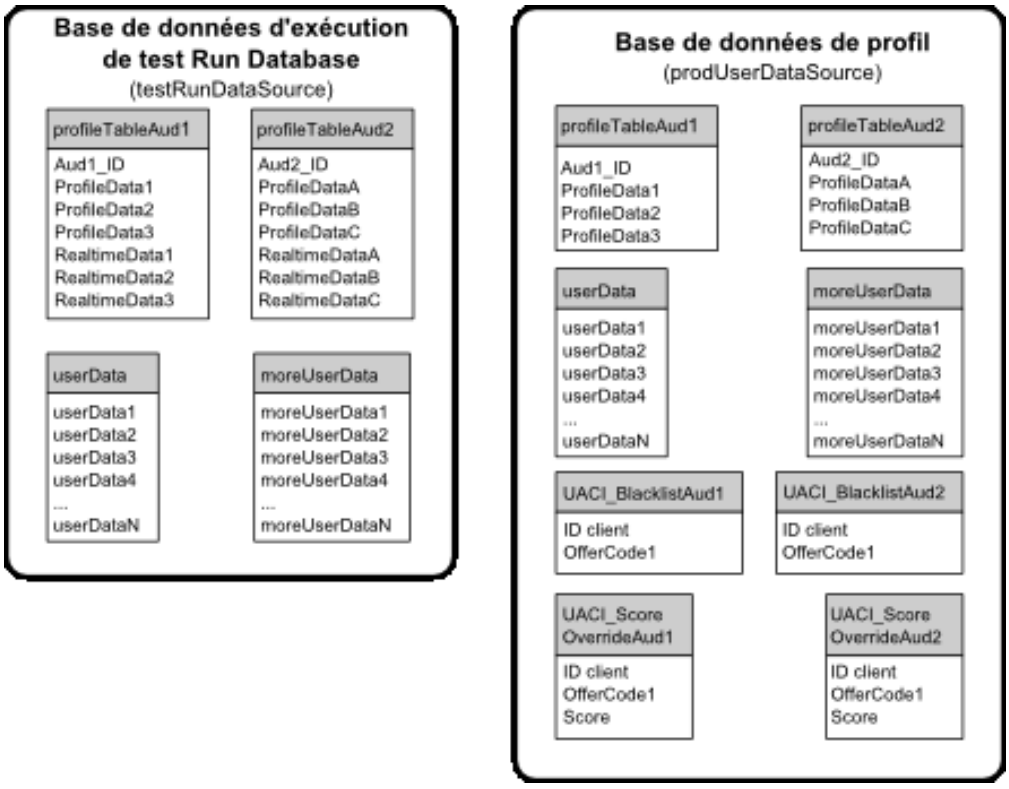
La suppression de la table d'offre peut contenir une ligne pour chaque offre supprimée pour un membre du référentiel, même si une entrée n'est pas requise pour tous les membres. Vous pouvez créer un exemple de table de suppression d'offre, `UACI_BlackList` en exécutant le script SQL `aci_usertab` sur votre base de données de profil.

Si vous définissez propriété `enableOfferSuppressionLookup` sur la valeur false, vous n'avez pas besoin d'inclure une table de suppression de score.

Une grande quantité de données dans l'une des tables peut dégrader les performances. Pour obtenir des résultats optimaux, placez des index appropriés sur les colonnes de référentiel pour les tables utilisées lors de l'exécution et qui comportent de grandes quantités de données.

Toutes les propriétés de configuration référencées ci-dessus se trouvent dans la catégorie **Interact > profil** ou **Interact > profil > Référentiels > *AudienceLevel***. Le script SQL `aci_usertab` se trouve dans le répertoire `ddl` dans votre répertoire d'installation de l'environnement d'exécution.

Le diagramme suivant illustre les tables d'exemple pour l'exécution de test et les bases de données de profil pour les référentiels Aud1 et Aud2.



Tables d'apprentissage

Si vous utilisez l'apprentissage intégré Interact, vous devez configurer les tables d'apprentissage. Ces tables contiennent toutes les données exploitées par la fonction d'apprentissage intégré.

Si vous utilisez des attributs d'apprentissage dynamique, vous devez renseigner la table `UACI_AttributeList`.

L'apprentissage implique l'écriture dans des tables de transfert intermédiaire et l'agrégation des informations à partir des tables de transfert vers les tables d'apprentissage. Les propriétés de configuration `insertRawStatsIntervalInMinutes` et `aggregateStatsIntervalInMinutes` dans la catégorie `Interact > offerserving > Built-in Learning Config` déterminent la fréquence de remplissage des tables d'apprentissage.

L'attribut `insertRawStatsIntervalInMinutes` détermine la fréquence à laquelle les informations d'acceptation et de contact de chaque client et de chaque offre sont déplacées de la mémoire vers les tables de transfert, `UACI_OfferStatsTX` et `UACI_OfferAllTx`. Les informations stockées dans les tables de transfert sont agrégées et déplacées vers les tables`UACI_OfferStats` et`UACI_OfferStatsAll` à intervalles réguliers qui sont déterminés par la propriété de configuration `aggregateStatsIntervalInMinutes`.

L'apprentissage intégré Interact utilise ces données afin de calculer les scores définitifs des offres.

Historique des contacts pour le suivi des réponses intersessions

Si vous activez la fonction de réponse inter-sessions, l'environnement d'exécution nécessite un accès en lecture seule aux tables de l'historique des contacts Campaign. Vous pouvez configurer l'environnement d'exécution pour afficher les tables système Campaign, ou vous pouvez créer une copie des tables d'historique des contacts de Campaign. Si vous créez une copie des tables, vous devez gérer le processus de mise à jour de la copie. Le module de l'historique des contacts et des réponses ne met pas à jour la copie des tables de l'historique des contacts.

Vous devez exécuter le script SQL `aci_crhtab` sur ces tables de l'historique des contacts pour ajouter des tables requises pour la fonction de suivi des réponses inter-sessions.

Utilisation des scripts de fonction Interact

Plusieurs des fonctions optionnelles disponibles avec Interact nécessitent des modifications de certaines tables dans vos bases de données de profil. Votre installation Interact, à la fois les environnement de conception et d'exécution, inclut des scripts de fonction ddl. Ces scripts ajoutent des colonnes spécifiques requises pour votre table.

Pour activer l'une de ces fonctions, exécutez le script approprié sur la base de données ou la table appropriée.

`dbType` est le type de base de données, par exemple `sqlsvr` pour Microsoft SQL Server.

Nom de fonction	Script de fonction	A exécuter sur	Changer
Offres globales, suppression d'offres et substitution de score	Répertoire d'installation de l'environnement d'exécution `\ddl\aci_usrtab_`*`dbType`*`.sql`	Votre base de données de profil (userProdDataSource)	Crée les tables `DefaultOffers`, `UACI_BlackList` et `UACI_ScoreOverride`.

Nom de fonction	Script de fonction	A exécuter sur	Changer
Score	Répertoire d'installation de l'environnement d'exécution \ddl\acifeatures\ aci_scoringfeature_*dbType*.sql	Tables de substitution de score dans votre base de données de profil (userProdDataSource)	Ajoute les colonnes LikelihoodScore et AdjExploreScore.
Apprentissage	Répertoire d'installation de l'environnement de conception \ddl\acifeatures\ aci_lrnfeature_*dbType*.sql	Base de données Campaign contenant les tables de l'historique des contacts	Ajoute la colonne RTSelectionMethod à la table UA_DtlContactHist.

A propos du suivi de l'historique des contacts et des réponses

Vous pouvez configurer l'environnement d'exécution afin qu'il enregistre l'historique des contacts et des réponses dans les tables de l'historique des contacts et des réponses de Campaign. Les serveurs d'exécution stockent l'historique des contacts et des réponses dans des tables de transfert. Le module de l'historique des contacts et des réponses copie ces données depuis les tables de transfert vers les tables de l'historique des contacts et des réponses de Campaign.

Le module de l'historique des contacts et des réponses fonctionne uniquement si vous définissez les propriétés interactInstalled et contactAndResponseHistTracking > isEnabled de la page de Configuration de l'environnement de conception sur yes.

Si vous utilisez le module de suivi des réponses intersessions, le module d'historique des contacts et des réponses est une entité distincte.

Configuration des types de contact et de réponse

Vous pouvez enregistrer un type de contact et deux types de réponses avec Interact, comme indiqué dans le tableau ci-dessous. Toutes ces propriétés se trouvent dans la catégorie contactAndResponseHistTracking.

Événement	Type de contact/réponse	Propriété de configuration
Journaliser le contact de l'offre	Contact	contacté
Journaliser l'acceptation de l'offre	Réponse	accepter
Journaliser le refus de l'offre	Réponse	rejet

Vous pouvez également enregistrer d'autres types de réponse personnalisés à l'aide de la méthode postEvent.

Vous devez également vous assurer que la colonne CountsAsResponse de la table UA_UsrResponseType dans les tables système Campaign est correctement configurée. Tous ces types de réponse doivent exister dans la table UA_UsrResponseType.

Pour que l'entrée soit valide dans UA_UsrResponseType, vous devez définir une valeur pour toutes les colonnes de la table, y compris CountsAsResponse. Les valeurs valides de CountsAsResponse sont 0, 1, ou 2. 0 indique l'absence de réponse, 1 indique une réponse, et 2 indique un refus. Ces réponses sont utilisées pour la génération de rapports.

Autres types de réponse

Dans Interact, vous pouvez utiliser la méthode postEvent dans l'API Interact API pour déclencher un événement qui journalise une action "accepter" ou "refuser" pour une offre. Vous pouvez également étendre le système pour permettre à l'appel `postEvent` d'enregistrer des réponses supplémentaires, telles que Explorer, Considérer, Valider, ou Réaliser. Tous ces types de réponse doivent exister dans la table `UA_UsrResponseType` dans les tables système Campaign. L'utilisation de paramètres d'événements spécifiques à la méthode `postEvent` vous permet d'enregistrer des types de réponse supplémentaires et de définir si une acceptation doit être inclue dans l'apprentissage.

Pour journaliser des types de réponse supplémentaires, vous devez ajouter les paramètres d'événement suivants :

- **UACIRESPONSETYPECODE** — chaîne représentant un code de type de réponse. La valeur doit être une entrée valide de la table `UA_UsrResponseType.`

  Pour que l'entrée dans `UA_UsrResponseType` soit valide, vous devez définir toutes les colonnes de la table, y compris `CountsAsResponse`. Les valeurs valides de `CountsAsResponse` sont 0, 1, ou 2. 0 indique l'absence de réponse, 1 indique une réponse, et 2 indique un refus. Ces réponses sont utilisées pour la génération de rapports.
- **UACILOGTOLEARNING** — Nombre ayant la valeur 1 ou 0. 1 indique que Interact doit consigner l'événement comme une acceptation pour l'apprentissage. 0 indique que Interact ne doit pas consigner l'événement pour l'apprentissage. Ce paramètre vous permet de créer plusieurs méthodes `postEvent` qui consignent différents types de réponse sans influence sur l'apprentissage. Si vous ne définissez pas `UACILOGTOLEARNING`, Interact considère que la valeur par défaut est 0.

Il peut être utile de créer plusieurs événements avec l'action Journaliser l'acceptation de l'offre : soit un événement par type de réponse à journaliser, soit un événement unique avec l'action Journaliser l'acceptation de l'offre utilisée pour chaque appel `postEvent` journalisant des types de réponse différents.

Par exemple, créez un événement avec l'action Journaliser l'acceptation de l'offre pour chaque type de réponse. Vous définissez les réponses personnalisées suivantes dans la table `UA_UsrResponseType` [sous le format Nom (code)] : Exploree (EXP), Considérer (CON), et Valider (CMT). Vous créez ensuite trois événements et les nommez LogAccept_Explore, LogAccept_Consider et LogAccept_Commit. Tous les trois événements sont exactement les mêmes (ils utilisent l'action Journaliser l'acceptation de l'offre), mais leurs noms sont différents afin que la personne travaillant avec l'API puisse les distinguer entre eux.

Vous avez également la possibilité de créer un événement unique avec l'action Journaliser l'acceptation de l'offre utilisée pour tous les types de réponses personnalisées. Par exemple, appelez-le LogCustomResponse.

Lorsque vous travaillez avec l'API, il n'y a aucune différence fonctionnelle entre les événements, mais les conventions de dénomination peuvent rendre le code plus clair. Par ailleurs, si vous donnez un nom distinct à chaque réponse personnalisée, le rapport Récapitulatif d'activité des événements du canal affiche les informations de façon plus précise.

Commencez par définir toutes les paires valeur-nom.

```
//Define name value pairs for the UACIRESPONSETYPECODE
// Response type Explore
NameValuePair responseTypeEXP = new NameValuePairImpl();
responseTypeEXP.setName("UACIRESPONSETYPECODE");
responseTypeEXP.setValueAsString("EXP");
responseTypeEXP.setValueDataType(NameValuePair.DATA_TYPE_STRING);

// Response type Consider
NameValuePair responseTypeCON = new NameValuePairImpl();
responseTypeCON.setName("UACIRESPONSETYPECODE");
responseTypeCON.setValueAsString("CON");
responseTypeCON.setValueDataType(NameValuePair.DATA_TYPE_STRING);

// Response type Commit
NameValuePair responseTypeCMT = new NameValuePairImpl();
responseTypeCMT.setName("UACIRESPONSETYPECODE");
responseTypeCMT.setValueAsString("CMT");
responseTypeCMT.setValueDataType(NameValuePair.DATA_TYPE_STRING);

//Define name value pairs for UACILOGTOLEARNING
//Does not log to learning
NameValuePair noLogToLearning = new NameValuePairImpl();
noLogToLearning.setName("UACILOGTOLEARNING");
noLogToLearning.setValueAsString("0");
noLogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

//Logs to learning
NameValuePair LogToLearning = new NameValuePairImpl();
LogToLearning.setName("UACILOGTOLEARNING");
LogToLearning.setValueAsString("1");
LogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

Ce premier exemple montre l'utilisation des événements individuels.

```
//EXAMPLE 1: This set of postEvent calls use the individually named events
//PostEvent with an Explore response
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogAccept_Explore, postEventParameters);

//PostEvent with a Consider response
NameValuePair[] postEventParameters = { responseTypeCON, noLogToLearning };
response = api.postEvent(sessionId, LogAccept_Consider, postEventParameters);

//PostEvent with a Commit response
NameValuePair[] postEventParameters = { responseTypeCOM, LogToLearning };
response = api.postEvent(sessionId, LogAccept_Commit, postEventParameters);
```

Ce deuxième exemple illustre l'utilisation d'un seul événement.

```
//EXAMPLE 2: This set of postEvent calls use the single event
//PostEvent with an Explore response
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogCustomResponse, postEventParameters);

//PostEvent with a Consider response
NameValuePair[] postEventParameters = { responseTypeCON, noLogToLearning };
response = api.postEvent(sessionId, LogCustomResponse, postEventParameters);

//PostEvent with a Commit response
NameValuePair[] postEventParameters = { responseTypeCOM, LogToLearning };
response = api.postEvent(sessionId, LogCustomResponse, postEventParameters);
```

Les deux exemples exécutent exactement les mêmes actions, mais une version peut être plus facile à lire que l'autre.

Mappage des tables de transfert de l'environnement d'exécution avec les tables d'historique Campaign

Les tableaux suivants illustrent comment mapper les tables de transfert de l'environnement d'exécution avec les tables d'historique Campaign. N'oubliez pas que vous devez disposer de l'une de ces tables pour chaque référentiel. Les noms de table affichés sont les exemples de tables créés pour le référentiel par défaut dans les tables d'exécution et les tables système Campaign.

Tableau 2. Historique des contacts

UACI_CHStaging **Nom de colonne dans la table de transfert de l'historique des contacts Interact**	**Table de l'historique des contacts Campaign**	**Nom de colonne de la table**
ID de contact	S/O	S/O
TreatmentCode	UA_Treatment	TreatmentCode
CampaignID	UA_Treatment	CampaignID
OfferID	UA_Treatment	OfferID
CellID	UA_Treatment	CellID
CustomerID	UA_DtlContactHist	CustomerID
ContactDate	UA_DtlContactHist	ContactDateTime
ExpirationDateTime	UA_Treatment	ExpirationDateTime
EffectiveDateTime	UA_Treatment	EffectiveDateTime
ContactType	UA_DtlContactHist	ContactStatusID
UserDefinedFields	UA_DtlContactHist	UserDefinedFields

`ContactID` est une clé qui permet de joindre `UACI_CHOfferAtrib` avec `UACI_CHStaging`.

Tableau 3. Attributs de l'offre

UACI_CHOfferAttrib **Nom de colonne dans la table de transfert de l'historique des contacts Interact**	**Table de l'historique des contacts Campaign**	**Nom de colonne de la table**
ID de contact	S/O	S/O
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

`ContactID` est une clé qui permet de joindre `UACI_CHOfferAtrib` avec `UACI_CHStaging`.

Tableau 4. Historique des réponses

UACI_RHStaging **Nom de colonne dans la table de transfert de l'historique des réponses Interact**	**Table de l'historique des réponses Campaign**	**Nom de colonne de la table**
SeqNum	S/O	S/O

Tableau 4. Historique des réponses (suite)

UACI_RHStaging **Nom de colonne dans la table de transfert de l'historique des réponses Interact**	**Table de l'historique des réponses Campaign**	**Nom de colonne de la table**
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime
Type de réponse	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

`SeqNum` est une clé utilisée par le module de l'historique des contacts et des réponses pour identifier les données, mais il n'est pas enregistré dans les tables Campaign.

La colonne `userDefinedFields` peut contenir les données de votre choix. Si vous ajoutez des colonnes aux tables de transfert, le module de l'historique des contacts et des réponses les écrit dans les tables `UA_DtlContactHist` ou `UA_ResponseHistory` dans les colonnes du même nom. Par exemple, si vous ajoutez la colonne `linkFrom` à la table`UACI_CHStaging`, le module d'historique des contacts et des réponses copiera ces données dans la colonne `linkFrom` de la table `UA_DtlContactHist`.

Important : Si vous avez d'autres colonnes dans vos vos tables d'historique de contacts et de réponses Campaign, vous devez ajouter des colonnes correspondantes aux tables de transfert avant d'exécuter le module de l'historique des contacts et des réponses.

Vous renseignez les colonnes supplémentaires des tables de transfert en créant des colonnes portant le même nom que vos paires nom/valeur dans vos données de session d'exécution. Par exemple, si vous créez les paires nom/valeur `NumberItemsInWishList` et `NumberItemsInShoppingCart`, lorsqu'un événement Journaliser l'acceptation de l'offre ou Journaliser le refus de l'offre se produit, si les colonnes `NumberItemsInWishList` et `NumberItemsInShoppingCart` existent dans votre table `UACI_RHStaging`, l'environnement d'exécution renseigne ces zones. L'environnement d'exécution remplit la table `UACI_CHStaging` lorsqu'un événement Journaliser le contact de l'offre survient.

Vous pouvez utiliser ces zones définies par l'utilisateur pour inclure le score utilisé pour présenter une offre. Ajoutez une colonne appelée `FinalScore` à la table `UACI_CHStaging` dans les tables d'exécution et à la table `UA_DtlContactHist` dans les tables système Campaign. Interact complète automatiquement la colonne `FinalScore` en y insérant le score final utilisé pour l'offre si vous utilisez un apprentissage intégré.

Si vous créez un module d'apprentissage personnalisé, vous pouvez utiliser la méthode `setActualValueUsed` de l'interface `ITreatment` et de la méthode `logEvent` de l'interface `ILearning`.

Si vous n'utilisez pas l'apprentissage, ajoutez une colonne appelée `Score` à la table `UACI_CHStaging` dans les tables d'exécution et à la table `UA_DtlContactHist` dans les tables système Campaign. Interact complète automatiquement la colonne `Score` en y insérant le score utilisé pour l'offre.

Configuration de la surveillance JMX pour le module de l'historique des contacts et des réponses

Dans Marketing Platform, pour l'environnement de conception, éditez les propriétés de configuration suivantes dans la catégorie `Campaign > monitoring`.

Propriété de configuration	Paramètre
`monitorEnabledForInteract`	**True**
port	Numéro de port du service JMX
protocole	**JMXMP** ou **RMI** La sécurité n'est pas activée pour le module d'historique des réponses et des contacts, même si vous sélectionnez le protocole JMXMP.

Lorsque vous affichez les données de l'historique des contacts et des réponses dans votre outil de surveillance JMX, les attributs sont organisés d'abord par la partition puis par référentiel.

L'adresse par défaut pour la surveillance du r le module d'historique des contacts et des réponses avec le protocole JMXMP est : `service:jmx:jmxmp://`*`CampaignServer`*`:`*`port`*`/campaign`.

L'adresse par défaut pour la surveillance du module d'historique des contacts et des réponses avec le protocole RMI est : `service:jmx:rmi:///jndi/rmi://`*`CampaignServer`*`:`*`port`*`/campaign`.

A propos du suivi des réponses inter-sessions

Les visiteurs ne peuvent pas toujours terminer une transaction au cours d'une seule visite à votre point de contact. Un client peut ajouter un article à son panier sur votre site Web et ne terminer la vente que deux jours plus tard. Il n'est pas possible de garder la session d'exécution active indéfiniment. Vous pouvez activer le suivi de réponse inter-sessions pour effectuer le suivi d'une présentation d'offre dans une session puis le faire correspondre avec une réponse dans une autre session.

Le suivi de réponses inter-sessions Interact peut correspondre à des codes de traitement ou des codes d'offre par défaut. Vous pouvez également le configurer de sorte qu'il corresponde à un code personnalisé de votre choix. La réponse inter-sessions établit une correspondance d'après les données disponibles. Par exemple, votre site Web inclut une offre avec un code promotionnel généré au moment de l'affichage et garantissant une remise pendant une semaine. Un utilisateur peut ajouter des articles à son chariot, mais ne terminer l'achat que trois jours plus tard. Lorsque vous utilisez l'appel `postEvent` pour journaliser un événement d'acceptation, vous pouvez inclure uniquement le code promotionnel. Etant donné que l'environnement d'exécution ne parvient pas à trouver un code de traitement ou d'offre correspondant dans la session en cours, l'exécution place l'événement d'acceptation avec les informations disponibles dans une table de transfert de la réponse inter-sessions (XSessResponse). Le service CrossSessionResponse lit régulièrement la table XSessResponse et tente d'établir une correspondance avec les enregistrements des données de l'historique de contact disponible. Le service CrossSessionResponse fait correspondre le code promotionnel avec l'historique des contacts et collecte toutes les données

nécessaires pour journaliser une réponse adéquate. Le service CrossSessionResponse écrit alors la réponse dans les tables de transfert de réponse, et si l'apprentissage est activé, dans les tables d'apprentissage. Le module de l'historique des contacts et des réponses écrit alors ces données dans les tables de l'historique des contacts et des réponses de Campaign.

Configuration de la source de données du suivi des réponses inter-sessions

Le suivi des réponses inter-sessions Interact fait correspondre les données de session de l'environnement d'exécution avec l'historique des contacts et des réponses Campaign. Par défaut, le suivi des réponses inter-sessions recherche des correspondances en fonction des codes de traitement ou des codes d'offre. Vous pouvez configurer l'environnement d'exécution afin de rechercher une correspondance avec un code personnalisé de remplacement.

- Si vous choisissez de correspondance avec un code de remplacement, vous devez définir ce code de remplacement dans la table `UACI_TrackingType` dans les tables d'exécution Interact.
- L'environnement d'exécution doit avoir accès aux tables de l'historique des contacts Campaign. Vous pouvez configurer l'environnement d'exécution afin qu'il accède aux tables de l'historique des contacts Campaign, ou vous pouvez créer une copie des tables de l'historique des contacts dans l'environnement d'exécution.

  L'accès est en lecture seule et est distinct de l'utilitaire de l'historique des contacts et des réponses.

  Si vous créez une copie des tables, vous devez gérer le processus garantissant l'exactitude des données dans la copie de l'historique des contacts. Vous pouvez configurer la durée pendant laquelle le service CrossSessionResponse conserve les réponses sans correspondance afin qu'elle corresponde à la fréquence à laquelle vous régénérez les données de la copie des tables d'historique des contacts à l'aide de la propriété `purgeOrphanResponseThresholdInMinutes`.Si vous utilisez le module de l'historique des contacts et des réponses, vous devez coordonner l'ETL pour avoir la certitude de disposer des données les plus à jour.

Configuration des tables de l'historique des contacts et des réponses pour le suivi des réponses inter-sessions

Lorsque vous créez une copie des tables de l'historique des contacts, ou lorsque vous utilisez les tables réelles dans les tables système Campaign, vous devez procéder comme suit.

1. Les tables d'historique des contacts et des réponses doivent être mappées correctement dans Campaign.
2. Vous devez exécuter le script SQL `aci_lrnfeature` dans le répertoire `interactDT/ddl/acifeatures` dans le répertoire d'installation de l'environnement de conception Interact sur les tables `UA_DtlContactHist` et `UA_ResponseHistory` dans vos tables système Campaign.

   Cette action ajoute la colonne `RTSelectionMethod` aux tables `UA_DtlContactHist` et `UA_ResponseHistory`. Exécutez le script `aci_lrnfeature` sur ces tables pour chacun de vos référentiels. Editez le script si nécessaire pour exploiter la table correcte pour chacun de vos référentiels.
3. Si vous envisagez de copier les tables de l'historique des contacts dans l'environnement d'exécution, faites-le maintenant.

4. Exécutez le script SQL aci_crhtab dans le répertoire ddl dans le répertoire d'installation de l'environnement de conception Interact sur la source de données de l'historique des contacts et des réponses.

   Ce script crée les tables UACI_XsessResponse et UACI_CRHTAB_Ver.
5. Créez une version de la table UACI_XsessResponse pour chaque référentiel.

Si vous créez une copie des tables de l'historique des contacts Campaign de accessibles à l'environnement d'exécution pour le suivi des réponses inter-sessions, appliquez les directives suivantes :

- Le suivi de réponse inter-sessions nécessite un accès en lecture seule à ces tables.
- Le suivi de réponse inter-sessions nécessite les tables suivantes dans l'historique des contacts Campaign.
  - UA_DtlContactHist (pour chaque référentiel)
  - UA_Treatment

  Vous devez mettre à jour les données de ces tables sur une base régulière pour assurer un suivi exact des réponses.

Pour améliorer les performances du suivi des réponses inter-sessions, il peut être nécessaire de limiter la quantité des données de l'historique des contacts, soit par la façon dont vous copiez les données de l'historique des contacts, soit en configurant une vue dans les tables d'historique des contacts Campaign. Par exemple, si votre pratique commerciale est de ne jamais proposer une durée de validité d'offre supérieure à 30 jours, vous devez limiter les données d'historique de contact aux 30 derniers jours.

Vous ne verrez pas les résultats du suivi des réponses inter-sessions tant que le module de l'historique des contacts et des réponses ne se sera pas exécuté. Par exemple, processSleepIntervalInMinutes est 60 minutes. Par conséquent, il faut au moins une heure avant que les réponses inter-sessions apparaissent dans votre historique des réponses Campaign.

Table UACI_TrackingType

La table UACI_TrackingType fait partie des tables de l'environnement d'exécution. Cette table définit les codes de suivi utilisés avec le suivi des réponses inter-sessions. Le code de suivi définit la méthode utilisée par l'environnement d'exécution pour faire correspondre à l'offre en cours dans une session d'exécution avec l'historique des contacts et des réponses.

Colonne	Type	Description
TrackingCodeType	int	Nombre qui définit le type de code de suivi. Ce nombre est référencé par les commandes SQL utilisées pour faire correspondre l'information des données de session avec les tables de l'historique des contacts et des réponses.
Name	varchar(64)	Nom du type de code de suivi. Il est transmis aux données de session à l'aide du paramètre réservé UACI_TrackingCodeType avec la méthode postEvent.
Description	varchar(512)	Brève description du type de code de suivi. Cette zone est facultative.

Par défaut, l'environnement d'exécution comporte deux types de code de suivi, comme indiqué dans le tableau ci-dessous. Pour tout code de remplacement, vous devez définir un TrackingCodeType unique.

TrackingCodeType	Nom	Description
1	Code de traitement	Code de traitement généré par UACI
2	Code offre	Code d'offre de campagne UAC

UACI_XSessResponse

Une instance de cette table pour chaque référentiel doit exister dans la source de données de l'historique des contacts et des réponses disponibles pour le suivi des réponses inter-sessions Interact.

Colonne	Type	Description
SeqNumber	bigint	Identificateur de la ligne des données. Le service CrossSessionResponse traite tous les enregistrements dans l'ordre SeqNumber.
ICID	bigint	ID de canal interactif
AudienceID	bigint	ID de ce référentiel. Le nom de cette colonne doit correspondre à l'ID de référentiel défini dans Campaign. L'exemple de table contient la colonne CustomerID.
TrackingCode	varchar(64)	La valeur transmise par le paramètre UACIOfferTrackingCode de la méthode postEvent.
TrackingCodeType	int	Représentation numérique d code de suivi. La valeur doit être une entrée valide de la table UACI_TrackingType.
OfferID	bigint	Id de l'offre défini dans Campaign.
Type de réponse	int	Type de réponse de cet enregistrement. La valeur doit être une entrée valide de la table UA_UsrResponseType.
ResponseTypeCode	varchar(64)	Code du type de réponse de cet enregistrement. La valeur doit être une entrée valide de la table UA_UsrResponseType.
ResponseDate	datetime	Date de la réponse.

Colonne	Type	Description
Mark	bigint	La valeur de cette zone identifie l'état de l'enregistrement. • 1 — En cours • 2 — Réussi • NULL — Nouvel essai • -1 — L'enregistrement est dans la base de données depuis plus de purgeOrphanResponseThresholdInMinutes minutes. Dans le cadre de la maintenance de l'administrateur de base de cette table, vous pouvez vérifier cette zone pour y rechercher les enregistrements n'ayant pas de correspondance, c'est-à-dire tous les enregistrements ayant la valeur -1. Tous les enregistrements contenant la valeur 2 sont automatiquement supprimés par le service CrossSessionResponse.
UsrDefinedFields	char(18)	Toutes les zones personnalisées à inclure lors de la mise ne correspondance des réponses aux offres avec l'historique des contacts et des réponses. Par exemple, si vous voulez établir une correspondance avec un code promotionnel, incluez une zone de code promotionnel définie par l'utilisateur.

Activation du suivi des réponses inter-sessions

Vous devez configurer le module de l'historique des contacts et des réponses afin qu'il tire pleinement parti du suivi des réponses inter-sessions.

Pour utiliser le suivi des réponses inter-sessions, vous devez configurer l'environnement d'exécution afin qu'il ait un accès en lecture-écriture aux tables de l'historique des contacts et des réponses Campaign. Vous pouvez effectuer une lecture depuis les tables de l'historique des contacts et des réponses Campaign dans l'environnement de conception, ou à partir d'une copie des tables dans les sources de données de l'environnement d'exécution. Cela est distinct de toute configuration du module de l'historique des contacts et de réponses.

Si vous recherchez une correspondance sur un autre élément que le code de traitement ou le code de l'offre, vous devez l'ajouter à la table UACI_TrackingType.

1. Créez les tables XSessResponse dans les tables de contacts et de réponses accessibles à l'environnement d'exécution.
2. Définissez les propriétés dans la catégorie contactAndResponseHistoryDataSource de l'environnement d'exécution.
3. Définissez la propriété crossSessionResponseTable de chaque référentiel.
4. Créez une catégorie OverridePerAudience pour chaque référentiel.

Mise en correspondance d'offre de réponse intersessions

Par défaut, le suivi des réponses intersessions recherche des correspondances en fonction des codes de traitement ou des codes d'offre. Le service crossSessionResponse utilise des commandes SQL pour établir les correspondances avec les codes de traitement, les codes d'offre ou un code personnalisé fourni par les données de session au contact Campaign contact et aux tables de d'historique

de réponses. Vous pouvez éditer ces commandes SQL pour faire correspondre les personnalisations que vous apportez à vos codes de suivi, vos codes d'offre ou vos codes personnalisés.

Correspondance par code de traitement

Le SQL avec lequel établir une correspondance par code de traitement doit renvoyer toutes les colonnes dans la table XSessResponse pour ce référentiel, plus une colonne appelée `OfferIDMatch`. La valeur de la colonne `OfferIDMatch` doit être l'`offerId` qui accompagne le code de traitement dans l'enregistrement XSessResponse.

Voici un exemple de commande SQL générée par défaut qui établit une correspondance avec les codes de traitement. Interact génère le SQL qui permet d'utiliser les noms de table corrects pour le référentiel. Ce SQL est utilisé si la propriété `Interact > services > crossSessionResponse > OverridePerAudience > AudienceLevel > TrackingCodes > byTreatmentCode > SQL` est définie sur **Utiliser System Generated SQL**.

```
select   distinct treatment.offerId as OFFERIDMATCH,
   tx.*,
            dch.RTSelectionMethod
from     UACI_XSessResponse tx
Left Outer Join UA_Treatment treatment ON tx.trackingCode=treatment.treatmentCode
Left Outer Join UA_DtlContactHist dch ON tx.CustomerID = dch.CustomerID
Left Outer Join UA_ContactHistory ch ON tx.CustomerID = ch.CustomerID
AND treatment.cellID = ch.cellID
AND treatment.packageID=ch.packageID
where   tx.mark=1
and       tx.trackingCodeType=1
```

Les valeurs `UACI_XsessResponse`, `UA_DtlContactHist`, `CustomerID`, et `UA_ContactHistory` sont définies par vos paramètres dans Interact. Par exemple, `UACI_XsessResponse` est défini par la propriété de configuration `Interact > profil > Référentiels > [AudienceLevelName] > crossSessionResponseTable`.

Si vous avez personnalisé vos tables d'historique de contact et de réponse, il peut être nécessaire de réviser ce SQL pour qu'il fonctionne avec vos tables. Vous pouvez définir des substitutions SQL dans la propriété `Interact > services > crossSessionResponse > OverridePerAudience > (AudienceLevel) > TrackingCodes > byTreatmentCode > OverrideSQL`. Si vous fournissez un SQL de substitution, vous devez également remplacer la propriété `SQL` par **Effacer SQL**.

Correspondance par code d'offre

Le SQL avec lequel établir une correspondance par code d'offre doit renvoyer toutes les colonnes de la table XSessResponse pour ce référentiel, plus une colonne appelée `TreatmentCodeMatch`. La valeur de la colonne `TreatmentCodeMatch` est le code de traitement associé à l'ID de l'offre (et le code de l'offre) dans l'enregistrement XSessResponse.

Voici un exemple de commande SQL générée par défaut qui établit une correspondance avec les codes d'offre. Interact génère le SQL qui permet d'utiliser les noms de table corrects pour le référentiel. Ce SQL est utilisé si la propriété `Interact > services > crossSessionResponse > OverridePerAudience > AudienceLevel > TrackingCodes > byOfferCode > SQL` est définie sur **Utiliser System Generated SQL**.

```
select   treatment.treatmentCode as TREATMENTCODEMATCH,
   tx.*,
dch.RTSelectionMethod
from     UACI_XSessResponse tx
Left Outer Join UA_DtlContactHist dch ON tx.CustomerID=dch.CustomerID
Left Outer Join UA_Treatment treatment ON tx.offerId = treatment.offerId
Left Outer Join
   (
   select   max(dch.contactDateTime) as maxDate,
      treatment.offerId,
      dch.CustomerID
   from     UA_DtlContactHist dch, UA_Treatment treatment, UACI_XSessResponse tx
   where  tx.CustomerID=dch.CustomerID
   and tx.offerID = treatment.offerId
   and dch.treatmentInstId = treatment.treatmentInstId
   group by dch.CustomerID,  treatment.offerId
   ) dch_by_max_date ON tx.CustomerID=dch_by_max_date.CustomerID
    and tx.offerId = dch_by_max_date.offerId
where   tx.mark = 1
and       dch.contactDateTime = dch_by_max_date.maxDate
and       dch.treatmentInstId = treatment.treatmentInstId
and       tx.trackingCodeType=2
union
select   treatment.treatmentCode as TREATMENTCODEMATCH,
   tx.*,
   0
from UACI_XSessResponse tx
Left Outer Join UA_ContactHistory ch ON tx.CustomerID =ch.CustomerID
Left Outer Join UA_Treatment treatment ON tx.offerId = treatment.offerId
Left Outer Join
   (
   select   max(ch.contactDateTime) as maxDate,
      treatment.offerId, ch.CustomerID
   from UA_ContactHistory ch, UA_Treatment treatment, UACI_XSessResponse tx
   where  tx.CustomerID =ch.CustomerID
   and tx.offerID = treatment.offerId
   and treatment.cellID = ch.cellID
   and treatment.packageID=ch.packageID
   group by ch.CustomerID,  treatment.offerId
   ) ch_by_max_date ON tx.CustomerID =ch_by_max_date.CustomerID
   and tx.offerId = ch_by_max_date.offerId
   and treatment.cellID = ch.cellID
   and treatment.packageID=ch.packageID
where   tx.mark = 1
   and       ch.contactDateTime = ch_by_max_date.maxDate
   and       treatment.cellID = ch.cellID
   and       treatment.packageID=ch.packageID
   and       tx.offerID = treatment.offerId
   and       tx.trackingCodeType=2
```

Les valeurs UACI_XsessResponse, UA_DtlContactHist, CustomerID, et UA_ContactHistory sont définies par vos paramètres dans Interact. Par exemple, UACI_XsessResponse est défini par la propriété de configuration Interact > profil > Référentiels > [AudienceLevelName] > crossSessionResponseTable.

Si vous avez personnalisé vos tables d'historique de contact et de réponse, il peut être nécessaire de réviser ce SQL pour qu'il fonctionne avec vos tables. Vous pouvez définir des substitutions SQL dans la propriété Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byOfferCode > OverrideSQL. Si vous fournissez un SQL de substitution, vous devez également remplacer la propriété SQL par **Effacer SQL**.

Correspondance par code alternatif

Vous pouvez définir une commande SQL afin d'établir une correspondance à un code alternatif de votre choix. Par exemple, vous pouvez avoir des codes promotionnels ou des codes produits distincts des codes d'offre ou de traitement.

Vous devez définir ce code de remplacement dans la table `UACI_TrackingType` dans les tables de l'environnement d'exécution Interact.

Vous devez fournir SQL ou une procédure mémorisée dans la propriété `Interact > services > crossSessionResponse > OverridePerAudience > (AudienceLevel) > TrackingCodes > byAlternateCode > OverrideSQL` qui renvoie toutes les colonnes de la table XSessResponse pour ce référentiel, plus les colonnes `TreatmentCodeMatch` et `OfferIDMatch`. Vous pouvez facultativement renvoyer `offerCode` à la place de `OfferIDMatch` (sous la forme `offerCode1`, `offerCode2`, ... `offerCodeN` pour N codes d'offre de partie). Les valeurs de la colonne `TreatmentCodeMatch` et `OfferIDMatch` (ou colonnes de code d'offre) doivent correspondre à `TrackingCode` dans l'enregistrement XSessResponse.

Par exemple, le pseudo-code SQL suivant correspond à la colonne `AlternateCode` de la table XSessResponse.

```
Select m.TreatmentCode as TreatmentCodeMatch, m.OfferID as OfferIDMatch, tx.*
From MyLookup m, UACI_XSessResponse tx
Where m.customerId = tx.customerId
And m.alternateCode = tx.trackingCode
And tx.mark=1
And tx.trackingCodeType = <x>
```

où *<x>* est le code de suivi défini dans la table `UACI_TrackingType`.

Utilisation d'un utilitaire de chargement de base de données avec l'environnement d'exécution

Par défaut, l'environnement d'exécution enregistre écrit les données d'historique des contacts et de réponses à partir des données de session dans les tables de transfert. Sur un système de production très actif, cependant, la quantité de mémoire nécessaire pour mettre en cache toutes les données avant que l'exécution puisse les écrire dans les tables de transfert peut être prohibitive. Vous pouvez configurer l'environnement d'exécution afin qu'il utilise un utilitaire de chargement de base de données pour améliorer les performances.

Lorsque vous activez un utilitaire de chargement de base de données, au lieu de contenir tout l'historique des contacts et des réponses dans la mémoire avant de l'écrire dans les tables de transfert, l'exécution écrit les données dans un fichier de transfert. Vous définissez l'emplacement du répertoire contenant les fichiers de transfert avec la propriété `externalLoaderStagingDirectory`. Ce répertoire contient plusieurs sous-répertoires. Le premier sous-répertoire est le répertoire de l'instance d'exécution, qui contient les répertoires `contactHist` et `respHist`. Les répertoires `contactHist` et `respHist` contiennent des sous-répertoires ayant un nom unique au format *audienceLevelName.uniqueID.currentState*, qui contient les fichiers de transfert.

Etat en cours	Description
CACHE	Contenu du répertoire en cours d'écriture dans un fichier.
READY	Contenu du répertoire prêt à être traité.
RUN	Contenu du répertoire en cours d'écriture dans la base de données.

Etat en cours	Description
PROCESSED	Contenu du répertoire qui a été écrit dans la base de données.
ERROR	Une erreur s'est produite lors de l'écriture du contenu du répertoire dans la base de données.
ATTN	Le contenu du répertoire nécessite d'être traité. Vous devrez peut-être effectuer certaines étapes manuelles pour terminer l'écriture du contenu de ce répertoire dans la base de données.
RERUN	Contenu du répertoire prêt à être écrit dans la base de données. Vous devez renommer un répertoire à partir de `ATTN` ou `ERROR` pour exécuter l'opération`RERUN` après avoir corrigé le problème.

Vous pouvez définir le répertoire d'instance de répertoire en définissant la propriété JVM interact.runtime.instance.name dans le script de démarrage du serveur d'application. Par exemple, `-Dinteract.runtime.instance.name=instance2` à votre script de démarrage du serveur d'applications Web. S'il n'est pas défini, le nom par défaut est `DefaultInteractRuntimeInstance`.

Le répertoire `samples` contient des fichiers exemple qui vous aident à écrire vos propres fichiers de contrôle de l'utilitaire de chargement de base de données.

Activation d'un utilitaire de chargement de base de données avec l'environnement d'exécution

Vous devez définir tous les fichiers de commande ou de contrôle pour votre utilitaire de chargement de base de données avant de configurer l'environnement d'exécution afin qu'il les utilise. Ces fichiers doivent exister dans le même emplacement sur tous les serveurs d'exécution dans le même groupe de serveurs.

Interact fournit des exemples de commande et des fichiers de contrôle dans le répertoire `loaderService` dans votre installation du serveur d'exécution Interact.

1. Confirmez que l'utilisateur de l'environnement d'exécution dispose de droits d'accès de connexion à la source de données des tables d'exécution définie dans Marketing Platform.

   Le nom de la source de données dans Marketing Platform doit être `systemTablesDataSource`.
2. Définissez les propriétés de configuration `Interact > general > systemTablesDataSource > loaderProperties`.
3. Définissez la propriété `Interact > services >externalLoaderStagingDirectory`.
4. Révisez les propriétés de configuration `Interact > services > responseHist > fileCache` si nécessaire.
5. Révisez les propriétés de configuration `Interact > services > contactHist > fileCache` si nécessaire.
6. Redémarrez le serveur d'exécution.

Chapitre 4. Présentation des offres

Vous pouvez configurer Interact de nombreuses façons afin d'améliorer la manière dont il choisit les offres à présenter. Les sections suivantes expliquent en détail ces fonctions en option.

Éligibilité d'une offre

Le but de Interact est de présenter des offres éligibles. Pour l'expliquer simplement, Interact présente les offres les plus optimales parmi les offres éligibles, en fonction du visiteur, du canal, et de la situation.

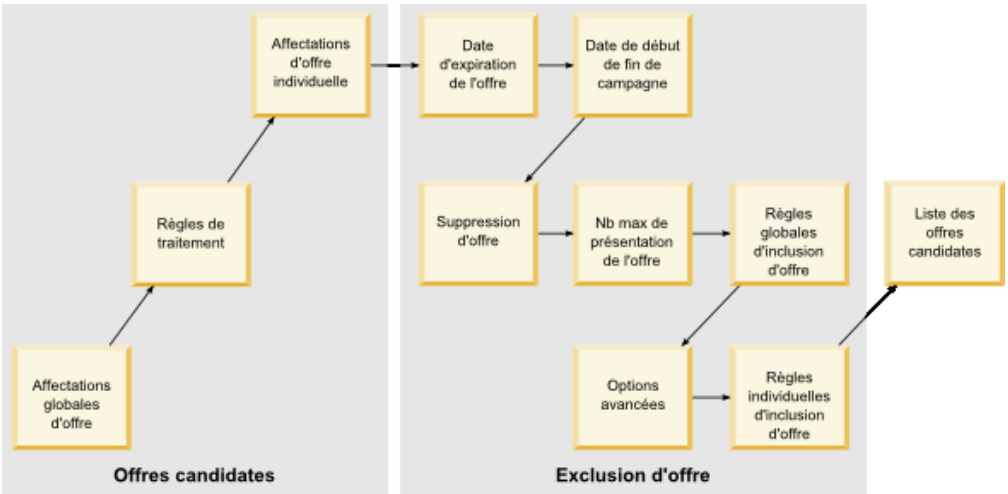
Les règles de traitement sont l'élément de départ qui permet à Interact de déterminer les offres les mieux adaptées à un client. Interact comporte plusieurs fonctions optionnelles que vous pouvez implémenter afin d'améliorer la manière dont l'environnement d'exécution détermine les offres à présenter. Aucune de ces fonctions ne garantit qu'une offre est présentée à un client. Ces fonctions influencent la probabilité qu'une offre puisse être présentée à un client. Vous pouvez utiliser le nombre de fonctions de votre choix, en fonction de la meilleure solution à retenir pour votre environnement.

Il existe trois domaines principaux dans lesquels vous pouvez influer sur l'éligibilité de l'offre : génération de la liste des offres candidates, détermination du score marketing et apprentissage.

Génération d'une liste d'offres candidates

La génération d'une liste d'offres candidates se compose deux grandes étapes. La première étape consiste à générer une liste de toutes les offres possibles concernant le client. La deuxième étape consiste à filtrer toute offre ne concernant plus le client. Il existe différents moments dans ces deux étapes auxquels vous pouvez influer la génération de la liste des offres candidates.

Ce diagramme illustre les étapes de la génération de la liste des offres candidates. Les flèches indiquent l'ordre de priorité. Par exemple, si une offre passe le filtre **Nombre maximal de présentation d'une offre**, mais ne passe pas le filtre **Règles d'inclusion des offres globales** , l'environnement d'exécution exclut l'offre.



© Copyright IBM Corp. 2001, 2012

- **Affectations d'offres globales** — Vous pouvez définir des offres globales par référentiel à l'aide de la table des offres globales.
- **Règles de traitement** — Méthode de base permettant de définir des offres par segment et par point d'interaction, à l'aide de l'onglet Stratégie d'interaction.
- **Affectations d'offres individuelles** — Vous pouvez définir des affectations d'offre spécifiques par client à l'aide de la table de substitution de score.
- **Date d'expiration de l'offre** — Lorsque vous créez une offre dans Campaign, vous pouvez définir une date d'expiration. Si la date d'expiration d'une offre est arrivée à échéance, l'environnement d'exécution exclut l'offre.
- **Date de début et de fin de campagne** - Lorsque vous créez une campagne dans Campaign, vous pouvez définir une date de début et de fin de la campagne. Si la date de début de la campagne n'est pas encore intervenue ou si la date de fin est arrivée à échéance, l'environnement d'exécution exclut l'offre.
- **Suppression d'offre** - Vous pouvez définir une suppression d'offre pour certains membres d'un référentiel à l'aide de la table de suppression des offres.
- **Nombre maximal de présentation d'une offre** — Lorsque vous définissez un canal interactif, vous définissez le nombre maximal de présentation d'une offre à un client par session. Si le nombre de présentations de l'offre a déjà été atteint, l'environnement d'exécution exclut l'offre.
- **Règles d'inclusion d'offres globales** — Vous pouvez définir une expression booléenne pour filtre les offres par référentiel à l'aide de la table des offres globales. Si le résultat est false, l'environnement d'exécution exclut l'offre.
- **Options avancées** -Vous pouvez utiliser l'option avancée**Considérer cette règle comme éligible si l'expression suivante est vraie...** dans une règle de traitement pour filtrer des offres au niveau d'un segment. Si le résultat est false, l'environnement d'exécution exclut l'offre.
- **Règles d'inclusion d'offre individuelles** — Vous pouvez définir une expression booléenne pour filtrer les offres au niveau d'un client à l'aide de la table de substitution de score. Si le résultat est false, l'environnement d'exécution exclut l'offre.

Calcul du score marketing

Il existe de nombreuses manières d'influencer ou de remplacer le score marketing (à l'aide d'un calcul). Le diagramme suivant illustre les différentes étapes auxquelles vous pouvez modifier ou substituer le score marketing. Les flèches indiquent l'ordre de priorité. Par exemple, si vous définissez une expression pour déterminer le score marketing dans les options avancées d'une règle de traitement et pour définir une expression dans la table de substitution de score, l'expression dans la table de substitution de score est prioritaire.

- **Score global** — Vous pouvez définir un score par référentiel à l'aide de la table des offres globales.
- **Score du marketeur** — Vous pouvez définir un score par segment à l'aide du curseur dans une règle de traitement.
- **Score de substitution de score** — Vous pouvez définir un score par client à l'aide de la table de substitution de score.
- **Règles d'inclusion des offres globales** — Vous pouvez définir une expression qui calcule un score par référentiel à l'aide de la table des offres globales.
- **Options avancées** — Vous pouvez définir une expression qui calcule un score par segment à l'aide de l'option avancée **Utiliser l'expression suivante comme score marketing...** dans une règle de traitement.
- **Règles d'inclusion des offres par substitution de score** — Vous pouvez définir une expression qui calcule un score par client à l'aide de la table de substitution de score.

Influencer l'apprentissage

Si vous utilisez le module d'apprentissage intégré Interact, vous pouvez influencer le résultat d'apprentissage en plus des configurations d'apprentissage standard telles que la liste des attributs d'apprentissage ou le niveau de fiabilité. Vous pouvez substituer les composants de l'algorithme d'apprentissage tout en utilisant les composants restants.

Vous pouvez substituer l'apprentissage à l'aide des colonnes `LikelihoodScore` et `AdjExploreScore` des offres par défaut et des tables de substitution de score. Vous pouvez ajouter ces colonnes aux offres par défaut et aux tables de substitution de score à l'aide de la fonction de script `aci_scoringfeature`. Pour utiliser correctement ces substitutions, vous avez besoin d'une compréhension approfondie de l'apprentissage intégré de Interact.

Ce module d'apprentissage prend la liste des offres candidates et le score de marketing par offre candidate et les utilise dans les calculs finaux. La liste des offres est utilisée avec les attributs d'apprentissage afin de calculer la probabilité (probabilité d'acceptation) que le client accepte l'offre. A l'aide de ces probabilités et le nombre historisé des présentations, à équilibrer entre exploration et exploitation, l'algorithme d'apprentissage détermine la pondération de l'offre. Enfin, l'apprentissage intégré utilise la pondération de l'offre, la multiplie par le score marketing final et renvoie un score final. Les offres sont triés en fonction de ce score final.

A propos de la suppression des offres

Il existe plusieurs façons de supprimer une offre pour l'environnement d'exécution :

- L'élément **Nbre maximal d'affichages d'une offre au cours d'une même visite** d'un canal interactif.

  Vous définissez le **Nbre maximal d'affichages d'une offre au cours d'une même visite** d'un lorsque vous créez ou éditez un canal interactif.
- L'utilisation d'une table de suppression d'offre.

  Vous créez une table de suppression d'offre dans votre base de données de profil.
- Offre dont la date d'expiration est dépassée.
- Offres provenant de campagnes arrivées à expiration.
- Offre exclues car elles ne transmettent pas une règle d'inclusion d'offre (option avancée de règle de traitement).
- Offres déjà explicitement acceptées ou refusées dans une session Interact. Si un client accepte ou refuse explicitement une offre, cette offre est supprimée pendant la durée de la session.

Activation de la table de suppression des offres

Vous pouvez configurer Interact afin qu'il fasse référence à une liste d'offres supprimées.

1. Créez la table `offerSuppressionTable`, une nouvelle table pour chaque référentiel contenant l'ID de référentiel et l'ID de l'offre.
2. Définissez la propriété `enableOfferSuppressionLookup` sur **true**.
3. Définissez la propriété `offerSuppressionTable` sur le nom de la table de suppression des offres pour le référentiel adéquat.

Table de suppression des offres

La table de suppression des offres vous permet de supprimer une offre pour un ID de référentiel spécifique. Par exemple, si votre référentiel est Client, vous pouvez supprimer une offre pour le client Eric Martin. Une version de cette table pour au moins un référentiel doit exister dans votre base de données de profil de production. Vous pouvez créer un exemple de table de suppression d'offres, `UACI_Blacklist`, en exécutant le script SQL `aci_usertab` sur votre base de données de profil. Le script SQL `aci_usertab` se trouve dans le répertoire `ddl` du répertoire d'installation de l'environnement d'exécution.

Vous devez définir les zones `AudienceID` et `OfferCode1` pour chaque ligne. Vous pouvez ajouter des colonnes supplémentaires si votre ID de référentiel ou votre code d'offre comprend plusieurs colonnes. Ces colonnes doivent correspondre aux noms de colonnes définis dans Campaign. Par exemple, si vous définissez le référentiel `Client` par les zones `HHold_ID` et `MemberNum`, vous devez ajouter `HHold_ID` et `MemberNum` à la table de suppression des offres.

Nom	Description
AudienceID	(Obligatoire) Le nom de cette colonne doit correspondre au nom de la colonne définissant l'ID de référentiel Campaign. Si votre ID de référentiel comprend plusieurs colonnes, vous pouvez les ajouter à cette table. Chaque ligne doit contenir l'ID de référentiel auquel vous affectez l'offre par défaut, par exemple, `client1`.
OfferCode1	(Obligatoire) Code de l'offre de l'offre que vous substituez. Si vos codes d'offre comprennent plusieurs zones, vous pouvez ajouter des colonnes supplémentaires, par exemple `OfferCode2` et ainsi de suite.

Offre globale et affectations individuelles

Vous pouvez configurer l'environnement d'exécution afin qu'il affecte des offres spécifiques en dehors des règles de traitement configurées dans l'onglet Stratégie d'interaction. Vous pouvez définir des offres globales pour n'importe quel membre d'un membre d'un référentiel et des affectations individuelles pour certains membres d'un référentiel. Par exemple, vous pouvez définir une offre globale pour tous les foyers à afficher lorsqu'aucun autre critère n'est disponible, puis créer une affectation d'offre individuelle au foyer de la famille Martin en particulier.

Vous pouvez imposer des contraintes aux offres globales et aux affectations individuelles par zone, de la cellule, et règles d'inclusion des offres. Les offres globales et les affectations individuelles sont configurées par l'ajout de données à des tables spécifiques dans votre base de données de profil de production

Pour que les offres globales et les affectations individuelles fonctionnent correctement, tous les codes de la cellule et toutes les offres référencées doivent exister dans le déploiement. Pour vous assurer que les données requises sont disponibles, vous devez configurer les codes de cellule par défaut et la table`UACI_ICBatchOffers`.

Définition des codes de cible par défaut

Si vous utilisez les offres par défaut ou les tables de substitution de score pour les affectations des offres globales ou individuelles, vous devez définir des codes de cible par défaut en définissant la propriété `DefaultCellCode` pour chaque référentiel et le type de table dans la catégorie `IndividualTreatment`.

`DefaultCellCode` est utilisé si vous n'avez pas défini un code de cible dans une ligne particulière dans les offres par défaut ou les tables de substitution des scores. La génération de rapports utilise le code de cible par défaut.

`DefaultCellCode` doit correspondre au format du code de cible défini dans Campaign. Ce code de cible est utilisé pour toutes les affectations d'offre figurant dans les rapports. Si vous définissez des codes de cible par défaut uniques, vous pouvez identifier facilement les offres affectées par les offres par défaut ou par les tables de substitution des scores.

Définition de la table UACI_ICBatchOffers

Si vous utilisez les offres par défaut ou les tables de substitution de score, vous devez vous assurer que tous les codes d'offre existent dans le déploiement. Si vous savez que toutes les offres que vous utilisez dans les offres par défaut ou dans les tables de substitution de score sont utilisées dans vos règles de traitement, les

offres existent dans le déploiement. Cependant, toute offre non utilisée dans une règle de traitement doit être définie dans la table UACI_ICBatchOffers.

La table UACI_ICBatchOffers existe dans les tables système Campaign.

Vous devez renseigner la table UACI_ICBatchOffers avec les codes d'offre utilisés dans l'offre par défaut ou dans les tables de substitution de score. La table a le format suivant.

Nom de la colonne	Type	Description
ICName	varchar(64)	Nom du canal interactif auquel l'offre est associée. Si vous utilisez la même offre avec deux canaux interactifs différents, vous devez fournir une ligne pour chaque canal interactif.
OfferCode1	varchar(64)	Première partie du code de l'offre.
OfferCode2	varchar(64)	Deuxième partie du code de l'offre.
OfferCode3	varchar(64)	Troisième partie du code de l'offre, le cas échéant.
OfferCode4	varchar(64)	Quatrième partie du code de l'offre, le cas échéant.
OfferCode5	varchar(64)	Cinquième partie du code de l'offre, le cas échéant.

A propos de la table des offres globales

La table des offres globales vous permet de définir des traitements pour le référentiel. Par exemple, vous pouvez définir une offre globale pour chaque membre du Foyer dans le référentiel.

Vous pouvez définir les paramètres globaux pour les éléments suivants de la proposition d'offre Interact.

- Affectation d'offre globale
- Score global du vendeur, désigné par un nombre ou une expression
- Expression booléenne de filtrage des offres
- Probabilité d'apprentissage et pondération, si vous utilisez l'apprentissage intégré Interact
- Substitution d'apprentissage globale

Activation de la table des offres globales

Vous pouvez configurer l'environnement d'exécution afin qu'il affecte des offres globales à un référentiel, en plus de ce qui est déjà défini dans les règles de traitement.

1. Créez une table appelée UACI_DefaultOffers dans votre base de données de profil.

   Vous pouvez créer la table UACI_DefaultOffers avec les colonnes correctes à l'aide du fichier DDL aci_usrtab.
2. Définissez la propriété enableDefaultOfferLookup sur **true**.

Table des offres globales

La table des offres globales doit exister dans votre profil de base de données. Vous pouvez créer la table des offres globales, UACI_DefaultOffers en exécutant le script

SQL aci_usertab sur votre base de données de profil. Le script SQL aci_usertab se trouve dans le répertoire ddl dans votre répertoire d'installation de l'environnement d'exécution.

Vous devez définir les zones AudienceLevel et OfferCode1 pour chaque ligne. Les autres zones sont facultatives et permettent d'imposer des contraintes supplémentaires aux affectations de votre offre ou d'influencer l'apprentissage intégré au référentiel.

Pour obtenir de meilleures performances, vous devez créer un index sur cette table dans la colonne du référentiel.

Nom	Type	Description
AudienceLevel	varchar(64)	(Obligatoire) Nom du référentiel que vous affectez à l'offre par défaut, par exemple, client ou foyer. Ce nom doit correspondre au référentiel, comme défini dans Campaign.
OfferCode1	varchar(64)	(Obligatoire) Code de l'offre de l'offre par défaut. Si vos codes d'offre comprennent plusieurs zones, vous pouvez ajouter des colonnes supplémentaires, par exemple CodeOffre2 et ainsi de suite. Si vous ajoutez cette offre pour proposer une affectation d'offre globale, vous devez ajouter cette offre à la table UACI_ICBatchOffers.
Score	float	Nombre définissant le score marketing de cette affectation d'offre.
OverrideTypeID	int	Si la valeur est définie sur 1, et si l'offre n'existe pas dans la liste des offres candidates, ajoutez cette offre à la liste et utilisez les données de scores de l'offre le cas échéant. En général, utilisez 1 pour fournir des affectations d'offre globale. Si la valeur est définie sur 0, *null*, ou sur un nombre différent de 1, utilisez toutes les données de l'offre uniquement si l'offre existe dans la liste des offres candidates. Dans la plupart des cas, une règle de traitement ou une affectation individuelle remplace ce paramètre.

Nom	Type	Description
Prédicat	varchar(4000)	Vous pouvez entrer des expressions dans cette colonne, comme c'est le cas pour les options avancées des règles de traitement. Vous pouvez utiliser les mêmes variables et les mêmes macros disponibles lorsque vous écrivez des options avancées pour les règles de traitement. Le comportement de cette colonne dépend de la valeur de la colonne `EnableStateID`. • Si `EnableStateID` est `2`, cette colonne fonctionne de la même manière que l'option **Considérer cette règle comme éligible si l'expression suivante est vraie...** dans les options avancées des règles de traitement permettant de contraindre cette affectation d'offre. Cette colonne doit contenir une expression booléenne, et donner la valeur true après résolution pour inclure cette offre. Si vous avez défini par erreur une expression qui donne un nombre après résolution, tout nombre différent de zéro est considéré comme ayant la valeur true et zéro est considéré comme ayant la valeur false. • Si `EnableStateID` a la valeur `3`, cette colonne fonctionne de la même manière que l'option **Utiliser l'expression suivante comme score marketing...** dans les options avancées des règles de traitement permettant de contraindre cette offre. Cette colonne doit contenir une expression correspondant à un nombre une fois résolue. • Si `EnableStateID` a la valeur `1`, Interact ignore toute valeur dans cette colonne.
FinalScore	float	Nombre qui remplace le score final utilisé pour ordonner la liste définitive des offres renvoyées. Cette colonne est utilisée si vous avez activé le module d'apprentissage intégré. Vous pouvez implémenter votre propre apprentissage pour utiliser cette colonne.
CellCode	varchar(64)	Code de cible d'un segment interactif auquel vous souhaitez affecter cette offre par défaut. Si vos codes de cible comportent plusieurs zones, vous pouvez ajouter les colonnes supplémentaires. Vous devez fournir un code de cible si `OverrideTypeID` est `0` ou null. Si vous n'incluez pas de code de cible, l'environnement d'exécution ignore cette ligne de données. Si `OverrideTypeID` est `1`, vous n'avez pas besoin de fournir de code de cible dans cette colonne. Si vous n'indiquez pas de code de cible, l'environnement d'exécution utilise le code de cible défini dans la propriété `DefaultCellCode` pour ce référentiel et cette table à des fins de génération de rapports.
Zone	varchar(64)	Nom de la zone à laquelle vous souhaitez appliquer cette affectation d'offre. Si la valeur est NULL, cela s'applique à toutes les zones.

Nom	Type	Description
EnableStateID	int	La valeur de cette colonne définit le comportement de la colonne `Prédicat`. • **1** - N'utilisez pas ma colonne `Prédicat`. • **2** - Utilisez `Prédicat` comme valeur booléenne pour filtrer l'offre. Cela suit les mêmes règles que l'option avancée **Considérer cette règle comme éligible si l'expression suivante est vraie...** dans une règle de traitement. • **3** - Utilisez `Prédicat` pour définir le score du vendeur. Cela suit les mêmes règles que l'option avancée **Utiliser l'expression suivante comme score marketing...** dans une règle de traitement. Toute ligne dans laquelle cette colonne a la valeur Null ou tout e autre valeur que 2 ou 3 ignore la colonne `Prédicat`.
LikelihoodScore	float	Cette colonne est utilisée uniquement pour influencer l'apprentissage intégré. Vous pouvez ajouter cette colonne avec le DDL `aci_scoringfeature`.
AdjExploreScore	float	Cette colonne est utilisée uniquement pour influencer l'apprentissage intégré. Vous pouvez ajouter cette colonne avec le DDL `aci_scoringfeature`.

A propos de la table de substitution de score

La table de substitution de score vous permet de définir des traitements pour un ID de référentiel ou pour un individu. Par exemple, si votre référentiel est Visiteur, vous pouvez créer des substitutions pour certains visiteurs.

Vous pouvez définir des substitutions pour les éléments suivants de la proposition d'offre Interact.

- Affectation d'offre individuelle
- Score individuel du vendeur, désigné par un nombre ou une expression
- Expression booléenne de filtrage des offres
- Probabilité d'apprentissage et pondération, si vous utilisez l'apprentissage intégré
- Substitution d'apprentissage individuelle

Activation de la table de substitution de score

Vous pouvez configurer Interact afin qu'il utilise un score généré par une application de modélisation au lieu du score marketing.

1. Créez une table de substitution de score pour chaque référentiel pour lequel vous souhaitez fournir des substitutions.

   Vous pouvez créer un exemple de table de substitution de score avec les colonnes correctes à l'aide du fichier DDL `aci_usrtab`.
2. Définissez la propriété `enableScoreOverrideLookup` sur **true**.
3. Définissez la propriété `scoreOverrideTable` sur le nom de la table de substitution de score pour chaque référentiel pour lequel vous souhaitez fournir des substitutions.

Vous n'avez pas besoin de fournir une table de substitution de score pour chaque référentiel.

Table de substitution de score

La table de substitution de score doit exister dans votre base de données de profil de production. Vous pouvez créer un exemple de table de substitution de score, `UACI_ScoreOverride` en exécutant le script SQL `aci_usertab` sur votre base de données de profil. Le script SQL `aci_usertab` se trouve dans le répertoire `ddl` du répertoire d'installation de l'environnement d'exécution.

Vous devez définir les zones*AudienceID*, `OfferCode1` et `Score` pour chaque ligne. Les valeurs des autres zones sont facultatives et permettent d'imposer des contraintes aux affectations d'offres individuelles ou de fournir des informations de substitution de score à l'apprentissage intégré.

Nom	Type	Description
AudienceID	varchar(64)	(Obligatoire) Le nom de cette colonne doit correspondre au nom de la colonne définissant l'ID de référentiel Campaign. L'exemple de table créé par le fichier ddl `aci_usertab` crée cette colonne comme la colonne `CustomerID`. Si votre ID de référentiel comprend plusieurs colonnes, vous pouvez les ajouter à cette table. Chaque ligne doit contenir l'ID de référentiel auquel vous affectez l'offre individuelle, par exemple, `client1`. Pour obtenir de meilleures performances, créez un index sur cette colonne.
OfferCode1	varchar(64)	(Obligatoire) Code de l'offre. Si vos codes d'offre comprennent plusieurs zones, vous pouvez ajouter des colonnes supplémentaires, par exemple `CodeOffre2` et ainsi de suite. Si vous ajoutez cette offre pour proposer une affectation d'offre individuelle, vous devez ajouter cette offre à la table `UACI_ICBatchOffers`.
Score	float	Nombre définissant le score marketing de cette affectation d'offre.
OverrideTypeID	int	Si la valeur est définie sur `0` ou *null*(ou sur un nombre différent de 1), utilisez toutes les données de l'offre uniquement si l'offre existe dans la liste des offres candidates. En général, utilisez `0` pour indiquer les substitutions de score. Vous devez fournir un code de cible. Si la valeur est définie sur `1`, et si l'offre n'existe pas dans la liste des offres candidates, ajoutez cette offre à la liste et utilisez les données de score de l'offre. En général, utilisez `1` pour fournir des affectations d'offres individuelles.

Nom	Type	Description
Prédicat	varchar(4000)	Vous pouvez entrer des expressions dans cette colonne, comme c'est le cas pour les options avancées des règles de traitement. Vous pouvez utiliser les mêmes variables et les mêmes macros disponibles lorsque vous écrivez des options avancées pour les règles de traitement. Le comportement de cette colonne dépend de la valeur de la colonne `EnableStateID`. • Si `EnableStateID` est 2, cette colonne fonctionne de la même manière que l'option **Considérer cette règle comme éligible si l'expression suivante est vraie...** dans les options avancées des règles de traitement permettant de contraindre cette affectation d'offre. Cette colonne doit contenir une expression booléenne, et donner la valeur true après résolution pour inclure cette offre. Si vous avez défini par erreur une expression qui donne un nombre après résolution, tout nombre différent de zéro est considéré comme ayant la valeur true et zéro est considéré comme ayant la valeur false. • Si `EnableStateID` a la valeur 3, cette colonne fonctionne de la même manière que l'option **Utiliser l'expression suivante comme score marketing...** dans les options avancées des règles de traitement permettant de contraindre cette offre. Cette colonne doit contenir une expression correspondant à un nombre une fois résolue. • Si `EnableStateID` a la valeur 1, Interact ignore toute valeur dans cette colonne.
FinalScore	float	Nombre qui remplace le score final utilisé pour ordonner la liste définitive des offres renvoyées. Cette colonne est utilisée si vous avez activé le module d'apprentissage intégré. Vous pouvez implémenter votre propre apprentissage pour utiliser cette colonne.
CellCode	varchar(64)	Code de cible d'un segment interactif auquel vous souhaitez affecter cette offre. Si vos codes de cible comportent plusieurs zones, vous pouvez ajouter les colonnes supplémentaires. Vous devez fournir un code de cible si `OverrideTypeID` est 0 ou null. Si vous n'incluez pas de code de cible, l'environnement d'exécution ignore cette ligne de données. Si `OverrideTypeID` est 1, vous n'avez pas besoin de fournir de code de cible dans cette colonne. Si vous n'indiquez pas de code de cible, l'environnement d'exécution utilise le code de cible, défini dans la propriété `DefaultCellCode` pour ce référentiel et cette table à des fins de génération de rapports.
Zone	varchar(64)	Nom de la zone à laquelle vous souhaitez appliquer cette affectation d'offre. Si la valeur est NULL, cela s'applique à toutes les zones.

Nom	Type	Description
EnableStateID	int	La valeur de cette colonne définit le comportement de la colonne `Prédicat`. • **1** - N'utilisez pas la colonne `Prédicat`. • **2** - Utilisez `Prédicat` comme valeur booléenne pour filtrer l'offre. Cela suit les mêmes règles que l'option avancée **Considérer cette règle comme éligible si l'expression suivante est vraie...** dans une règle de traitement. • **3** - Utilisez `Prédicat` pour définir le score du vendeur. Cela suit les mêmes règles que l'option avancée **Utiliser l'expression suivante comme score marketing...** dans une règle de traitement. Toute ligne dans laquelle cette colonne a la valeur Null ou tout e autre valeur que 2 ou 3 ignore la colonne `Prédicat`.
LikelihoodScore	float	Cette colonne est utilisée uniquement pour influencer l'apprentissage intégré. Vous pouvez ajouter cette colonne avec le DDL `aci_scoringfeature`.
AdjExploreScore	float	Cette colonne est utilisée uniquement pour influencer l'apprentissage intégré. Vous pouvez ajouter cette colonne avec le DDL `aci_scoringfeature`.

Présentation de l'apprentissage intégré Interact

Même si vous faites tout votre possible pour proposer les offres adaptées aux segments appropriés, vous pouvez toujours apprendre des sélections réellement effectuées par vos visiteurs. Le comportement réel de vos visiteurs doit influencer votre stratégie. Vous pouvez exécuter des outils de modélisation sur l'historique des réponses pour obtenir un score que vous pouvez ensuite inclure dans vos diagrammes interactifs. Toutefois, ces données ne sont pas des données en temps réel.

Interact fournit deux options qui vous permettent d'apprendre en temps réel à partir des actions de votre visiteur :

- Module d'apprentissage intégré - L'environnement d'exécution dispose d'un module d'apprentissage bayesien naïf. Ce module surveille les attributs du client de votre choix et utilise ces données pour sélectionner les offres à présenter.
- API d'apprentissage — L'environnement d'exécution comporte aussi une API d'apprentissage qui vous permet d'écrire votre propre module d'apprentissage.

Vous n'êtes pas obligé d'utiliser l'apprentissage. Il est désactivé par défaut.

Comprendre l'apprentissage dans Interact

Le module d'apprentissage Interact surveille le réponses des visiteurs aux offres, ainsi que les attributs des visiteurs. Le module d'apprentissage a deux modes généraux :

- **Exploration** — le module d'apprentissage présente les offres afin afin de collecter suffisamment de données de réponse pour optimiser l'estimation utilisée plus tard au cours de l'exploitation. Les offres présentées pendant l'exploration ne correspondent pas nécessairement au choix optimal.

- **Exploitation** — Lorsque suffisamment de données ont été collectées par la phase d'exploration, le module d'apprentissage utilise les probabilités pour vous aider à sélectionner les offres à proposer.

Le module d'apprentissage alterne entre l'exploration et l'exploitation en fonction de deux propriétés : un niveau de confiance que vous configurez avec la propriété `confidenceLevel` et une probabilité de présentation d'une offre aléatoire que vous configurez avec la propriété `percentRandomSelection`.

Vous pouvez définir le `confidenceLevel` sur un pourcentage qui représente le degré de certitude (ou de confiance) que doit avoir le module d'apprentissage avant que ses résultats pour une offre soient utilisés dans l'arbitrage. Au départ, lorsque le module d'apprentissage ne dispose d'aucune donnée, il s'appuie donc uniquement sur le score marketing. Lorsque chaque offre a été présentée autant de fois que cela a été défini dans `minPresentCountThreshold`, le module d'apprentissage passe en mode exploration. En l'absence d'une quantité importante de données, le module d'apprentissage n'est pas sûr que les pourcentages qu'il calcule sont corrects. Il reste alors par conséquent en mode d'exploration.

Le module d'apprentissage attribue des pondérations à chaque offre. Pur calculer les pondérations, il applique une formule qui prend comme base de départ le niveau de confiance configuré ainsi que les données historisées d'acceptation et les données de la session en cours. La formule établit une équilibre inhérent entre exploration et exploitation et renvoie la pondération adéquate.

Pour éviter que le système favorise trop l'offre optimale dès les premières phases, Interact présente une offre aléatoire en appliquant la propriété `percentRandomSelection` qui correspond à un pourcentage de temps. Cela force le module d'apprentissage à recommander d'autres offres que les plus certaines de réussir, afin de déterminer si d'autres offres réussissaient mieux si elles avaient une plus grande exposition. Par exemple, si vous configurez `percentRandomSelection` sur 5, cela signifie que pendant 5% du temps, le module d'apprentissage présente une offre aléatoire et ajoute les données de réponse à ses calculs.

Le module d'apprentissage détermine les offres présentées de la manière suivante.

1. Il calcule la probabilité qu'un visiteur sélectionne une offre.
2. Il calcule la pondération de l'offre en appliquant la probabilité de l'étape 1 et détermine s'il doit se placer en mode d'exploration ou d'exploitation.
3. Il calcule un score final pour chaque offre à l'aide du score marketing et de la pondération de l'offre à partir de l'étape 2.
4. Il trie les offres en fonction des scores déterminés à l'étape 3 et renvoie le nombre d'offres obtenant les meilleurs résultats.

Par exemple, le module d'apprentissage détermine qu'un visiteur a 30 % de chances d'accepter l'offre A et 70 % de chances d'accepter l'offre B et qu'il doit donc utiliser cette information. Selon la règle de traitement, le score marketing de l'offre A est de 75 et de 55 pour l'offre B. Toutefois, les calculs de l'étape 3 établissent que le score final de l'offre B est supérieur à celui de l'offre A. L'environnement d'exécution recommande donc l'offre B.

L'apprentissage est également basé sur la propriété `recencyWeightingFactor` et la propriété `recencyWeightingPeriod`. Ces propriétés vous permettent d'attribuer une pondération plus importante aux données les plus récentes. `recencyWeightingFactor` est le pourcentage de pondération des données récentes. `recencyWeightingPeriod` correspond à la durée considérée comme récente. Par

exemple, vous configurez `recencyWeightingFactor` sur 0,30 et `recencyWeightingPeriod` sur 24. Cela signifie que les 24 heures de données précédentes représentent 30% de toutes les données prises en compte. Si vous avez l'équivalent d'une semaine de données, toutes les données servant à établir la moyenne sur les six premiers jours représentent 70% des données, tandis que le dernier jour correspond à 30% des données.

Chaque session écrit les données suivantes dans une table intermédiaire d'apprentissage :

- Contact de l'offre
- Acceptation de l'offre
- Attributs d'apprentissage

A un intervalle configurable, un agrégateur lit ces données dans la table intermédiaire, les compile et les écrit dans une table. Le module d'apprentissage lit ces données agrégées et les utilise dans ses calculs.

Activation du module d'apprentissage

Tous les serveurs d'exécution ont un module d'apprentissage intégré. Par défaut, ce module d'apprentissage est désactivé. Vous pouvez l'activer en modifiant une propriété de configuration.

Dans Marketing Platform, pour l'environnement d'exécution, éditez les propriétés de configuration suivantes dans la catégorie `Interact > offerserving`.

Propriété de configuration	Paramètre
`optimizationType`	**BuiltInLearning**

Attributs d'apprentissage

Le module d'apprentissage s'enrichit et évolue à l'aide des attributs visiteur et des données d'acceptation des offres. Vous pouvez sélectionner les attributs visiteur à surveiller. Ces attributs visiteur peuvent être toute donnée contenue dans un profil client, par exemple un attribut stocké dans une table de dimension référencée dans un graphique de flux interactif, ou un paramètre d'événement collecté en temps réel.

Vous pouvez configurer le nombre d'attributs de votre choix à surveiller, mais IBM vous conseille de ne pas configurer plus de dix attributs d'apprentissage entre les attributs d'apprentissage statique et dynamique, et de suivre ces instructions.

- Sélectionnez des attributs indépendants.

  Ne sélectionnez pas d'attributs similaires. Par exemple, si vous créez un attribut appelé ValeurÉlevée et si cet attribut est défini à partir d'un calcul basé sur le salaire, ne sélectionnez pas les deux attributs ValeurÉlevée et Salaire. Les as similaires ne sont pas utiles pour l'algorithme d'apprentissage.
- Sélectionnez des attributs ayant des valeurs discrètes.

  Si un attribut comporte des plages de valeur, vous devez sélectionner une valeur exacte. Par exemple, si vous souhaitez utiliser le salaire en tant qu'attribut, vous devez donner à chaque plage de salaire une valeur spécifique. La plage de 20 000 à 30 000 correspondrait ainsi à la valeur A, et la plage de 30 001 à 40 000 à la valeur B, etc.
- Limitez le nombre d'attributs suivis afin de ne pas dégrader les performances.

Le nombre d'attributs que vous pouvez suivre dépend de vos exigences de performances et de votre installation Interact. Si vous le pouvez, utilisez un autre outil de modélisation (par exemple PredictiveInsight) pour déterminer les dix attributs de prévision principaux. Vous pouvez configurer le module d'apprentissage afin qu'il supprime automatiquement les attributs non prévisibles, mais qui impactent aussi les performances.

Vous pouvez gérer les performances en définissant le nombre d'attributs surveillés et le nombre de valeurs par attribut à surveiller. La propriété `maxAttributeNames` définit le nombre maximum d'attributs visiteur suivis. La propriété `maxAttributeValues` définit le nombre maximum de valeurs suivies par attribut. Toutes les autres valeurs sont affectés à une catégorie définie par la valeur de la propriété `otherAttributeValue`. Cependant, le moteur d'apprentissage ne suit que les premières valeurs qu'il rencontre. Imaginons par exemple que vous suivez l'attribut visiteur correspondant à la couleur des yeux du visiteur. Vous êtes uniquement intéressé par les valeurs bleu, brun et vert, et définissez par conséquent `maxAttributeValues` sur 3. Toutefois, les trois premiers visiteurs ont les valeurs bleu, brun et noisette. Cela signifie que tous les visiteurs ayant les yeux verts se voient affecter la valeur `otherAttributeValue`.

Vous pouvez également utiliser les attributs d'apprentissage dynamique qui vous permettent de définir vos critères d'apprentissage plus précisément. Les attributs d'apprentissage dynamique vous permettent d'en savoir plus sur la combinaison de deux attributs sous la forme d'une entrée unique. Par exemple, examinez les informations de profil suivantes :

ID visiteur	Type de carte	Solde de la carte
1	Carte Gold	$1000
2	Carte Gold	$9000
3	Carte Bronze	$1000
4	Carte Bronze	$9000

Si vous utilisez des attributs d'apprentissage standard, vous pouvez uniquement obtenir des informations sur le type de carte et son solde individuellement. Les visiteurs 1 et 2 seront regroupés ensemble en fonction du type de carte, et les visiteurs 2 et 4 en fonction du du solde de la carte. Ces attributs ne sont pas forcément un moyen précis de prévoir le comportement d'acceptation de l'offre. Si les détenteurs de la carte Gold ont tendance à avoir des soldes plus élevés, le comportement du Visiteur peut être radicalement différent de celui du Visiteur 4, ce qui fausserait les attributs d'apprentissage standard. Toutefois, si vous utilisez des attributs d'apprentissage dynamique, des informations d'apprentissage sont collectées individuellement sur chacun de et les estimations seront alors plus précises.

Si vous utilisez des attributs d'apprentissage dynamique, et si le visiteur a deux valeurs valides pour un attribut, le module d'apprentissage sélectionne la première valeur trouvée.

Si vous définissez la propriété `enablePruning` sur `yes`, le module d'apprentissage utilise l'algorithme pour déterminer quels attributs ne sont pas prévisibles et cesse de prendre en compte ces attributs lors du calcul des pondérations. Par exemple, si vous effectuez le suivi d'un attribut représentant la couleur de cheveux, et si le module d'apprentissage détermine qu'il n'existe pas de modèle d'acceptation d'une offre basée sur la couleur de cheveux du visiteur, le module d'apprentissage cesse

de considérer l'attribut de couleur de cheveux. Les attributs sont réévalués chaque fois que le processus d'agrégation d'apprentissage s'exécute (il est défini par la propriété aggregateStatsIntervalInMinutes). Les attributs d'apprentissage dynamique sont aussi supprimés.

Définition d'un attribut d'apprentissage

Vous pouvez configurer au maximum le nombre maxAttributeNames d'attributs de visiteurs.

Dans Marketing Platform, pour l'environnement de conception, éditez les propriétés de configuration suivantes dans la catégorie Campaign > partitions > partition*n* > Interact > learning.

(learningAttributes) est un modèle permettant de créer de nouveaux attributs d'apprentissage. Vous devez entrer un nouveau nom pour chaque attribut. Vous ne pouvez pas créer deux catégories ayant le même nom

Propriété de configuration	Paramètre
attributeName	attributeName doit correspondre au nom d'une paire nom-valeur dans les données de profil. Ce nom est insensible à la casse.

Définition d'attributs d'apprentissage dynamique

Pour définir les attributs d'apprentissage dynamique, vous devez renseigner la table UACI_AttributeList dans la source de données Apprentissage.

Toutes les colonnes de cette table ont le type varchar(64).

Colonne	Description
AttributeName	Nom de l'attribut dynamique à utiliser pour l'apprentissage. Il doit s'agir d'une valeur réelle possible dans AttributeNameCol.
AttributeNameCol	Nom qualifié complet de la colonne (structure hiérarchique, à partir de la table de profil) dans laquelle se trouve AttributeName. Ce nom de colonne ne doit pas forcément être un attribut d'apprentissage standard.
AttributeValueCol	Nom qualifié complet de la colonne (structure hiérarchique, à partir de la table de profil) dans laquelle se trouve la valeur associée à AttributeName.

Par exemple, examinez la table de profil suivante et sa table de dimension associée.

Tableau 5. MaTableDeProfil

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

Tableau 6. MaTableDeDimension

KeyField	CardType	CardBalance
Key1	Carte Gold	1000
Key2	Carte Gold	9000
Key3	Carte Bronze	1000
Key4	Carte Bronze	9000

Voici un exemple de table `UACI_AttributeList` établissant une correspondance avec le type de carte et le solde.

Tableau 7. UACI_AttributeList

AttributeName	AttributeNameCol	AttributeValueCol
Carte Gold	MaTableDeProfil.MaTable-DeDimension. CardType	MaTableDeProfil.MaTable-DeDimension. CardBalance
Carte Bronze	MaTableDeProfil.MaTable-DeDimension. CardType	MaTableDeProfil.MaTable-DeDimension. CardBalance

Activation de l'apprentissage externe

Vous pouvez utiliser l'API d'apprentissage Java API pour écrire votre propre module d'apprentissage. Vous devez configurer l'environnement d'exécution afin qu'il reconnaisse votre utilitaire d'apprentissage dans Marketing Platform.

Dans Marketing Platform, pour l'environnement d'exécution, éditez les propriétés de configuration suivantes dans la catégorie `Interact > offerserving`.Les propriétés de configuration de l'API de l'optimiseur d'apprentissage existent dans la catégorie `Interact > offerserving > External Learning Config`.

Propriété de configuration	Paramètre
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Nom de classe de l'apprentissage externe
`externalLearningClassPath`	Chemin d'accès à la classe ou fichiers JAR sur le serveur d'exécution pour l'apprentissage externe. Si vous utilisez un groupe de serveurs et si tous les serveurs d'exécution référencent la même instance de Marketing Platform, une copie de la classe ou des fichiers jar doit être présente au même emplacement sur chaque serveur.

Vous devez redémarrer le serveur d'exécution Interact pour que les modifications prennent effet.

Chapitre 5. Compréhension de l'API Interact

Interact sert les offres de manière dynamique à une grande variété de points de contact. Par exemple, vous pouvez configurer l'environnement d'exécution et votre point de contact pour envoyer des messages aux employés de votre centre d'appels pour les informer des meilleurs prospects pour une vente optimisée ou croisée pour un client ayant formulé un type spécifique de demande de service. Vous pouvez également configurer l'environnement d'exécution et votre point de contact pour fournir des offres adaptées à un client (visiteur), qui est entré dans une zone particulière de votre site Web.

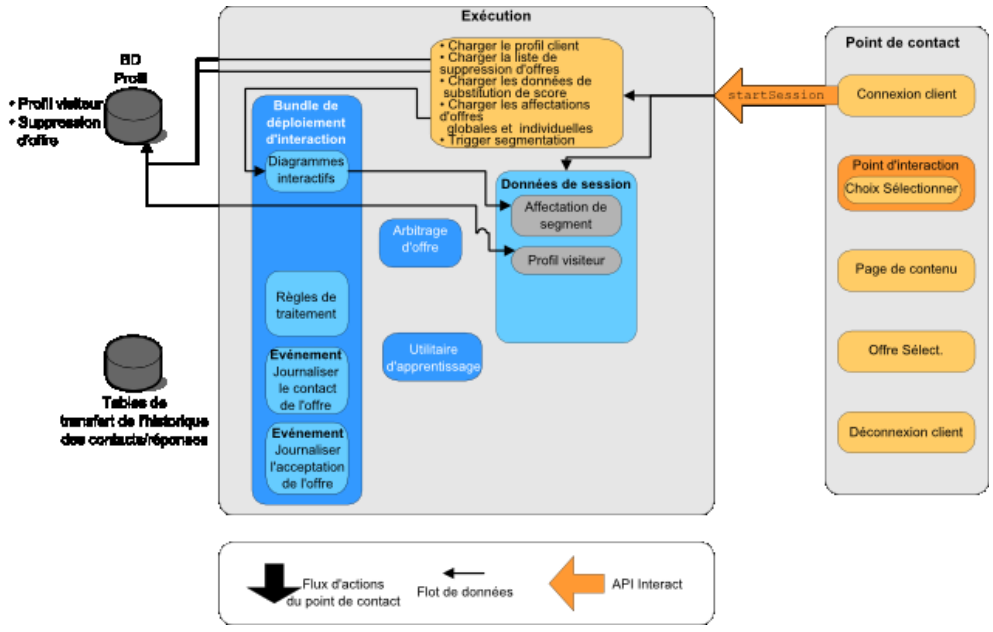
L'API Interact vous permet de configurer votre point de contact et un serveur d'exécution afin qu'ils coopèrent pour proposer les meilleures offres possibles. Avec l'API, le point de contact peut demander des informations à partir du serveur d'exécution pour affecter le visiteur à un groupe (un segment) et présente des offres basées sur ce segment. Vous pouvez également journaliser les données en vue d'une analyse ultérieure pour affiner vos stratégies de présentation d'offres.

Afin de vous fournir avec la plus grande souplesse possible en intégrant Interact à vos environnements, IBM fournit un service Web accessible à l'aide de l'API Interact.

Flux de données de l'API Interact

La figure suivante montre une implémentation simple de l'API Interact. Un visiteur se connecte à un site Web et accède à une page qui affiche des offres. Le visiteur sélectionne une offre et se déconnecte. Alors que l'interaction est simple, plusieurs événements se produisent à la fois dans le point de liaison et sur le serveur d'exécution.

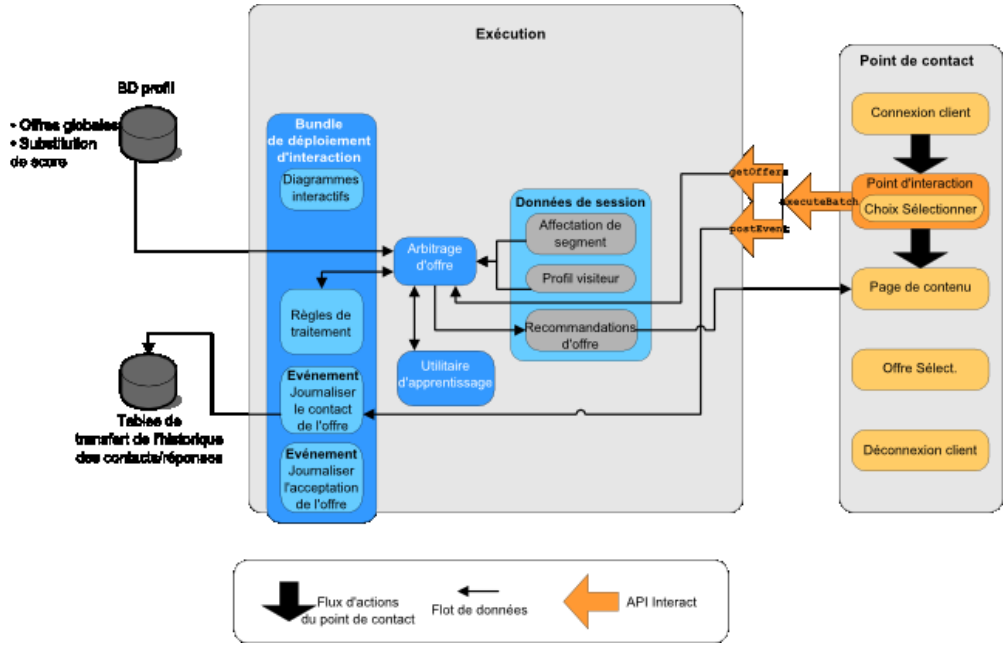
Lorsqu'un visiteur se connecte, cela déclenche un `startSession`.



© Copyright IBM Corp. 2001, 2012

Dans cet exemple, la méthode `startSession` exécute quatre opérations. Elle crée tout d'abord une nouvelle session d'exécution. Elle envoie ensuite une requête de chargement des données du profil client dans la session. Elle envoie ensuite une requête d'utilisation des données du profil et démarre un organigramme interactif pour placer le client dans des segments. Cette exécution de l'organigramme s'exécute en mode asynchrone. Pour terminer, l'exécution charge les informations de suppression d'offre et les informations de traitement globales et individuelles des offres dans la session. Les données de session sont conservées en mémoire pendant toute la durée de la session.

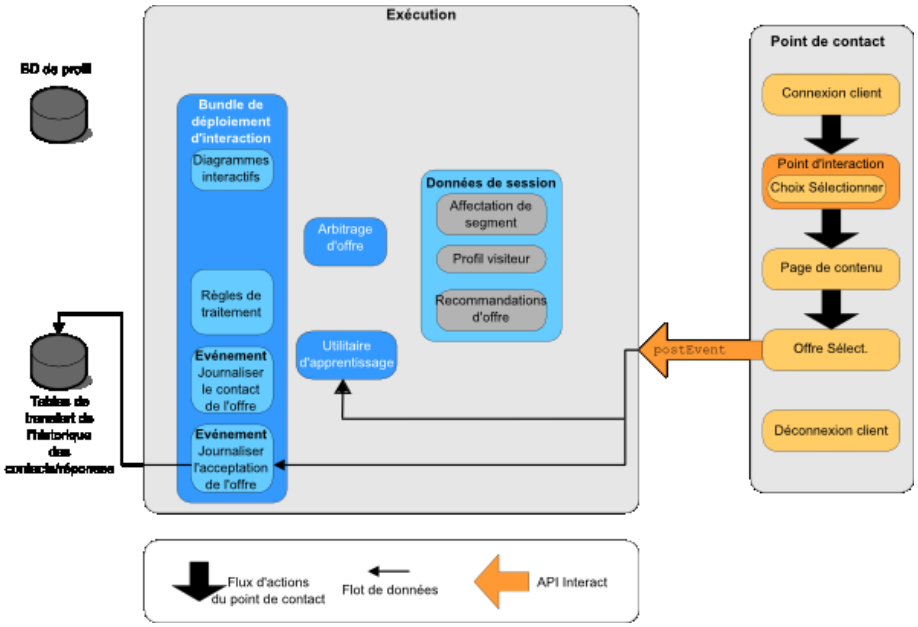
Le visiteur navigue dans le site jusqu'à ce qu'il atteigne un point d'interaction prédéfini. Dans la figure, le deuxième point d'interaction (choix Sélectionner) est un emplacement dans lequel le visiteur clique sur un lien qui lui présente un ensemble d'offres. Le gestionnaire de point de contact a configuré le lien de façon à ce qu'il déclenche une méthode `executeBatch`.



La méthode `executeBatch` permet d'appeler plus d'une méthode dans un seul appel au serveur d'exécution. Cette méthode executeBatch appelle deux autres méthodes, `getOffers` et `postEvent`. La méthode `getOffers` demande une liste d'offres. L'exécution utilise les données de segmentation, la liste de suppression des offres, les règles de traitement et le module d'apprentissage pour proposer un ensemble d'offres. L'exécution renvoie un ensemble d'offres qui s'affichent dans la page de contenu.

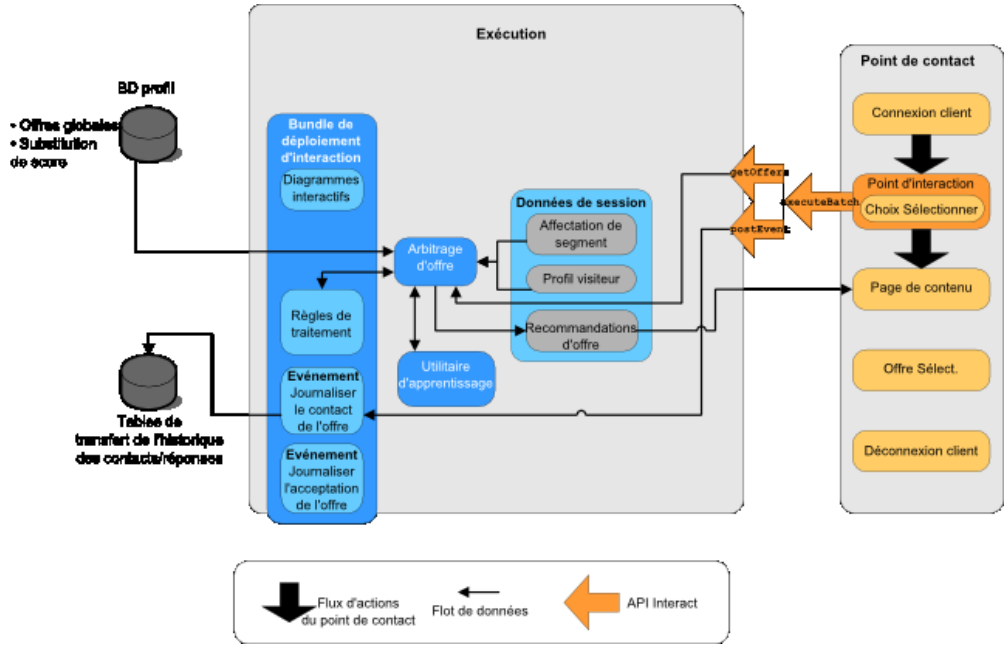
La méthode `postEvent` déclenche l'un des événements définis dans l'environnement de conception. Dans ce cas particulier, l'événement envoie une demande de journalisation des offres présentées à l'historique des contacts.

Le visiteur sélectionne l'un des offres (offre Sélect.)



Le bouton associé à la sélection de l'offre est configuré pour envoyer une autre méthode `postEvent`. Cet événement envoie une demande de journalisation de l'acceptation de l'offre dans l'historique des réponses.

Le visiteur, après avoir sélectionné l'offre, a terminé sa transaction sur le site Web et se déconnecte. La commande de déconnexion est liée à la méthode `endSession`.



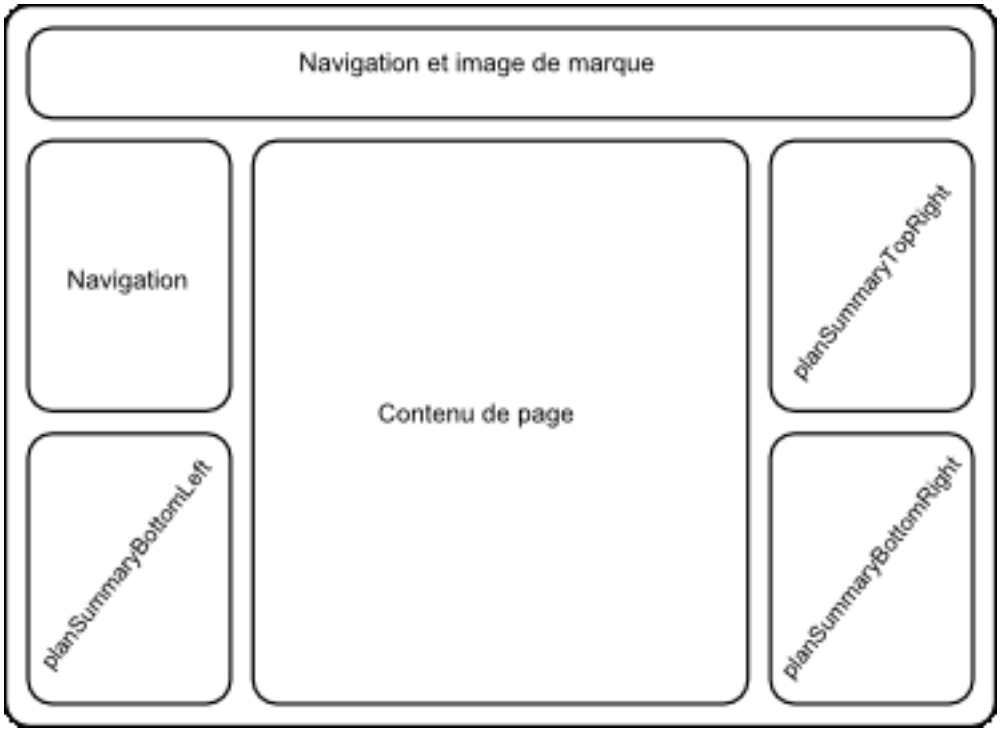
La méthode `endSession` termine la session. Si le visiteur oublie de se déconnecter, un délai d'expiration de session configurable permet de faire prendre fin à toutes les sessions. Si vous souhaitez conserver des données transmises à la session, telles que les informations incluses dans les paramètres des méthodes `startSession` ou `setAudience`, collaborez avec la personne chargée de créer les organigrammes interactifs. Cette dernière peut utiliser le processus Extraction pour écrire ces données dans une base de données avant que la fin de la session et la perte des

données. Vous pouvez alors utiliser la méthode `postEvent` pour appeler l'organigramme interactif contenant le processus Extraction.

Cet exemple est très simple. Le visiteur exécute seulement quatre actions : connexion, accès à la page affichant les offres, sélection d'une offre et déconnexion. C'est donc une interaction simple. Il illustre les principes fondamentaux qui régissent le fonctionnement de l'API entre votre point de contact et l'environnement d'exécution. Vous pouvez concevoir une intégration beaucoup plus complexe si nécessaire, à condition de respecter vos contraintes en termes d'exigences de performances.

Exemple simple de planification d'interaction

Vous créez une interaction pour le site Web d'une société de téléphonie cellulaire. Le diagramme suivant illustre la présentation de la page de récapitulatif du plan concernant un modèle de téléphone donné.



Vous définissez les éléments suivants pour répondre aux exigences de la page de récapitulatif du plan prévu pour ce téléphone.

L'une des offres doit être affichée dans une zone dédiée aux offres de mise à niveau

- La zone de la page affichant l'offre de mise à niveau doit être définie. En outre, lorsque Interact sélectionne une offre à afficher, les informations doivent être journalisées.

  Point d'interaction :: `ip_planSummaryBottomRight`

  Evénement: `evt_logOffer`

Deux offres pour les mises à niveau du téléphone

- Chaque zone de la page affichant les mises à niveau du téléphone doit être définie.

  Point d'interaction : `ip_planSummaryTopRight`

Point d'interaction :: `ip_planSummaryBottomLeft`

Pour l'analyse, vous devez journaliser les offres qui sont acceptées et celles qui sont refusées.

Evénement :: `evt_offerAccept`

Evénement : `evt_offerReject`

Vous savez également que vous devez transmettre le code de traitement d'une offre chaque fois que vous journalisez le contact d'une offre ou l'acceptation ou le refus d'une offre. Si nécessaire, vous créez une paire valeur-nom `NameValuePair` contenant le code de traitement, comme dans l'exemple suivant.

```
NameValuePair evtParam_TreatmentCode = new NameValuePairImpl();
evtParam_TreatmentCode.setName("UACIOfferTrackingCode");
evtParam_TreatmentCode.setValueAsString(offer.getTreatmentCode());
evtParam_TreatmentCode.setValueDataType(NameValuePair.DATA_TYPE_STRING);
```

Vous pouvez maintenant demander à l'utilisateur de l'environnement de conception de créer les points d'interaction et les événements pendant que vous commencez à coder l'intégration avec votre point de contact.

Pour chaque point d'interaction qui doit afficher une offre, vous devez d'abord obtenir une offre, puis extraire les informations dont vous avez besoin pour afficher cette offre. Par exemple, demandez l'affichage d'une offre pour la zone inférieure droite de votre page Web (`planSummaryBottomRight`)

```
Response response=getOffers(sessionID, ip_planSummaryBottomRight, 1)
```

Cela renvoie un objet de réponse, notamment une réponse `OfferList`. Cependant, votre page Web ne peut pas utiliser un objet `OfferList`. Vous avez besoin d'un fichier image pour l'offre. Il constitue l'un des des attributs de l'offre (`offerImg`). Vous devez extraire l'attribut d'offre dont vous avez besoin à partir de la liste `OfferList`.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
 Offer offer = offerList.getRecommendedOffers()[0];
 NameValuePair[] attributes = offer.getAdditionalAttributes();
 for(NameValuePair attribute: attributes)
 {
  if(attribute.getName().equalsIgnoreCase("offerImg"))
  {
   /* Use this value in your code for the page, for
   example: stringHtml = "<img src="+attribute.getValueAsString()+ " >   */
  }
 }
}
```

Maintenant que l'offre est affichée, vous souhaitez la journaliser en tant que contact.

```
NameValuePair evtParam_TreatmentCode = new NameValuePairImpl();
evtParam_TreatmentCode.setName("UACIOfferTrackingCode");
evtParam_TreatmentCode.setValueAsString(offer.getTreatmentCode());
evtParam_TreatmentCode.setValueDataType(NameValuePair.DATA_TYPE_STRING);
postEvent(sessionID, evt_logOffer, evtParam_TreatmentCode)
```

Au lieu d'appeler chacune de ces méthodes séparément, vous pouvez utiliser la méthode `executeBatch`, comme illustré dans l'exemple suivant pour la partie `planSummaryBottomLeft` de la page Web.

```
Command getOffersCommand = new CommandImpl();
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(ip_planSummaryBottomLeft);
getOffersCommand.setNumberRequested(1);

Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEvent(evt_logOffer);

/** Build command array */
Command[] commands =
{
 getOffersCommand,
 postEventCommand
};

/** Make the call */
BatchResponse batchResponse = api.executeBatch(sessionId, commands);
```

Vous n'avez pas besoin de définir le UACIOfferTrackingCode dasn l'exemple car le serveur d'exécution Interact journalise automatiquement la dernière liste recommandée des traitements en tant que contacts si vous ne fournissez pas le UACIOfferTrackingCode.

Vous avez également écrit du code qui permet de changer l'image affichée toutes les 30 secondes. Cela permet à la deuxième zone de la page d'afficher une offre de mise à niveau du téléphone. Vous avez décidé d'alterner entre trois images. Pour extraire l'ensemble des offres à mettre en cache et les utiliser dans votre code pour alterner les images, utilisez ce qui suit.

```
Response response=getOffers(sessionID, ip_planSummaryBottomLeft, 3)
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
 for(int x=0;x<3;x++)
 {
  Offer offer = offerList.getRecommendedOffers()[x];
  if(x==0)
  {
   // grab offerimg attribute value and store somewhere;
   // this will be the first image to display
  }
  else if(x==1)
  {
   // grab offerimg attribute value and store somewhere;
   // this will be the second image to display
  }
  else if(x==2)
  {
   // grab offerimg attribute value and store somewhere;
   // this will be the third image to display
  }
 }
}
```

Vous devez écrire votre extraction de code client à partir du cache local et journaliser pour le contact une fois seulement pour chaque offre après l'affichage de l'image de l'offre. Pour journaliser le contact, le paramètre UACITrackingCode doit être publié de la même façon qu'auparavant. Chaque offre aura un code de suivi différent.

```
NameValuePair evtParam_TreatmentCodeSTR = new NameValuePairImpl();
NameValuePair evtParam_TreatmentCodeSBR = new NameValuePairImpl();
NameValuePair evtParam_TreatmentCodeSBL = new NameValuePairImpl();
```

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
 {
 for(int x=0;x<3;x++)
  {
  Offer offer = offerList.getRecommendedOffers()[x];
  if(x==0)
  {
   evtParam_TreatmentCodeSTR.setName("UACIOfferTrackingCode");
   evtParam_TreatmentCodeSTR.setValueAsString(offer.getTreatmentCode());
   evtParam_TreatmentCodeSTR.setValueDataType(NameValuePair.DATA_TYPE_STRING);
  }
  else if(x==1)
  {
   evtParam_TreatmentCodeSBR.setName("UACIOfferTrackingCode");
   evtParam_TreatmentCodeSBR.setValueAsString(offer.getTreatmentCode());
   evtParam_TreatmentCodeSBR.setValueDataType(NameValuePair.DATA_TYPE_STRING);
  }
  else if(x==2)
  {
   evtParam_TreatmentCodeSBL.setName("UACIOfferTrackingCode");
   evtParam_TreatmentCodeSBL.setValueAsString(offer.getTreatmentCode());
   evtParam_TreatmentCodeSBL.setValueDataType(NameValuePair.DATA_TYPE_STRING);
  }
 }
}
```

Pour chaque offre, si l'offre est sélectionnée (par un clic), vous devez journaliser l'offre acceptée et l'offre refusée. (Dans ce scénario, les offre non sélectionnées explicitement sont considérées comme refusées.) Voici un exemple d'un cas où l'offre `ip_planSummaryTopRight` est sélectionnée :

```
postEvent(sessionID, evt_offerAccept, evtParam_TreatmentCodeSTR)
postEvent(sessionID, evt_offerReject, evtParam_TreatmentCodeSBR)
postEvent(sessionID, evt_offerReject, evtParam_TreatmentCodeSBL)
```

Dans la pratique, il serait préférable d'envoyer ces trois appels `postEvent` avec la méthode `executeBatch`.

Il s'agit d'un exemple de base, et ne décrit pas le meilleur moyen d'écrire l'intégration. Par exemple, aucun de ces exemples n'inclut de vérification d'erreur avec la classe Response.

Conception de l'intégration de l'API Interact

La génération de votre intégration de l'API Interact API à votre point de contact requiert un certain nombre de tâches de conception avant de commencer l'implémentation. Vous devez travailler avec votre équipe marketing pour décider à quel endroit de votre point de contact vous souhaitez que l'environnement d'exécution propose des offres (pour définir vos points d'interaction) et choisir un autre type de suivi ou de fonctionnalité interactive à utiliser (pour définir vos événements). Dans la phase de conception, ces décisions peuvent être très schématiques. Par exemple, pour un site Web de télécommunications, la page de résumé du plan du client doit afficher une offre client de mise à niveau du plan, et deux offres pour les mises à niveau de téléphone.

Lorsque votre société a décidé où et comment elle souhaite interagir avec ses clients, vous devez utiliser Interact pour définir les détails. Un auteur de diagramme doit concevoir des diagrammes interactifs qui seront utilisés lorsque des événements de resegmentation se produisent. Vous devez choisir le nombre et les noms des points d'interaction et des événements, ainsi que des données qui

doivent être transmises pour que la segmentation, la publication d'événement et la récupération des offres se déroulent correctement. L'utilisateur de l'environnement de conception définit les points d'interaction et d'événements pour le canal interactif. Vous pouvez ensuite utiliser ces noms lorsque vous codez l'intégration à votre point de contact dans l'environnement d'exécution. Vous devez également définir les informations de mesure requises pour déterminer quand vous devez vous journaliser les des contacts et les réponses des offres.

Points à prendre en compte

Gardez à l'esprit les conseils suivants lorsque vous écrivez votre intégration.

- Lors de la conception de votre point de contact, créez un contenu de remplissage par défaut (généralement un message de marque bénin ou de contenu vide) pour chaque point d'interaction où les offres peuvent être présentées. Il s'agit de parer au cas où il n'existe aucune offre à proposer au visiteur en cours dans la situation actuelle. Vous devez affecter ce contenu de remplissage par défaut en tant que la chaîne par défaut du point d'interaction.
- Lors de la conception de votre point de contact, incluez une méthode présentant le contenu au cas où votre point de contact ne pourrait pas atteindre le groupe de serveurs d'exécution pour une raison imprévue.
- Lors du déclenchement d'événements qui resegmentent votre visiteur, y compris `postEvent` et `setAudience`, n'oubliez pas que l'exécution de diagrammes demandent un certain temps. La méthode `getOffers` attend que la segmentation soit terminée avant de s'exécuter. Une resegmentation trop fréquente peut dégrader mes performances de réponse d'appel `getOffers`.
- Vous devez décider ce que signifie un "refus d'offre". Plusieurs rapports, comme le rapport de récapitulatif des performances des offres du canal, présentent le nombre de refus d'une offre. Il s'agit d'un nombre de fois où l'action Journaliser le refus de l'offre a été déclenchée par `postEvent`. Vous devez déterminer si Journaliser le refus de l'offre correspond à un refus réel (par exemple un clic sur un lien intitulé "Non, merci") ou à une offre qui a été ignorée (par exemple une page affichant trois publicités de bannières différentes, dont aucun n'est sélectionnée).
- Il existe plusieurs fonctions facultatives que vous pouvez activer pour améliorer l'offre de sélection de Interact , y compris l'apprentissage, la suppression de l'offre, les affectations d'offre individuelle et d'autres éléments de la proposition de l'offre. Vous devez déterminer combien, le cas échéant, de ces fonctions facultatives amélioreraient vos interactions.

Chapitre 6. Gestion de l'API IBM Unica Interact

Chaque fois que vous utilisez la méthode `startSession` , vous créez une session d'exécution Interact sur le serveur d'exécution. Vous pouvez utiliser les propriétés de configuration pour gérer les sessions sur un serveur d'exécution. Il peut être nécessaire de configurer ces paramètres lorsque vous implémentez votre intégration Interact avec votre point de contact.

Ces propriétés de configuration se trouvent dans la catégorie `sessionManagement`.

Paramètres régionaux et API Interact

Vous pouvez utiliser Interact pour les points de contact qui ne sont pas en anglais. Le point de contact et toutes les chaînes de l'API utilisent les paramètres régionaux définis pour l'utilisateur de l'environnement d'exécution.

Vous pouvez sélectionner un seul groupe de paramètres régionaux par groupe de serveurs.

Par exemple, dans l'environnement d'exécution, vous créez deux utilisateurs, asm_admin_en avec des paramètres régionaux définis comme l'anglais, et asm_admin_fr des paramètres régionaux définis comme le français. Si votre point de contact est conçu pour des francophones, définissez la propriété `asmUserForDefaultLocale` pour l'environnement d'exécution en tant que asm_admin_fr.

A propos de la surveillance JMX

Interact fournit le service de surveillance Java Management Extensions (JMX) auquel vous pouvez accéder via n'importe quelle application de surveillance JMX. La surveillance JMX vous permet de surveiller et de gérer vos serveurs d'exécution. Les attributs JMX fournissent une grande quantité d'informations détaillées sur le serveur d'exécution. Par exemple, l'attribut `ErrorCount` indique le nombre de messages d'erreur consignés depuis la dernière réinitialisation ou le dernier démarrage du système. Vous pouvez utiliser ces informations pour voir la fréquence des erreurs dans votre système. Si vous avez codé votre site Web de façon à ce qu'il n'appelle une session de fin que si un utilisateur termine une transaction, vous pouvez également comparer `startSessionCount` avec `endSessionCount` pour voir le nombre de transactions incomplètes.

Interact prend en charge les protocoles RMI et JMXMP, comme le définit JSR 160. Vous pouvez vous connecter au service de surveillance JMX avec n'importe quel client compatible JSR160.

Les diagrammes interactifs peuvent être contrôlés uniquement avec la surveillance JMX. Les informations relatives aux diagrammes interactifs n'apparaissent pas dans Campaign Monitoring.

Remarque : Si vous utilisez IBM WebSphere avec un gestionnaire de noeud, vous devez définir l'argument générique JVM pour activer la surveillance JMX.

© Copyright IBM Corp. 2001, 2012

Configuration d'Interact pour une utilisation de la surveillance JMX avec le protocole RMI

Dans Marketing Platform pour l'environnement de conception, éditez les propriétés de configuration suivantes dans la catégorie `Interact > monitoring`.

Propriété de configuration	Paramètre
`protocole`	**RMI**
`port`	Numéro de port du service JMX
`enableSecurity`	**False** L'implémentation Interact du protocole RMI ne prend pas en charge la sécurité.

L'adresse par défaut pour la surveillance pour le protocole RMI est : `service:jmx:rmi:///jndi/rmi://`*`RuntimeServer`*`:`*`port`*`/interact`.

Configuration d'Interact pour une utilisation de la surveillance JMX avec le protocole JMXMP

Le protocole JMXMP nécessite deux bibliothèques supplémentaires dans l'ordre suivant dans le chemin d'accès aux classes `InteractJMX.jar` et `jmxremote_optional.jar`. Ces deux fichiers peuvent être trouvés dans le répertoire `lib` de votre installation d'environnement d'exécution.

Remarque : Si vous activez la sécurité, le nom d'utilisateur et le mot de passe doivent correspondre à un utilisateur dans Marketing Platform pour l'environnement d'exécution. Vous ne pouvez pas utiliser de mot de passe vide.

Dans Marketing Platform, pour l'environnement de conception, éditez les propriétés de configuration suivantes dans la catégorie `Interact > monitoring`.

Propriété de configuration	Paramètre
`protocole`	**JMXMP**
`port`	Numéro de port du service JMX
`enableSecurity`	**False** pour désactiver la sécurité, ou **True** pour activer la sécurité

L'adresse par défaut pour la surveillance pour le protocole JMXMP est : `service:jmx:jmxmp://`*`RuntimeServer`*`:`*`port`*.

Utilisation des scripts jconsole

Si vous ne disposez pas d'une application de surveillance JMX distincte, vous pouvez utiliser le jconsole installée avec la machine virtuelle Java. Vous pouvez démarrer jconsole à l'aide des scripts de démarrage dans le répertoire `Interact/tools`.

1. Ouvrez `Interact\tools\jconsole.bat` (Windows) ou `Interact/tools/jconsole.sh` (Unix) dans un éditeur de texte.
2. Définissez `INTERACT_LIB` sur le chemin d'accès complet au répertoire *`InteractInstallationDirectory`*`/lib`.

3. Définissez `HOTE` sur le nom d'hôte du serveur d'exécution que vous souhaitez surveiller.
4. Définissez `PORT` sur le port que JMX est configuré pour écouter avec la propriété `Interact > monitoring > port`.
5. Si vous surveillez via le protocole RMI, ajouter un commentaire avant la connexion JMXMP et supprimez le commentaire avant la connexion RMI.

   Le script effectue une surveillance via le protocole JMXMP par défaut.

   Par exemple, voir les paramètres par défaut de `jconsole.bat`.

   La connexion JMXMP

   ```
   %JAVA_HOME%\bin\jconsole.exe -J-Djava.class.path=%JAVA_HOME%\lib\jconsole.jar;
     INTERACT_LIB%\interactJMX.jar; INTERACT_LIB%\jmxremote_optional.jar
     service:jmx:jmxmp://%HOST%:%PORT%
   ```

   La connexion RMI

   ```
   %JAVA_HOME%\bin\jconsole.exe -J-Djava.class.path=%JAVA_HOME%\lib\jconsole.jar;
        INTERACT_LIB%\jmxremote_optional.jar
        service:jmx:rmi:///jndi/rmi://%HOST%:%PORT%/interact
   ```

Attributs JMX

Les tableaux suivants décrivent les attributs disponibles avec la surveillance JMX.

Toutes les données fournies par la surveillance JMX le sont depuis la dernière réinitialisation ou le dernier démarrage du système. Par exemple, un décompte correspond au nombre des articles depuis la dernière réinitialisation ou le dernier démarrage du système, pas depuis l'installation.

Tableau 8. Moniteur ETL de l'historique des réponses des contacts

Attribut	Description
AvgCHExecutionTime	Nombre de millisecondes nécessaire au module de l'historique des réponses et des contacts pour écrire dans la table de l'historique des contacts. Cette moyenne est calculée uniquement pour les opérations qui ont réussi et pour lesquelles au moins un enregistrement a été écrit dans la table d'historique des contacts.
AvgETLExecutionTime	Nombre de millisecondes moyen nécessaire au module de l'historique des réponses et des contacts pour lire des données depuis l'environnement d'exécution. La moyenne comprend la durée des opérations ayant réussi mais aussi de celles ayant échoué.
AvgRHExecutionTime	Nombre de millisecondes moyen nécessaire au module de l'historique des réponses et des contacts pour écrire dans la table d'historique des réponses. Cette moyenne est calculée uniquement pour les opérations qui ont réussi et pour lesquelles au moins un enregistrement a été écrit dans la table d'historique des réponses.

Tableau 8. Moniteur ETL de l'historique des réponses des contacts (suite)

Attribut	Description
ErrorCount	Nombre des messages d'erreur consignés depuis la dernière réinitialisation ou le dernier démarrage du système, le cas échéant.
HighWaterMarkCHExecutionTime	Nombre maximum de millisecondes nécessaire au module de l'historique des réponses et des contacts pour écrire dans la table de l'historique des contacts. Cette valeur est calculée uniquement pour les opérations qui ont réussi et pour lesquelles au moins un enregistrement a été écrit dans la table d'historique des contacts.
HighWaterMarkETLExecutionTime	Nombre de millisecondes maximum nécessaire au module de l'historique des réponses et des contacts pour lire des données depuis l'environnement d'exécution. Le calcul comprend la durée des opérations ayant réussi mais aussi de celles ayant échoué.
HighWaterMarkRHExecutionTime	Nombre maximum de millisecondes nécessaire au module de l'historique des réponses et des contacts pour écrire dans la table de l'historique des réponses. Cette valeur est calculée uniquement pour les opérations qui ont réussi et pour lesquelles au moins un enregistrement a été écrit dans la table d'historique des réponses.
LastExecutionDuration	Nombre de millisecondes nécessaire au module de l'historique des contacts et des réponses pour exécuter la dernière copie.
NumberOfExecutions	Nombre d'exécutions du module de l'historique des contacts et des réponses depuis l'initialisation.
LastExecutionStart	Date et heure du démarrage de la dernière exécution du module de l'historique des contacts et des réponses.
LastExecutionSuccessful	Si cette condition est true, la dernière exécution du module de l'historique des contacts et des réponses a réussie. Si sa valeur est false, elle signifie qu'une erreur s'est produite.
NumberOfContactHistoryRecordsMarked	Nombre d'enregistrements de l'historique des contacts dans la table `UACI_CHStaging` qui est déplacée pendant l'exécution en cours du module de l'historique des contacts et des réponses. Cette valeur est supérieure à zéro uniquement si le module de l'historique des contacts et des réponses est en cours d'exécution.

Tableau 8. Moniteur ETL de l'historique des réponses des contacts (suite)

Attribut	Description
NumberOfResponseHistoryRecordsMarked	Nombre d'enregistrements de l'historique des réponses dans la table `UACI_RHStaging` qui est déplacée pendant l'exécution en cours du module de l'historique des contacts et des réponses. Cette valeur est supérieure à zéro uniquement si le module de l'historique des contacts et des réponses est en cours d'exécution.

Les attributs du Moniteur ETL de l'historique des contacts et des réponses font partie de l'environnement de conception. Tous les attributs suivants font partie de l'environnement d'exécution.

Tableau 9. Exceptions

Attribut	Description
errorCount	Nombre de messages d'erreur consignés depuis la dernière réinitialisation ou le dernier démarrage du système.
warningCount	Nombre de messages d'avertissement consignés depuis la dernière réinitialisation ou le dernier démarrage du système.

Tableau 10. Statistiques du moteur de diagramme

Attribut	Description
activeProcessBoxThreads	Décompte actif des unités d'exécution de processus du diagramme (partagées entre toutes les exécutions) qui sont en cours d'exécution.
activeSchedulerThreads	Décompte actif des unités d'exécution du planificateur de diagramme (Flowchart Scheduler) qui sont en cours d'exécution.
avgExecutionTimeMillis	Durée d'exécution moyenne du diagramme en millisecondes.
CurrentJobsInProcessBoxQueue	Nombre de travaux en attente d'être exécutés par les unités d'exécution de processus du diagramme.
CurrentJobsInSchedulerQueue	Nombre de travaux en attente d'être exécutés par les unités d'exécution du planificateur de diagramme (Flowchart Scheduler).
maximumProcessBoxThreads	Nombre maximum des unités d'exécution de processus du diagramme (partagées entre toutes les exécutions) pouvant être exécutées.
maximumSchedulerThreads	Nombre maximal d'unités d'exécution du planificateur de diagramme (une unité d'exécution par exécution) pouvant être exécutées.

Tableau 10. Statistiques du moteur de diagramme (suite)

Attribut	Description
numExecutionsCompleted	Nombre total d'exécutions de diagramme qui se sont terminées.
numExecutionsStarted	Nombre total d'exécutions de diagramme qui ont démarré.

Tableau 11. Diagrammes spécifiques par canal interactif

Attribut	Description
AvgExecutionTimeMillis	Durée d'exécution moyenne en millisecondes pour ce diagramme dans ce canal interactif.
HighWaterMarkForExecutionTime	Durée d'exécution maximum en millisecondes pour ce diagramme dans ce canal interactif.
LastCompletedExecutionTimeMillis	Durée d'exécution en millisecondes de la dernière exécution de ce diagramme dans ce canal interactif.
NumExecutionsCompleted	Nombre total d'exécutions qui se sont terminées pour ce diagramme dans ce canal interactif.
NumExecutionsStarted	Nombre total d'exécutions démarrées pour ce diagramme dans ce canal interactif.

Tableau 12. Paramètres régionaux

Attribut	Description
paramètres régionaux	Paramètres régionaux du client JMX.

Tableau 13. Configuration du consignateur

Attribut	Description
{\f2 Cat}é{\f2 gorie :}	Modifiez la catégorie de journal dans laquelle le niveau de journalisation peut être manipulé.

Tableau 14. Statistiques du pool d'unités d'exécution des services

Attribut	Description
activeContactHistThreads	Nombre approximatif d'unités d'exécution qui exécutent activement les tâches pour l'historique des contacts et l'historique des réponses.
activeFlushCacheToDBThreads	Nombre approximatif d'unités d'exécution qui exécutent activement les tâches pour vider les statistiques de mise en cache vers le magasin de données.

Tableau 14. Statistiques du pool d'unités d'exécution des services (suite)

Attribut	Description
activeOtherStatsThreads	Nombre approximatif d'unités d'exécution qui exécutent activement les tâches pour les statistiques relatives à l'offre d'éligibilité, les activités de l'événement et les statistiques par défaut.
CurrentHighWaterMarkInContactHistQueue	Nombre le plus important d'entrées en file d'attente devant être consignées par le service qui collecte les données de l'historique des contacts et des réponses.
CurrentHighWaterMark InFlushCachetoDBQueue	Nombre le plus important d'entrées en file d'attente devant être consignées par le service qui écrit les données de la mémoire cache dans les tables de la base de données.
CurrentHighWaterMarkInOtherStatsQueue	Nombre le plus important d'entrées en file d'attente devant être journalisées par le service qui collecte les statistiques relatives à l'offre d'éligibilité, les statistiques sur l'utilisation de chaîne par défaut, les statistiques sur l'activité des événements, et le journal personnalisé pour les envoyer aux données de la table.
currentMsgsInContactHistQueue	Nombre de travaux dans la file d'attente pour le pool d'unités d'exécution utilisé pour l'historique des contacts et l'historique des réponses.
currentMsgsInFlushCacheToDBQueue	Nombre de travaux dans la file d'attente pour le pool d'unités d'exécution utilisées pour vider les statistiques mises en cache vers le magasin de données.
currentMsgsInOtherStatsQueue	Nombre de travaux dans la file d'attente pour le pool d'unités d'exécution utilisé pour les statistiques relatives à l'offre d'éligibilité, l'activité d'événement et les statistiques par défaut.
maximumContactHistThreads	Nombre le plus important d'unités d'exécution s'étant trouvées simultanément dans le pool utilisé pour l'historique des contacts et l'historique des réponses.
maximumFlushCacheToDBThreads	Nombre le plus important d'unités d'exécution s'étant trouvées simultanément dans le pool utilisé pour vider les statistiques mises en cache vers le magasin des données.

Tableau 14. Statistiques du pool d'unités d'exécution des services (suite)

Attribut	Description
maximumOtherStatsThreads	Nombre le plus important d'unités d'exécution s'étant trouvées simultanément dans le pool utilisé pour les statistiques relatives à l'offre d'éligibilité, les activités de l'événement et les statistiques par défaut.

Les statistiques de service se composent d'un ensemble d'attributs pour chaque service.

- ContactHistoryMemoryCacheStatistics — Service qui collecte les données pour les tables de transfert de l'historique des contacts.
- CustomLoggerStatistics — Service qui collecte les données personnalisées à écrire dans une table (un événement utilisant le paramètre d'événement `UACICustomLoggerTableName`).
- Statistiques par défaut — Le service qui collecte les statistiques relatives au nombre d'utilisations de la chaîne par défaut du point d'interaction a été utilisé.
- Statistiques relatives à l'offre d'éligibilité — Le service qui collecte les statistiques relatives aux offres d'éligibilité.
- Statistiques de l'activité de l'événement - Le service qui collecte les statistiques d'événements, c'est-à-dire à la fois les événements système tels que `getOffer` ou `startSession` et les événements utilisateur déclenchés par `postEvent`.
- Statistiques de cache mémoire de l'historique des réponses — Service qui écrit dans les tables de transfert de l'historique des réponses.
- Statistiques de réponse intersession — Service qui collecte les données de suivi de réponse intersessions.

Tableau 15. Statistiques de service

Attribut	Description
Décompte	Nombre de messages traités.
ExecTimeInsideMutex	Temps passé à traiter les messages de ce service, à l'exclusion du temps d'attente des autres unités d'exécution, en millisecondes. S'il y existe une grande différence entre ExecTimeInsidMutex et ExecTimeMillis, il peut être nécessaire de modifier la taille du pool d'unités d'exécution du service.
ExecTimeMillis	Temps passé à traiter les messages de ce service, y compris le temps d'attente des autres unités d'exécution, en millisecondes.
ExecTimeOfDBInsertOnly	Durée en millisecondes consacré au traitement de la partie d'insertion par lots uniquement.
HighWaterMark	Nombre maximum de messages traités pour ce service.
NumberOfDBInserts	Nombre total d'insertions par lots exécutées.

Tableau 15. Statistiques de service (suite)

Attribut	Description
TotalRowsInserted	Nombre total de lignes insérées dans la base de données.

Tableau 16. Statistiques de service - Utilitaire de chargement de la base de données

Attribut	Description
ExecTimeOfWriteToCache	Durée en millisecondes consacré à l'écriture dans le cache des fichiers, y compris l'écriture dans les fichiers et l'obtention de la clé primaire à partir de la base de données si nécessaire.
ExecTimeOfLoaderDBAccessOnly	Durée en millisecondes consacrée à l'exécution de la partie du chargeur de base de données uniquement.
ExecTimeOfLoaderThreads	Durée en millisecondes correspondant aux unités d'exécution du chargeur de base de données.
ExecTimeOfFlushCacheFiles	Durée en millisecondes consacrée au vidage de la cache et à la création de nouveaux caches.
ExecTimeOfRetrievePKDBAccess	Durée en millisecondes consacrée à la récupération de l'accès de la base de données à la clé primaire.
NumberOfDBLoaderRuns	Nombre total d'exécutions du chargeur de base de données.
NumberOfLoaderStagingDirCreated	Nombre total de répertoires de transfert créés.
NumberOfLoaderStagingDirRemoved	Nombre total de répertoires de transfert supprimés.
NumberOfLoaderStagingDirMovedToAttention	Nombre total de répertoires de transfert renommés attention.
NumberOfLoaderStagingDirMovedToError	Nombre total de répertoires de transfert renommés erreur.
NumberOfLoaderStagingDirRecovered	Nombre total de répertoires de transfert récupérées, y compris lors du démarrage et de la réexécution par les unités d'exécution en arrière-plan.
NumberOfTimesRetrievePKFromDB	Nombre total d'extractions de la clé primaire depuis la base de données.
NumberOfLoaderThreadsRuns	Nombre total d'exécutions des unités d'exécution du chargeur de base de données.
NumberOfFlushCacheFiles	Nombre total de vidages du cache des fichiers.

Tableau 17. Statistiques API

Attribut	Description
endSessionCount	Nombre d'appels de l'API `endSession` depuis la dernière réinitialisation ou le dernier démarrage du système.
endSessionDuration	Temps écoulé pour le dernier appel de l'API `endSession`.
executeBatchCount	Nombre d'appels de l'API `executeBatch` depuis la dernière réinitialisation ou le dernier démarrage du système.
executeBatchDuration	Temps écoulé pour le dernier appel de l'API `executeBatch`.
getOffersCount	Nombre d'appels de l'API `getOffers` depuis la dernière réinitialisation ou le dernier démarrage du système.
getOffersDuration	Temps écoulé pour le dernier appel de l'API `getOffer`.
getProfileCount	Nombre d'appels de l'API `getProfile` depuis la dernière réinitialisation ou le dernier démarrage du système.
getProfileDuration	Temps écoulé pour le dernier appel de l'API `getProfileDuration`.
getVersionCount	Nombre d'appels de l'API `getVersion` depuis la dernière réinitialisation ou le dernier démarrage du système.
getVersionDuration	Temps écoulé pour le dernier appel de l'API `getVersion`.
loadOfferSuppressionDuration	Temps écoulé pour le dernier appel de l'API `loadOfferSuppression`.
LoadOffersBySQLCount	Nombre d'appels de l'API `LoadOffersBySQL` depuis la dernière réinitialisation ou le dernier démarrage du système.
LoadOffersBySQLDuration	Temps écoulé pour le dernier appel de l'API `LoadOffersBySQL`.
loadProfileDuration	Temps écoulé pour le dernier appel de l'API `loadProfile`.
loadScoreOverrideDuration	Temps écoulé pour le dernier appel de l'API `loadScoreOverride`.
postEventCount	Nombre d'appels de l'API `postEvent` depuis la dernière réinitialisation ou le dernier démarrage du système.
postEventDuration	Temps écoulé pour le dernier appel de l'API `postEvent`.
runSegmentationDuration	Temps écoulé pour le dernier appel de l'API `runSegmentation`.
setAudienceCount	Nombre d'appels de l'API `setAudience` depuis la dernière réinitialisation ou le dernier démarrage du système.

Tableau 17. Statistiques API (suite)

Attribut	Description
setAudienceDuration	Temps écoulé pour le dernier appel de l'API `setAudience`.
setDebugCount	Nombre d'appels de l'API `setDebug` depuis la dernière réinitialisation ou le dernier démarrage du système.
setDebugDuration	Temps écoulé pour le dernier appel de l'API `setDebug`.
startSessionCount	Nombre d'appels de l'API `startSession` depuis la dernière réinitialisation ou le dernier démarrage du système.
startSessionDuration	Temps écoulé pour le dernier appel de l'API `startSession`.

Tableau 18. Statistiques de l'optimiseur d'apprentissage

Attribut	Description
LearningOptimizerAcceptCalls	Nombre événements d'acceptation transmis au module d'apprentissage.
LearningOptimizerAcceptTrackingDuration	Nombre total de millisecondes consacré à la consignation des événements d'acceptation dans le module d'apprentissage.
LearningOptimizerContactCalls	Nombre événements de contact transmis au module d'apprentissage.
LearningOptimizerContactTrackingDuration	Nombre total de millisecondes consacré à la consignation des événements de contact dans le module d'apprentissage.
LearningOptimizerLogOtherCalls	Nombre événements autres que les événements de contact et d'acceptation transmis au module d'apprentissage.
LearningOptimizerLogOtherTrackingDuration	Durée en millisecondes consacrée à la consignation des autres événements (ni contact, ni acceptation) dans le module d'apprentissage.
LearningOptimizerNonRandomCalls	Nombre d'applications de l'implémentation d'apprentissage configurée.
LearningOptimizerRandomCalls	Nombre de fois où l'implémentation d'apprentissage configuré a été contournée et où la sélection aléatoire a été appliquée.
LearningOptimizerRecommendCalls	Nombre de demandes de recommandation transmises au module d'apprentissage.
LearningOptimizerRecommendDuration	Durée totale en millisecondes consacrée à la logique de recommandation d'apprentissage.

Tableau 19. Statistiques d'offre par défaut

Attribut	Description
LoadDefaultOffersDuration	Temps écoulé pour le chargement des offres par défaut.
DefaultOffersCalls	Nombre de chargements des offres par défaut.

Opérations JMX

Le tableau suivant décrit les opérations disponibles pour la surveillance JMX.

Groupe	Attribut	Description
Configuration du consignateur	activateDebug	Définissez le niveau de journalisation pour le fichier journal défini dans `Interact/conf/interact_log4j.properties` sur débogage.
Configuration du consignateur	activateError	Définissez le niveau de journalisation pour le fichier journal défini dans `Interact/conf/interact_log4j.properties` sur erreur.
Configuration du consignateur	activateFatal	Définissez le niveau de journalisation pour le fichier journal défini dans `Interact/conf/interact_log4j.properties` sur fatal.
Configuration du consignateur	activateInfo	Définissez le niveau de journalisation pour le fichier journal défini dans `Interact/conf/interact_log4j.properties` sur info.
Configuration du consignateur	activateTrace	Définissez le niveau de journalisation pour le fichier journal défini dans `Interact/conf/interact_log4j.properties` sur trace.
Configuration du consignateur	activateWarn	Définissez le niveau de journalisation pour le fichier journal défini dans `Interact/conf/interact_log4j.properties` sur avertissement.
Paramètres régionaux	changeLocale	Modifiez les paramètres régionaux du client JMX. Les paramètres régionaux Interact pris en charge sont `de`, `en`, `es` et `fr`.
ContactResponseHistory ETLMonitor	reset	Réinitialisez tous les compteurs.
Statistiques d'offre par défaut	updatePollPeriod	Met à jour defaultOfferUpdatePollPeriod. Cette valeur, en secondes, indique au système la durée d'attente à observer avant de mettre à jour les offres par défaut dans le cache. Si la valeur est définie sur `-1`, le système ne lit que le nombre d'offres par défaut au démarrage.

Chapitre 7. Classes et méthodes de l'API IBM Unica Interact

Les sections suivantes listent les exigences et d'autres informations que vous devez connaître avant d'utiliser l'API Interact.

Remarque : Cette section suppose que vous soyez familiarisé avec votre point de contact, le langage de programmation Java , et que vous travaillez avec une API Java.

L'API Interact dispose d'un adaptateur client Java qui utilise une sérialisation Java via HTTP. En outre, Interact fournit un WSDL pour la prise en charge des clients SOAP. Le WSDL présente le même ensemble de fonctions que l'd'adaptateur de client Java, de sorte que les sections suivantes, à l'exception des exemples, continuent de s'appliquer.

Interact API Classes

L'API Interact se base sur la classe `InteractAPI`. Il existe 6 interfaces de prise en charge.

- `AdvisoryMessage`
- `BatchResponse`
- `NameValuePair`
- `Offre`
- `OfferList`
- `Response`

Ces interfaces comportent 3 classes concrètes de prise en charge. Les deux classes concrètes suivantes doivent être instanciées et transmises en tant qu'arguments dans les méthodes d'API Interact :

- NameValuePairImpl
- CommandImpl

Une troisième classe concrète, appelée `AdvisoryMessageCode` est disponible pour fournir les constantes utilisées pour distinguer les codes de message renvoyés par le serveur chaque fois que nécessaire.

Le reste de cette section décrit les méthodes qui composent l'API Interact.

Prérequis de la sérialisation Java via HTTP

1. Avant de commencer à utiliser l'adaptateur de sérialisation Java, vous devez ajouter le fichier suivant à votre `CLASSPATH` :

   `Interact_Runtime_Environment_Installation_Directory/lib/interact_client.jar`
2. Tous les objets échangés entre le client et le serveur se trouvent dans le module `com.unicacorp.interact.api`. Voir le Javadoc de l'API Interact pour plus de détails.
3. Pour obtenir une instance de la classe InteractAPI, appelez la méthode statique `getInstance` avec l'URL du serveur d'exécution Interact.

© Copyright IBM Corp. 2001, 2012

Prérequis SOAP

Important : Les tests de performances démontrent que l'adaptateur de sérialisation Java a un débit beaucoup plus élevé qu'un client SOAP généré. Pour des performances optimales, utilisez l'adaptateur de sérialisation Java chaque fois que possible.

Pour accéder au serveur d'exécution à l'aide de SOAP, vous devez procéder comme suit :

1. Convertissez l'API WSDL Interact API à l'aide du kit d'outils SOAP de votre choix.

   L'API WSDL Interact API est installée dans Interact dans le répertoire `Interact/conf`.

   Le texte du WSDL est disponible à la fin de ce guide.
2. Installez et configurez le serveur d'exécution.

   Le serveur d'exécution doit être actif pour tester entièrement votre intégration.

Versions SOAP

Interact utilise axis2 1,3 comme infrastructure SOAP sur les serveurs d'exécution Interact. Pour plus de détails sur ce que les versions de SOAP axis2 1.3 prend en charge, consultez le site Web suivant :

Apache Axis2

Interact a été testé avec les clients axis2, XFire, JAX-WS-Ri, DotNet, SOAPUI et IBM RAD SOAP.

API JavaDoc

En complément de ce guide, la documentation JavaDoc de l'API Interact est installée avec le serveur d'exécution. La JavaDoc est installée à titre de votre référence dans le répertoire `Interact/docs/apiJavaDoc`.

A propos des exemples d'API

Tous les exemples contenus dans ce guide ont été créés à l'aide de la sérialisation Java via l'adaptateur HTTP. Si vous utilisez SOAP, car les classes générés à partir du WSDL peuvent varier en fonction du kit d'outils SOAP et les options que vous sélectionnez, ces exemples peuvent ne pas fonctionner exactement de la même façon dans votre environnement.

Gestion des données de session

Lorsque vous lancez une session avec la méthode `startSession`, les données de session sont chargées en mémoire. Tout au long de la session, vous pouvez lire et écrire les données de session (qui sont un sur-ensemble du profil de données statiques). La session contient les données suivantes :

- Données de profil statique
- Affectations de segments
- Données en temps réel
- Recommandations d'offres

Toutes les données de session sont disponibles jusqu'à ce que vous appeliez la méthode `endSession`, ou que le délai `sessionTimeout` soit écoulé. À la fin de la session, toutes les données qui ne sont pas explicitement sauvegardées dans l'historique des contacts ou des réponses ou dans une autre table de base de données sont perdues.

Les données sont stockées sous la forme d'un ensemble de paires nom/valeur. Si les données sont lues à partir d'une table de base de données, le nom est la colonne de la table.

Vous pouvez créer ces paires nom-valeur lorsque vous utilisez l'API Interact. Vous n'avez pas besoin de déclarer toutes les paires nom-valeur dans une zone globale. Si vous définissez de nouveaux paramètres d'événement en tant que paires nom/valeur, l'environnement d'exécution ajoute des paires nom-valeur aux données de session. Par exemple, si vous utilisez les paramètres d'événement avec la méthode `postEvent` , l'environnement d'exécution ajoute les paramètres d'événement aux données de session, même si les paramètres d'événement n'étaient pas disponibles dans les données de profil. Ces données existent uniquement dans les données de session.

Vous pouvez écraser les données de session à tout moment. Par exemple, si une partie du profil client inclut `creditScore`, vous pouvez transmettre un paramètre d'événement avec le type personnalisé `NameValuePair`. Dans la classe `NameValuePair`, vous pouvez utiliser les méthodes `setName` et `setValueAsNumeric` pour modifier la valeur. Le nom doit correspondre. Dans les données de session, le nom n'est pas sensible à la casse. Par conséquent, le nom `creditscore` ou `CrEdItScOrE` écraserait `creditScore`.

Seules les dernières données écrites dans les données de session sont conservées. Par exemple, `startSession` charge les données de profil pour la valeur `lastOffer`. La méthode `postEvent` écrase `lastOffer`. Une deuxième méthode `postEvent` écrase ensuite `lastOffer`. L'environnement d'exécution conserve uniquement les données écrites par la deuxième méthode `postEvent` dans les données de session.

Lorsque la session se termine, les données sont perdues, sauf si vous avez pris des mesures spéciales telles que l'utilisation d'un processus Extraction dans votre diagramme interactif pour écrire les données dans une table de base de données. Si vous envisagez d'utiliser des processus Extraction, n'oubliez pas que les noms doivent respecter les limites de votre base de données. Par exemple, si vous êtes autorisé à utiliser uniquement 256 caractères pour le nom d'une colonne, le nom de la paire nom-valeur ne doit pas dépasser 256 caractères.

A propos de la classe InteractAPI

La classe InteractAPI contient les méthodes que vous pouvez utiliser pour intégrer votre point de contact au serveur d'exécution. Toutes les classes et méthodes de l'API Interact prend en charge les méthodes de cette classe.

Vous devez compiler votre implémentation par rapport à `interact_client.jar` situé dans le répertoire `lib` de votre installation d'environnement d'exécution Interact.

endSession

`endSession(String` *`sessionID`*`)`

La méthode endSession marque la fin de la session d'exécution. Lorsque le serveur d'exécution reçoit cette méthode, il se connecte à l'historique, efface la mémoire, etc.

- **sessionID** — Chaîne unique identifiant la session.

Si la méthode endSession n'est pas appelée, les sessions d'exécution expirent. Le délai d'attente de session est configurable avec la propriété sessionTimeout.

Valeur de retour

Le serveur d'exécution répond à la méthode endSession avec un objet Response dans lequel les attributs suivants sont renseignés :

- SessionID
- ApiVersion
- StatusCode
- AdvisoryMessages

Exemple

L'exemple suivant illustre la méthode endSession et montre comment vous pouvez analyser la réponse. sessionId est la même chaîne permettant d'identifier la session utilisée par l'appel startSession qui a démarré cette session.

```
response = api.endSession(sessionId);
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("endSession call processed with no warnings or errors");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("endSession call processed with a warning");
    }
    else
    {
        System.out.println("endSession call processed with an error");
    }
    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("endSession",
   response.getAdvisoryMessages());
```

executeBatch

executeBatch(String *sessionID*, *CommandImpl*[] commands)

La méthode executeBatch vous permet d'exécuter plusieurs méthodes via une seule demande au serveur d'exécution.

- **sessionID** — Chaîne identifiant l'ID session. Cet ID de session est utilisé pour toutes les commandes exécutées par cet appel de méthode.
- **commandImpl[]** — Tableau d'objets CommandImpl, un pour chaque commande que vous souhaitez exécuter.

Le résultat de l'appel de cette méthode équivaut à appeler explicitement chaque méthode dans la table Commande. Cette méthode réduit le nombre de demandes réel au serveur d'exécution. Le serveur d'exécution exécute chaque méthode en série. Pour chaque appel, toute erreur ou tout avertissement est capturé dans l'objet de réponse qui correspond à cet appel de méthode. Si une erreur est détectée,

executeBatch continue à traiter le reste des appels dans le lot. Si l'exécution de toute méthode aboutit à une erreur, le statut de niveau supérieur de l'objet BatchResponse indique l'erreur. Si aucune erreur ne s'est produite, le statut de niveau supérieur reflète les avertissements qui ont pu se produire. Si aucun avertissement ne s'est produit, le statut de niveau supérieur indique une exécution réussie du lot.

Valeur de retour

Le serveur d'exécution répond à `executeBatch` avec un objet BatchResponse.

Exemple

L'exemple suivant montre comment appeler toutes les méthodes getOffer et postEvent avec un seul appel `executeBatch`, et donne une suggestion de gestion de la réponse.

```
/** Define all variables for all members of the executeBatch*/
String sessionId="MySessionID-123";
String interactionPoint = "Overview Page Banner 1";
int numberRequested=1;
String eventName = "logOffer";

/** build the getOffers command */
Command getOffersCommand = new CommandImpl();
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(interactionPoint);
getOffersCommand.setNumberRequested(numberRequested);

/** build the postEvent command  */
Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEventParameters(postEventParameters);
postEventCommand.setEvent(eventName);

/** Build command array */
Command[] commands =
{
  getOffersCommand,
  postEventCommand,
};

/** Make the call */
BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Process the response appropriately */
// Top level status code is a short cut to determine if there
// are any non-successes in the array of Response objects
if(batchResponse.getBatchStatusCode() == Response.STATUS_SUCCESS)
{
  System.out.println("ExecuteBatch ran perfectly!");
}
else if(batchResponse.getBatchStatusCode() == Response.STATUS_WARNING)
{
  System.out.println("ExecuteBatch call processed with at least one warning");
}
else
{
  System.out.println("ExecuteBatch call processed with at least one error");
}

// Iterate through the array, and print out the message for any non-successes
for(Response response : batchResponse.getResponses())
{
  if(response.getStatusCode()!=Response.STATUS_SUCCESS)
```

```
  {
  printDetailMessageOfWarningOrError("executeBatchCommand",
 response.getAdvisoryMessages());
  }
}
```

getInstance

```
getInstance(String URL)
```

La méthode `getInstance` crée une instance de l'API Interact qui communique avec le serveur d'exécution indiqué.

Important : Chaque application que vous créez à l'aide de l'API Interact doit appeler `getInstance` pour instancier un objet InteractAPI qui est mappé à un serveur d'exécution spécifié par le paramètre URL.

Pour les groupes de serveurs, si vous utilisez un équilibreur de charge, utilisez le nom d'hôte et le port que vous configurez avec l'équilibreur de charge. Si vous ne disposez pas d'un équilibreur de charge, vous devrez inclure la logique permettant d'alterner entre les serveurs d'exécution disponibles.

Cette méthode est applicable pour la sérialisation Java via l'adaptateur HTTP uniquement. Il n'existe pas de méthode correspondante définie dans le fichier WSDL SOAP. Chaque implémentation du client SOAP a sa propre façon de créer l'URL du noeud final.

- **URL** — Chaîne identifiant l'URL de l'instance d'exécution. Par exemple, `http://localhost:7001/Interact/servlet/InteractJSService`.

Valeur de retour

Le serveur d'exécution renvoie InteractAPI.

Exemple

L'exemple suivant montre comment instancier un objet InteractAPI pointant vers une instance de serveur d'exécution active sur la même machine que votre point de contact.

```
InteractAPI api=InteractAPI.getInstance("http://localhost:7001/interact/servlet/InteractJSService");
```

getOffers

```
getOffers(String sessionID, String interactionPoint, int numberOfOffers)
```

La méthode `getOffers` vous permet de demander des offres à partir du serveur d'exécution.

- **sessionID** — Chaîne identifiant la session en cours.
- **interactionPoint** — Chaîne identifiant le nom du point d'interaction référencé par cette méthode.

  Remarque : Ce nom doit correspondre exactement au nom du point d'interaction défini dans le canal interactif.
- **numberOfOffers** — Entier identifiant le nombre d'offres demandées.

La méthode `getOffers` attend le nombre de millisecondes défini dans la propriété `segmentationMaxWaitTimeInMS` afin de permettre à toute la resegmentation de se terminer avant de s'exécuter. Par conséquent, si vous appelez une méthode

postEvent qui déclenche une resegmentation ou appelez une méthode setAudience juste avant un appel getOffers, il peut y avoir un retard.

Valeur de retour

Le serveur d'exécution répond à getOffers à l'aide d'un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

Exemple

Cet exemple illustre une demande d'offre unique pour le point d'interaction Bannière de la page Présentation 1 et un moyen de traiter la réponse.

sessionId est la même chaîne que celle qui permet d'identifier la session d'exécution utilisée par l'appel startSession qui a démarré cette session.

```
String interactionPoint = "Bannière de la page Présentation 1";
int numberRequested=1;

/** Make the call */
response = api.getOffers(sessionId, interactionPoint, numberRequested);

/** Process the response appropriately */
 // check if response is successful or not
 if(response.getStatusCode() == Response.STATUS_SUCCESS)
 {
  System.out.println("getOffers call processed with no warnings or errors");

        /** Check to see if there are any offers */
        OfferList offerList=response.getOfferList();

        if(offerList.getRecommendedOffers() != null)
        {
            for(Offer offer : offerList.getRecommendedOffers())
            {
                // print offer
                System.out.println("Offer Name:"+offer.getOfferName());
            }
        }
        else // count on the default Offer String
        System.out.println("Default offer:"+offerList.getDefaultString());
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getOffers call processed with a warning");
    }
    else
    {
        System.out.println("getOffers call processed with an error");
    }
    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getOffers",
   response.getAdvisoryMessages());
```

getOffersForMultipleInteractionPoints

```
getOffersForMultipleInteractionPoints(String sessionID, String requestStr)
```

La méthode `getOffersForMultipleInteractionPoints` vous permet de demander des offres à partir du serveur d'exécution pour plusieurs points d'interaction avec dédoublonnage.

- **sessionID** — Chaîne identifiant la session en cours.
- **requestStr** — Chaîne fournissant un tableau d'objets `GetOfferRequest`.

  Chaque objet `GetOfferRequest` spécifique ce qui suit :

  – **ipName** — Nom du point d'interaction (IP) pour lequel l'objet demande des offres

  – **numberRequested** — Nombre d'offres uniques nécessaires pour le point d'interaction indiqué

  – **offerAttributes** — Configuration requise pour les attributs des offres distribuées à l'aide d'une instance `OfferAttributeRequirements`

  – **duplicationPolicy** — ID de stratégie de duplication pour les offres à distribuer

    Les règles de duplication déterminent si les offres en double seront renvoyées dans différents points d'interaction dans un seul appel de méthode. (*Dans* un point d'interaction individuel, les offres en double ne sont jamais renvoyées). Actuellement, deux règles de duplication sont prises en charge.

    - NO_DUPLICATION (valeur d'ID = 1). Aucune des offres incluses dans les instances précédentes de `GetOfferRequest` ne seront inclues dans cette instance de `GetOfferRequest` (c'est-à-dire que Interact appliquera le dédoublonnage).
    - ALLOW_DUPLICATION (valeur ID = 2). Toutes les offres répondant aux exigences indiquées dans cette instance de `GetOfferRequest` seront incluses. Les offres qui ont été inclus dans les précédentes instances de `GetOfferRequest` ne seront pas rapprochées.

  L'ordre des demandes dans le paramètre de tableau est également l'ordre de priorité lorsque des offres sont en cours de distribution.

  Par exemple, supposons que les points d'interaction dans la demande sont IP1, puis IP2, qu'aucune offre en double n'est autorisée (ID de règle de duplication = 1), et que chacun demande deux offres. Si Interact propose A, B et C à IP1 et A et D à IP2, la réponse contiendra les offres A et B pour IP1, et seulement l'offre D pour IP2.

  Notez également que lorsque l'ID de règle de dédoublonnage est 1, les offres distribuées via un point d'interaction ayant une priorité plus élevée ne seront pas distribuées via ce point d'interaction.

La méthode `getOffersForMultipleInteractionPoints` attend le nombre de millisecondes défini dans la propriété `segmentationMaxWaitTimeInMS` afin de permettre à toute la re-segmentation de se terminer avant de s'exécuter. Par conséquent, si vous appelez une méthode `postEvent` qui déclenche une resegmentation ou appelez une méthode `setAudience` juste avant un appel `getOffers`, il peut y avoir un retard.

Valeur de retour

Le serveur d'exécution répond à `getOffersForMultipleInteractionPoints` à l'aide d'un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages

- ApiVersion
- array of OfferList
- SessionID
- StatusCode

Exemple

```
InteractAPI api = InteractAPI.getInstance("url");
 String sessionId = "123";
 String requestForIP1 = "{IP1,5,1,(5,attr1=1|numeric;attr2=value2|string,
     (3,attr3=value3|string)(3,attr4=4|numeric))}";
 String requestForIP2 = "{IP2,3,2,(3,attr5=value5|string)}";
 String requestForIP3 = "{IP3,2,1}";
 String requestStr = requestForIP1 + requestForIP2 + requestForIP3;
 Response response = api.getOffersForMultipleInteractionPoints(sessionId,
          requestStr);

 if (response.getStatusCode() == Response.STATUS_SUCCESS) {
  // Check to see if there are any offers
  OfferList[] allOfferLists = response.getAllOfferLists();
  if (allOfferLists != null) {
   for (OfferList ol : allOfferLists) {
    System.out.println("The following offers are delivered for interaction
               point " + ol.getInteractionPointName() + ":");
    for (Offer o : ol.getRecommendedOffers()) {
     System.out.println(o.getOfferName());
    }
   }
  }
 }
 else {
  System.out.println("getOffersForMultipleInteractionPoints() method calls
          returns an error with code: " + response.getStatusCode());
 }
```

Notez que la syntaxe de `requestStr` est la suivante :

```
requests_for_IP[<requests_for_IP]
```

où

```
<requests_for_IP> = {ip_name,number_requested_for_this_ip,
     dupe_policy[,child_requirements]]}
attribute_requirements = (number_requested_for_these_attribute_requirements
     [,attribute_requirement[;individual_attribute_requirement])
     [,(attribute_requirements))
individual_attribute_requirement = attribute_name=attribute_value | attribute_type
```

Dans l'exemple ci-dessus, `requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))})` signifie que pour le point d'interaction IP1, 5 offres distinctes au maximum doivent être distribuées. Ces offres ne peuvent en outre pas être renvoyées pour les autres points d'interaction pendant le même appel de méthode. Chacune de ces 5 offres doit avoir un attribut numérique nommé `attr1` qui doit avoir la valeur 1, et doit avoir un attribut de chaîne nommé `attr2` qui doit avoir la valeur *value2*. Sur ces 5 offres, un maximum de 3 doit avoir un attribut de chaîne appelé `attr3` ayant la valeur *value3*, un maximum de 3 doit avoir un attribut and numérique appelé `attr4` ayant la valeur *4*.

Les types d'attribut autorisés sont numérique, chaîne, date/heure (numeric, string et datetime) et le format d'une valeur d'attribut date/heure doit être `MM/jj/aaaa HH:mm:ss`. Pour extraire les offres renvoyées, utilisez la méthode

Response.getAllOfferLists(). Pour vous aider à comprendre la syntaxe, l'exemple dans setGetOfferRequests génère la même instance de GetOfferRequests, tout en utilisant des objets Java, ce qui est recommandé.

getProfile

getProfile(String *sessionID*)

La méthode getProfile vous permet d'extraire le profil et les informations temporaires sur le visiteur consultant le point de contact.

- **sessionID** — Chaîne identifiant l'ID session.

Valeur de retour

Le serveur d'exécution répond à getProfile avec un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

Exemple

Voici un exemple d'utilisation de getProfile et un moyen de traiter la réponse.

sessionId est la même chaîne permettant d'identifier la session utilisée par l'appel startSession qui a démarré cette session.

```
response = api.getProfile(sessionId);
/**  Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("getProfile call processed with no warnings or errors");
        // Print the profile - it's just an array of NameValuePair objects
        for(NameValuePair nvp : response.getProfileRecord())
        {
            System.out.println("Name:"+nvp.getName());
            if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
            {
                System.out.println("Value:"+nvp.getValueAsDate());
            }
            else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
            {
                System.out.println("Value:"+nvp.getValueAsNumeric());
            }
            else
            {
                System.out.println("Value:"+nvp.getValueAsString());
            }
        }
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getProfile call processed with a warning");
    }
    else
    {
        System.out.println("getProfile call processed with an error");
    }
```

```
    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getProfile",
   response.getAdvisoryMessages());
```

getVersion

```
getVersion()
```

La méthode `getVersion` renvoie la version de l'implémentation actuelle du serveur d'exécution Interact.

La meilleure pratique consiste à utiliser cette méthode lorsque vous initialisez le point de contact avec l'API Interact.

Valeur de retour

Le serveur d'exécution répond à `getVersion` à l'aide d'un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages
- ApiVersion
- StatusCode

Exemple

Cet exemple présente un moyen simple d'appeler `getVersion` et de traiter les résultats.

```
response = api.getVersion();
/** Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("getVersion call processed with no warnings or errors");
        System.out.println("API Version:" + response.getApiVersion(););
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getVersion call processed with a warning");
    }
    else
    {
        System.out.println("getVersion call processed with an error");
    }

    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getVersion",
   response.getAdvisoryMessages());
```

postEvent

```
postEvent(String sessionID, String eventName, NameValuePairImpl[] eventParameters)
```

La méthode `postEvent` vous permet d'exécuter n'importe quel événement défini dans le canal interactif.

- **sessionID** — Chaîne identifiant l'ID session.
- **eventName** — Chaîne identifiant le nom de l'événement.

  Remarque : Ce nom doit correspondre exactement au nom de l'événement défini dans le canal interactif. Ce nom est insensible à la casse.

- **eventParameters** — Objets NameValuePairImpl identifiant tous les paramètres à passer avec l'événement. Ces valeurs sont stockées dans les données de session.

  Si cet événement déclenche la resegmentation, vous devez veiller à ce que toutes les données requises par les diagrammes interactifs soient disponibles dans les données de session. Si ces valeurs n'ont pas été renseignées par des actions précédentes (par exemple, depuis `startSession` ou `setAudience`, ou en chargeant la table de profil), vous devez inclure uneventParameter pour chaque valeur manquante. Par exemple, si vous avez configuré toutes les tables de profil à charger dans la mémoire, vous devez inclure un NameValuePair pour les données temporelles requises pour les diagrammes interactifs.

  Si vous utilisez plusieurs référentiels, vous avez probablement différents ensembles de eventParameters pour chaque référentiel. Vous devez inclure une logique afin d'avoir la certitude de sélectionner l'ensemble correct de paramètres du référentiel.

  Important : Si cet événement se connecte à l'historique des réponses, vous devez transmettre le code de traitement de l'offre. Vous devez définir le nom de NameValuePair comme "`UACIOfferTrackingCode`".

  Vous ne pouvez transmettre qu'un seul code de traitement par événement. Si vous ne transmettez pas le code de traitement du contact d'une offre, Interact journalise un contact d'offre pour chaque offre dans la dernière liste des offres recommandées. Si vous ne transmettez pas le code de traitement d'une réponse, Interact renvoie une erreur.
- Il existe plusieurs autres paramètres réservés utilisés avec `postEvent` et d'autres méthodes, qui sont décrits ultérieurement dans cette section.

Toute demande de resegmentation ou d'écriture dans l'historique des contacts ou des réponses n'attend pas de réponse.

Sauf si vous l'avez indiqué avec le paramètre `UACIExecuteFlowchartByName`, la segmentation exécute tous les diagrammes interactifs associés à ce canal interactif pour le référentiel actuel. La méthode `getOffers` attend la fin de la resegmentation avant de s'exécuter. Par conséquent, si vous appelez une méthode `postEvent` qui déclenche une resegmentation juste avant un appel `getOffers`, il peut y avoir un retard.

Valeur de retour

Le serveur d'exécution répond à `postEvent` avec un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Exemple

L'exemple `postEvent` ci-dessous illustre l'envoi de nouveaux paramètres pour un événement qui déclenche la resegmentation, et indique un moyen de traiter la réponse.

`sessionId` est la même chaîne permettant d'identifier la session utilisée par l'appel `startSession` qui a démarré cette session.

```
String eventName = "SearchExecution";

NameValuePair parmB1 = new NameValuePairImpl();
parmB1.setName("SearchString");
parmB1.setValueAsString("mortgage");
parmB1.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair parmB2 = new NameValuePairImpl();
parmB2.setName("TimeStamp");
parmB2.setValueAsDate(new Date());
parmB2.setValueDataType(NameValuePair.DATA_TYPE_DATETIME);

NameValuePair parmB3 = new NameValuePairImpl();
parmB3.setName("Browser");
parmB3.setValueAsString("IE6");
parmB3.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair parmB4 = new NameValuePairImpl();
parmB4.setName("FlashEnabled");
parmB4.setValueAsNumeric(1.0);
parmB4.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

NameValuePair parmB5 = new NameValuePairImpl();
parmB5.setName("TxAcctValueChange");
parmB5.setValueAsNumeric(0.0);
parmB5.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

NameValuePair parmB6 = new NameValuePairImpl();
parmB6.setName("PageTopic");
parmB6.setValueAsString("");
parmB6.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair[] postEventParameters = { parmB1,
 parmB2,
 parmB3,
 parmB4,
 parmB5,
 parmB6
};

/** Make the call */
response = api.postEvent(sessionId, eventName, postEventParameters);

/** Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("postEvent call processed with no warnings or errors");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("postEvent call processed with a warning");
    }
    else
    {
        System.out.println("postEvent call processed with an error");
    }

    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("postEvent",
   response.getAdvisoryMessages());
```

setAudience

```
setAudience(String sessionID, NameValuePairImpl[] audienceID,
     String audienceLevel, NameValuePairImpl[] parameters)
```

La méthode setAudience vous permet de définir l'ID de référentiel et le référentiel d'un visiteur.

- **sessionID** — Chaîne identifiant l'ID session.
- **audienceID** — Tableau d'objets NameValuePairImpl définissant l'ID de référentiel.
- **audienceLevel** — Chaîne définissant le référentiel.
- **parameters** — Objets NameValuePairImpl identifiant tous les paramètres à passer avec setAudience. Ces valeurs sont stockées dans les données de session et peuvent être utilisées pour la segmentation.

  Vous devez avoir une valeur dans chaque colonne de votre profil. Il s'agit d'un sur-ensemble de toutes les colonnes de toutes les tables définies pour le canal interactif et de toutes les données en temps réel. Si vous avez déjà rempli toutes les données de session avec startSession ou postEvent, vous n'avez pas besoin d'envoyer de nouveaux paramètres.

La méthode setAudience déclenche une resegmentation. La méthode getOffers attend la fin de la resegmentation avant de s'exécuter. Par conséquent, si vous appelez une méthode setAudience juste avant un appel getOffers, il peut y avoir un retard.

La méthode setAudience charge également les données de profil de l'ID de référentiel. Vous pouvez utiliser la méthode setAudience pour forcer un rechargement des mêmes données de profil chargées par la méthode startSession.

Valeur de retour

Le serveur d'exécution répond à setAudience à l'aide d'un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Exemple

Dans cet exemple, le référentiel reste le même, mais l'ID change, comme si un utilisateur anonyme se connectait et était identifié.

sessionId et audienceLevel sont les mêmes chaînes permettant d'identifier la session et le référentiel utilisés par l'appel startSession qui a démarré cette session.

```
NameValuePair custId2 = new NameValuePairImpl();
custId2.setName("CustomerId");
custId2.setValueAsNumeric(123.0);
custId2.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

NameValuePair[] newAudienceId = { custId2 };

/** Parameters can be passed in as well.  For this example, there are no parameters,
* therefore pass in null */
NameValuePair[] noParameters=null;

/** Make the call  */
response = api.setAudience(sessionId, newAudienceId, audienceLevel, noParameters);

 /** Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("setAudience call processed with no warnings or errors");
```

```
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("setAudience call processed with a warning");
    }
    else
    {
        System.out.println("setAudience call processed with an error");
    }

    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setAudience",
   response.getAdvisoryMessages());
```

setDebug

setDebug(String *sessionID*, boolean *debug*)

La méthode setDebug vous permet de définir le niveau de prolixité de la journalisation pour tous les chemins de code de la session.

- **sessionID** — Chaîne identifiant l'ID session.
- **debug** — Valeur booléenne qui active ou désactive les informations de débogage. Les valeurs admises sont true ou false. Si elle est true, Interact journalise les informations de débogage dans les journaux du serveur d'exécution.

Valeur de retour

Le serveur d'exécution répond à setDebug avec un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Exemple

L'exemple suivant illustre la modification du niveau de débogage de la session.

sessionId est la même chaîne permettant d'identifier la session utilisée par l'appel startSession qui a démarré cette session.

```
boolean newDebugFlag=false;
/** make the call */
response = api.setDebug(sessionId, newDebugFlag);

/** Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("setDebug call processed with no warnings or errors");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("setDebug call processed with a warning");
    }
    else
    {
        System.out.println("setDebug call processed with an error");
    }
```

```
// For any non-successes, there should be advisory messages explaining why
if(response.getStatusCode() != Response.STATUS_SUCCESS)
    printDetailMessageOfWarningOrError("setDebug",
response.getAdvisoryMessages());
```

startSession

```
startSession(String sessionID,
   boolean relyOnExistingSession,
   boolean debug,
   String interactiveChannel,
   NameValuePairImpl[] audienceID,
   String audienceLevel,
   NameValuePairImpl[] parameters)
```

La méthode `startSession` crée et définit une session d'exécution. `startSession` peut déclencher cinq actions au maximum :

- Créer une session d'exécution.
- Charger les données de profil du visiteur correspondant au référentiel en cours dans la session d'exécution, notamment les tables de dimension marquées en vue d'un chargement dans le mappage de table défini pour le canal interactif.
- Déclencher la segmentation, en exécutant tous les diagrammes interactifs correspondant au référentiel en cours.
- Charger les données de suppression de l'offre dans la session, si la propriété `enableOfferSuppressionLookup` est définie sur true.
- Charger les données de substitution de score dans la session, si la propriété `enableScoreOverrideLookup` est définie sur true.

La méthode `startSession` nécessite les paramètres suivants :

- **sessionID** — chaîne identifiant l'ID session. Vous devez définir l'ID session. Par exemple, vous pouvez utiliser une combinaison de l'ID client et de l'horodatage.

  Pour définir ce qui constitue une session d'exécution, un ID session doit être indiqué. Cette valeur est gérée par le client. Tous les appels de méthode au même ID de session doivent être synchronisés par le client. Le comportement des appels API simultanés ayant le même ID de session n'est pas défini.
- **relyOnExistingSession** — valeur booléenne qui définit si cette session utilise une session nouvelle ou existante. Les valeurs admises sont `true` ou `false`. Si elle est `true`, vous devez fournir un ID de session existant avec la méthode `startSession`. Si elle est `false`, vous devez fournir un nouvel ID de session.

  Si vous définissez `relyOnExistingSession` sur `true` et s'il existe une session, l'environnement d'exécution utilise les données de la session existante et ne recharge pas les données ou la segmentation de déclenchement. Si la session n'existe pas, l'environnement d'exécution crée une nouvelle session, y compris les données de chargement et la segmentation de déclenchement. Le fait de définir `relyOnExistingSession` sur true et de l'utiliser avec tous les appels `startSession` est utile si votre point de contact a une durée de session plus longue que celle de la session d'exécution. Par exemple, une session de site Web est active pendant 2 heures, mais la session d'exécution est uniquement active pendant 20 minutes.

  Si vous appelez `startSession` deux fois avec le même ID session, tous les données de session du premier appel `startSession` sont perdues si `relyOnExistingSession` a la valeur false.
- **debug** — valeur booléenne qui active ou désactive les informations de débogage. Les valeurs admises sont `true` ou `false`. Si elle est true, Interact journalise les informations de débogage dans les journaux du serveur

d'exécution. L'indicateur de débogage est défini individuellement pour chaque session. Par conséquent, vous pouvez effectuer le suivi des des données de débogage pour une session individuelle.

- **interactiveChannel** — chaîne définissant le nom du canal interactif auquel cette session fait référence. Ce nom doit correspondre exactement au nom du canal interactif défini dans Campaign.
- **audienceID** — tableau d'objets NameValuePairImpl, dans lequel les noms doivent correspondre aux noms de colonne physique de toute table contenant l'ID de référentiel.
- **audienceLevel** — Chaîne définissant le référentiel.
- **parameters** — Objets NameValuePairImpl identifiant tous les paramètres à passer avec `startSession`. Ces valeurs sont stockées dans les données de session et peuvent être utilisées pour la segmentation.

  Si vous avez plusieurs diagrammes interactifs pour le même référentiel, vous devez inclure un sur-ensemble de toutes les colonnes dans toutes les tables. Si vous configurez l'exécution de façon à ce qu'elle change la table de profil, et si la table de profil contient toutes les colonnes requises, il n'est pas nécessaire de passer de paramètres, sauf si vous souhaitez écraser les données dans la table de profil. Si votre table de profil contient un sous-ensemble des colonnes requises, vous devez inclure les colonnes manquantes en tant que paramètres.

Si le `audienceID` ou `audienceLevel` ne sont pas valides et si `relyOnExistingSession` a la valeur false, l'appel `startSession` échoue. Si le `interactiveChannel` n'est pas valide, `startSession` échoue, que la valeur de `relyOnExistingSession` soit true ou false.

Si `relyOnExistingSession` a la valeur true, et si vous effectuez un deuxième appel `startSession` avec le même `sessionID`, alors que la première session a expiré, Interact crée une nouvelle session.

Si `relyOnExistingSession` a la valeur et si vous effectuez un deuxième appel `startSession` avec le même `sessionID`, mais avec un autre `audienceID` ou `audienceLevel`, le serveur d'exécution change le référentiel de la session existante.

Si `relyOnExistingSession` a la valeur true, et si vous effectuez un deuxième appel `startSession` avec le même `sessionID` mais un autre `interactiveChannel`, le serveur d'exécution crée une nouvelle session.

Valeur de retour

Le serveur d'exécution répond à `startSession` avec un objet de réponse dans lequel les attributs suivants sont renseignés :

- AdvisoryMessages (si StatusCode n'est pas égal à 0)
- ApiVersion
- SessionID
- StatusCode

Exemple

L'exemple suivant montre une façon d'appeler `startSession`.

```
String sessionId="MySessionID-123";
String audienceLevel="Customer";
NameValuePair custId = new NameValuePairImpl();
custId.setName("CustomerId");
```

```
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
boolean relyOnExistingSession=false;
boolean initialDebugFlag=true;
String interactiveChannel="Accounts Website";
NameValuePair parm1 = new NameValuePairImpl();
parm1.setName("SearchString");
parm1.setValueAsString("");
parm1.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair parm2 = new NameValuePairImpl();
parm2.setName("TimeStamp");
parm2.setValueAsDate(new Date());
parm2.setValueDataType(NameValuePair.DATA_TYPE_DATETIME);

NameValuePair parm3 = new NameValuePairImpl();
parm3.setName("Browser");
parm3.setValueAsString("IE6");
parm3.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair parm4 = new NameValuePairImpl();
parm4.setName("FlashEnabled");
parm4.setValueAsNumeric(1.0);
parm4.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

NameValuePair parm5 = new NameValuePairImpl();
parm5.setName("TxAcctValueChange");
parm5.setValueAsNumeric(0.0);
parm5.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

NameValuePair parm6 = new NameValuePairImpl();
parm6.setName("PageTopic");
parm6.setValueAsString("");
parm6.setValueDataType(NameValuePair.DATA_TYPE_STRING);

/** Specifying the parameters (optional) */
NameValuePair[] initialParameters = { parm1,
 parm2,
 parm3,
 parm4,
 parm5,
 parm6
 };

/** Make the call */
response = api.startSession(sessionId, relyOnExistingSession, initialDebugFlag,
 interactiveChannel, initialAudienceId, audienceLevel, initialParameters);

/** Process the response appropriately */
processStartSessionResponse(response);
```

processStartSessionResponse is a method which handles the response object returned by startSession.

```
public static void processStartSessionResponse(Response response)
{
  // check if response is successful or not
  if(response.getStatusCode() == Response.STATUS_SUCCESS)
 {
  System.out.println("startSession call processed with no warnings or errors");
 }
  else if(response.getStatusCode() == Response.STATUS_WARNING)
 {
  System.out.println("startSession call processed with a warning");
 }
  else
```

```
 {
  System.out.println("startSession call processed with an error");
 }

 // For any non-successes, there should be advisory messages explaining why
 if(response.getStatusCode() != Response.STATUS_SUCCESS)
  printDetailMessageOfWarningOrError("StartSession",
   response.getAdvisoryMessages());
}
```

Paramètres réservés

Il existe plusieurs paramètres réservés qui sont utilisés avec l'API Interact. Certains sont requis pour le serveur d'exécution, d'autres peuvent être utilisés pour des fonctions supplémentaires.

Fonctions postEvent

Fonction	Paramètre	Description
Journaliser dans une table personnalisée	UACICustomLoggerTableName	Nom d'une table dans la source de données des tables d'exécution. Si vous fournissez ce paramètre avec un nom de table valide, l'environnement d'exécution écrit toutes les données de session dans la table sélectionnée. Tous les noms de colonne de la table correspondant aux données de session NameValuePair sont renseignés. L'environnement d'exécution renseigne n'importe quelle colonne qui ne correspond pas à une paire nom de session-valeur avec une valeur NULL. Vous pouvez gérer le processus qui écrit dans la base de données avec les propriétés de configuration customLogger.
Types de réponse multiples	UACILogToLearning	Nombre entier ayant la valeur 1 ou 0. 1 indique que l'environnement d'exécution doit journaliser l'événement comme une acceptation pour l'apprentissage. 0 indique que l'environnement d'exécution ne doit pas journaliser l'événement pour l'apprentissage. Ce paramètre vous permet de créer plusieurs méthodes postEvent qui journalisent différents types de réponse sans influence sur l'apprentissage. Vous n'avez pas besoin de définir ce paramètre pour les événements définis pour journaliser un contact, une acceptation ou un refus. Vous devez utiliser ce paramètre avec UACIResponseTypeCode. Si vous ne définissez pas UACILOGTOLEARNING, l'environnement d'exécution adopte la valeur par défaut 0 (sauf si l'événement déclenche un contact de journal, une acceptation ou un refus).
	UACIResponseTypeCode	Valeur représentant un code de type de réponse. La valeur doit être une entrée valide de la table UA_UsrResponseType

Fonction	Paramètre	Description
Suivi des réponses	UACIOfferTrackingCode	Code de traitement de l'offre. Vous devez définir ce paramètre si l'événement est journalisé dans l'historique des contacts ou des réponses. Vous ne pouvez transmettre qu'un seul code de traitement par événement. Si vous ne transmettez pas le code de traitement du contact d'une offre, l'environnement d'exécution journalise un contact d'offre pour chaque offre dans la dernière liste des offres recommandées. Si vous ne transmettez pas le code de traitement d'une réponse, l'environnement d'exécution renvoie une erreur. Si vous configurez le suivi de réponse de session croisée, vous pouvez utiliser le paramètre UACIOfferTrackingcodeType pour définir le type de code de suivi que vous utilisez s'il ne s'agit pas d'un code de traitement.
Suivi des réponses intersessions	UACIOfferTrackingCodeType	Nombre qui définit le type de code de suivi. 1 est le code de traitement par défaut, et 2 est le code de l'offre. Tous les codes doivent être des entrées valides dans la table UACI_TrackingType. Vous pouvez ajouter d'autres codes personnalisés dans cette table.
Exécution de diagramme spécifique	UACIExecuteFlowchartByName	Si vous définissez ce paramètre pour une méthode déclenchant la segmentation (startSession, setAudience, ou un postEvent qui déclenche la resegmentation), au lieu d'exécuter tous les diagrammes du référentiel en cours, Interact exécute uniquement les diagrammes nommés. Vous pouvez fournir une liste des diagrammes séparés par un caractère de barre verticale (|).

paramètres réservés de l'environnement d'exécution

Les paramètres réservés suivants sont utilisés par l'environnement d'exécution. N'utilisez pas ces noms pour vos paramètres d'événement.

- UACIEventID
- UACIEventName
- UACIInteractiveChannelID
- UACIInteractiveChannelName
- UACIInteractionPointID
- UACIInteractionPointName
- UACISessionID

A propos de la classe AdvisoryMessage

La classe advisoryMessage contient des méthodes qui définissent l'objet de message de recommandation. L'objet de message de recommandation est contenu dans l'objet de réponse. Chaque méthode d'InteractAPI renvoie un objet de réponse, (sauf la méthode executeBatch , qui renvoie un objet batchResponse.) S'il y a une

erreur ou un avertissement, le serveur Interact renseigne l'objet de message de recommandation. L'objet de message de recommandation contient les attributs suivants :

- **DetailMessage** — Description prolixe du message de recommandation. Cet attribut peut ne pas être disponible pour tous les messages de recommandation. Si elle est disponible, DetailMessage ne peut pas être localisé.
- **Message** — Description brève du message de recommandation.
- **MessageCode** — Numéro de code du message de recommandation.
- **StatusLevel** — Numéro de code de la gravité du message de recommandation.

Vous extrayez les objets advisoryMessage à l'aide de la méthode `getAdvisoryMessages`.

getDetailMessage

```
getDetailMessage()
```

La méthode `getDetailMessage` renvoie la description détaillée et prolixe d'un objet de Message de recommandation.

Tous les messages ne s'accompagnent pas d'un message détaillé.

Valeur de retour

L'objet Message de recommandation renvoie une chaîne.

Exemple

```
// For any non-successes, there should be advisory messages explaining why
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
 for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
  System.out.println(msg.getMessage());
  // Some advisory messages may have additional detail:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessage

```
getMessage()
```

La méthode `getMessage` renvoie la description brève d'un objet de Message de recommandation.

Valeur de retour

L'objet Message de recommandation renvoie une chaîne.

Exemple

La méthode suivante imprime le code de message le message et le message détaillé d'un objet AdvisoryMessage.

```
// For any non-successes, there should be advisory messages explaining why
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
 for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
```

```
  System.out.println(msg.getMessage());
  // Some advisory messages may have additional detail:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessageCode

```
getMessageCode()
```

La méthode `getMessageCode` renvoie le code d'erreur interne associé à un objet Advisory Message (message de recommandation) si le niveau de statut est 2 (STATUS_LEVEL_ERROR).

Valeur de retour

L'objet AdvisoryMessage renvoie un entier.

Exemple

La méthode suivante imprime le code de message d'un objet AdvisoryMessage.

```
public static void printMessageCodeOfWarningOrError(String command,AdvisoryMessage[] messages)
    {
        System.out.println("Calling "+command);
        for(AdvisoryMessage msg : messages)
        {
            System.out.println(msg.getMessageCode());
```

getStatusLevel

```
getStatusLevel()
```

La méthode `getStatusLevel` renvoie le niveau de statut d'un objet de Message de recommandation.

Valeur de retour

L'objet de Message de recommandation renvoie un entier.

- 0 - STATUS_LEVEL_SUCCESS — La méthode appelée s'est terminée sans erreurs.
- 1 - STATUS_LEVEL_WARNING — La méthode appelée s'est terminée avec au moins un avertissement (mais sans erreurs).
- 2 - STATUS_LEVEL_ERROR — La méthode appelée ne s'est pas terminée correctement et comporte au moins une erreur.

Exemple

La méthode suivante imprime le niveau de statut d'un objet AdvisoryMessage.

```
public static void printMessageCodeOfWarningOrError(String command,AdvisoryMessage[] messages)
    {
        System.out.println("Calling "+command);
        for(AdvisoryMessage msg : messages)
        {
            System.out.println(msg.getStatusLevel());
```

A propos de la classe AdvisoryMessageCode

La classe `advisoryMessageCode` contient des méthodes qui définissent les codes du message de recommandation. Vous extrayez les codes du message de recommandation avec la méthode `getMessageCode`.

Codes des messages de recommandation

Code	Description
1	INVALID_SESSION_ID - L'ID de session ne fait pas référence à une session valide
2	ERROR_TRYING_TO_ABORT_SEGMENTATION - Une erreur s'est produite pendant la tentative d'abandon de la segmentation pendant une `endSession`
3	INVALID_INTERACTIVE_CHANNEL - L'argument transmis pour le canal interactif ne fait pas référence à un canal interactif valide
4	INVALID_EVENT_NAME - L'argument transmis pour l'événement ne fait pas référence à un événement valide pour le canal interactif en cours
5	INVALID_INTERACTION_POINT - L'argument transmis pour le point d' interaction ne fait pas référence à un point d'interaction valide pour le canal interactif en cours
6	ERROR_WHILE_MAKING_INITIAL_SEGMENTATION_REQUEST - Une erreur s'est produite lors de la soumission d'une requête de segmentation
7	SEGMENTATION_RUN_FAILED - La segmentation a été exécuté en partie, mais a terminé sur une erreur
8	PROFILE_LOAD_FAILED - La tentative de chargement du profil ou des tables de dimension a échoué
9	OFFER_SUPPRESSION_LOAD_FAILED - La tentative de chargement de la table de suppression de l'offre a échoué
10	COMMAND_METHOD_UNRECOGNIZED - La méthode de commande indiquée pour une commande dans un `executeBatch` n'est pas valide
11	ERROR_TRYING_TO_POST_EVENT_PARAMETERS - Une erreur s'est produite lors de l'envoi de paramètres d'événement
12	LOG_SYSTEM_EVENT_EXCEPTION - Une exception s'est produite lors de la tentative de soumission des événements système (Terminer la session, Obtenir l'offre, Obtenir le profil, Définir le référentiel, Définir le débogage ou Démarrer session) pour la journalisation
13	LOG_USER_EVENT_EXCEPTION - une exception s'est produite lors de la tentative de soumission d'un événement utilisateur pour la journalisation
14	ERROR_TRYING_TO_LOOK_UP_EVENT - Une erreur s'est produite lors de la tentative de recherche du nom d'événement
15	ERROR_TRYING_TO_LOOK_UP_INTERACTIVE_CHANNEL - Une erreur s'est produite lors de la tentative de recherche du nom de canal interactif
16	ERROR_TRYING_TO_LOOK_UP_INTERACTION_POINT - Une erreur s'est produite lors de la tentative de recherche du nom de point d'interaction
17	RUNTIME_EXCEPTION_ENCOUNTERED - Une exception d'exécution inattendue s'est produite
18	ERROR_TRYING_TO_EXECUTE_ASSOCIATED_ACTION - Erreur lors de la tentative d'exécution de l'action associée (Déclencher la resegmentation, Journaliser le contact de l'offre, Journaliser l'acceptation de l'offre, ou Journaliser le refus de l'offre)
19	ERROR_TRYING_RUN_FLOWCHART - Erreur lors de la tentative d'exécution du diagramme
20	FLOWCHART_FAILED - Echec du diagramme
21	FLOWCHART_ABORTED - Abandon du diagramme
22	FLOWCHART_NEVER_RUN - Le diagramme ne s'est jamais exécuté
23	FLOWCHART_STILL_RUNNING - Le diagramme s'exécute toujours

Code	Description
24	ERROR_WHILE_READING_PARAMETERS - Erreur lors de la lecture des paramètres
25	ERROR_WHILE_LOADING_RECOMMENDED_OFFERS - Erreur lors du chargement des offres recommandées
26	ERROR_WHILE_LOGGING_DEFAULT_TEXT_STATISTICS - Erreur lors de la journalisation des statistiques de texte par défaut (nombre d'affichages de la chaîne par défaut du point d'interaction)
27	SCORE_OVERRIDE_LOAD_FAILED - Echec du chargement de la table de substitution de score
28	NULL_AUDIENCE_ID - L'ID référentiel est vide
29	UNRECOGNIZED_AUDIENCE_LEVEL - Niveau de référentiel non reconnu
30	MISSING_AUDIENCE_FIELD - Zone de référentiel manquante
31	INVALID_AUDIENCE_FIELD_TYPE - Type de zone de référentiel non valide
32	UNSUPPORTED_AUDIENCE_FIELD_TYPE - Type de zone de référentiel non pris en charge
33	TIMEOUT_REACHED_ON_GET_OFFERS_CALL - L'appel getOffers a expiré sans renvoyer d'offres
34	INTERACT_INITIALIZATION_NOT_COMPLETED_SUCCESSFULLY - L'initialisation de l'exécution ne s'est pas terminée correctement
35	SESSION_ID_UNDEFINED - ID session non défini
36	INVALID_NUMBER_OF_OFFERS_REQUESTED - Nombre non valide d'offres demandées
37	NO_SESSION_EXIST_BUT_WILL_CREATE_NEW_ONE
38	AUDIENCE_ID_NOT_FOUND_IN_PROFILE_TABLE
39	LOG_CUSTOM_LOGGER_EVENT_EXCEPTION - Une exception s'est produite lors de la tentative de soumission de l'événement des données de journalisation personnalisées
40	SPECIFIED_FLOWCHART_FOR_EXECUTION_DOES_NOT_EXIST
41	AUDIENCE_NOT_DEFINED_IN_CONFIGURATION

A propos de la classe BatchResponse

La classe `BatchResponse` contient des méthodes qui définissent les résultats de la méthode `executeBatch`. L'objet Batch Response contient les attributs suivants :

- **BatchStatusCode** — Code de statut plus haute valeur pour toutes les réponses demandées par la méthode `executeBatch`.
- **Responses** - Un tableau des objets de réponse demandés par la méthode `executeBatch`.

getBatchStatusCode

```
getBatchStatusCode()
```

La méthode `getBatchStatusCode` renvoie le code de statut plus élevé à partir du tableau de commandes exécutées par la méthode `executeBatch`.

Valeur de retour

La méthode getBatchStatusCode renvoie un entier.

- 0 - STATUS_SUCCESS — La méthode appelée s'est terminée sans erreurs.
- 1 - STATUS_WARNING — La méthode appelée s'est terminée avec au moins un avertissement (mais sans erreurs).
- 2 - STATUS_ERROR — La méthode appelée ne s'est pas terminée correctement et comporte au moins une erreur.

Exemple

L'exemple de code suivant donne un exemple de la manière d'extraire BatchStatusCode.

```
// Top level status code is a short cut to determine if there are any
// non-successes in the array of Response objects
if(batchResponse.getBatchStatusCode() == Response.STATUS_SUCCESS)
{
  System.out.println("ExecuteBatch ran perfectly!");
}
else if(batchResponse.getBatchStatusCode() == Response.STATUS_WARNING)
{
  System.out.println("ExecuteBatch call processed with at least one warning");
}
else
{
  System.out.println("ExecuteBatch call processed with at least one error");
}

// Iterate through the array, and print out the message for any non-successes
for(Response response : batchResponse.getResponses())
{
  if(response.getStatusCode()!=Response.STATUS_SUCCESS)
  {
      printDetailMessageOfWarningOrError("executeBatchCommand",
   response.getAdvisoryMessages());
  }
}
```

getResponses

```
getResponses()
```

La méthode getResponses renvoie le tableau d'objets de réponse correspondant au tableau de commandes exécutées par la méthode executeBatch.

Valeur de retour

La méthode getResponses renvoie un tableau d'objets Response.

Exemple

L'exemple suivant sélectionne toutes les réponses et imprime tous les messages de recommandation si la commande n'a pas abouti.

```
for(Response response : batchResponse.getResponses())
{
    if(response.getStatusCode()!=Response.STATUS_SUCCESS)
    {
```

```
        printDetailMessageOfWarningOrError("executeBatchCommand",
   response.getAdvisoryMessages());
    }
}
```

A propos de l'interface de commande

La méthode `executeBatch` vous demande de transmettre un tableau d'objets qui implémentent l'interface de commande. Vous devez utiliser l'implémentation par défaut, `CommandImpl` pour transmettre les objets Commande.

Le tableau suivant répertorie la commande, la méthode de la classe InteractAPI que la commande représente, et les méthodes de l'interface de commande vous devez utiliser pour chaque commande. Vous n'avez pas besoin d'inclure un ID de session car la méthode `executeBatch` inclut déjà l'ID de session.

Commande	Méthode API Interact	Méthodes de l'interface de commande
COMMAND_ENDSESSION	endSession	Aucune.
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	Aucune.
COMMAND_GETVERSION	getVersion	Aucune.
COMMAND_POSTEVENT	postEvent	• setEvent • setEventParameters
COMMAND_SETAUDIENCE	setAudience	• setAudienceID • setAudienceLevel • setEventParameters
COMMAND_SETDEBUG	setDebug	setDebug
COMMAND_STARTSESSION	startSession	• setAudienceID • setAudienceLevel • setDebug • setEventParameters • setInteractiveChannel • setRelyOnExistingSession

setAudienceID

`setAudienceID(`*`audienceID`*`)`

La méthode `setAudienceID` définit AudienceID pour les commandes `setAudience` et `startSession`.

- **audienceID** — Tableau d'objets NameValuePair qui définissent AudienceID.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode executeBatch appelant startSession et setAudience.

```
NameValuePair custId = new NameValuePairImpl();
custId.setName("CustomerId");
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceID(newAudienceId);
. . .
/** Build command array */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Make the call */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Process the response appropriately */
    processExecuteBatchResponse(batchResponse);
```

setAudienceLevel

setAudienceLevel(*audienceLevel*)

La méthode setAudienceLevel définit le Référentiel pour les commandes setAudience et startSession.

- *audienceLevel* — Chaîne définissant le Référentiel.

  Important : Le nom de *audienceLevel* doit correspondre exactement au référentiel défini dans Campaign. Ce nom est sensible à la casse.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode executeBatch appelant startSession et setAudience.

```
String audienceLevel="Customer";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceLevel(audienceLevel);
. . .
/** Build command array */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
```

```
    };
/** Make the call */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Process the response appropriately */
    processExecuteBatchResponse(batchResponse);
```

setDebug

`setDebug(debug)`

La méthode `setDebug` définit le niveau de débogage pour la commande `startSession`. Si sa valeur est`true`, le serveur d'exécution journalise les informations de débogage dans le journal du serveur d'exécution. Si elle est `false`, le serveur d'exécution ne journalise pas les informations de débogage. L'indicateur de débogage est défini individuellement pour chaque session. Par conséquent, vous pouvez effectuer le suivi des données de débogage pour une session d'exécution individuelle.

- **debug** — Valeur booléenne (`true` ou `false`).

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode `executeBatch` appelant `startSession` et `setDebug`.

```
boolean initialDebugFlag=true;
boolean newDebugFlag=false;
. . .
/* build the startSession command */
Command startSessionCommand = new CommandImpl();
startSessionCommand.setDebug(initialDebugFlag);
. . .

/* build the setDebug command */
Command setDebugCommand = new CommandImpl();
setDebugCommand.setMethodIdentifier(Command.COMMAND_SETDEBUG);
setDebugCommand.setDebug(newDebugFlag);

/** Build command array */
Command[] commands =
    {
        startSessionCommand,
        setDebugCommand,
    };
/** Make the call */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Process the response appropriately */
    processExecuteBatchResponse(batchResponse);
```

setEvent

`setEvent(event)`

La méthode `setEvent` définit le nom de l'événement utilisé par la commande `postEvent`.

- **event** — Chaîne contenant le nom de l'événement.

Important : Le nom de *event* doit correspondre exactement au nom de l'événement défini dans le canal interactif. Ce nom est sensible à la casse.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode `executeBatch` appelant `postEvent`.

```
String eventName = "SearchExecution";

Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEventParameters(postEventParameters);
postEventCommand.setEvent(eventName);
```

setEventParameters

`setEventParameters(`*`eventParameters`*`)`

La méthode `setEventParameters` définit les para mètre d'événement utilisés par la commande `postEvent`. Ces valeurs sont stockées dans les données de session.

- **eventParameters** — Tableau d'objets NameValuePair définissant les paramètres d'événement.

Par exemple, si l'événement journalise une offre dans l'historique des contacts, vous devez inclure le code de traitement de l'offre.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode `executeBatch` appelant `postEvent`.

```
NameValuePair parmB1 = new NameValuePairImpl();
parmB1.setName("SearchString");
parmB1.setValueAsString("mortgage");
parmB1.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair parmB2 = new NameValuePairImpl();
parmB2.setName("TimeStamp");
parmB2.setValueAsDate(new Date());
parmB2.setValueDataType(NameValuePair.DATA_TYPE_DATETIME);

NameValuePair parmB3 = new NameValuePairImpl();
parmB3.setName("Browser");
parmB3.setValueAsString("IE6");
parmB3.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair parmB4 = new NameValuePairImpl();
parmB4.setName("FlashEnabled");
parmB4.setValueAsNumeric(1.0);
parmB4.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

NameValuePair parmB5 = new NameValuePairImpl();
parmB5.setName("TxAcctValueChange");
parmB5.setValueAsNumeric(0.0);
parmB5.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

```
NameValuePair parmB6 = new NameValuePairImpl();
parmB6.setName("PageTopic");
parmB6.setValueAsString("");
parmB6.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair[] postEventParameters = { parmB1,
 parmB2,
 parmB3,
 parmB4,
 parmB5,
 parmB6
  };
. . .
Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEventParameters(postEventParameters);
postEventCommand.setEvent(eventName);
```

setGetOfferRequests

```
setGetOfferRequests(numberRequested)
```

La méthode **setGetOfferRequests** définit le paramètre de récupération des offres utilisé par la commande `getOffersForMultipleInteractionPoints`.

- **numberRequested** — Tableau d'objets `GetOfferRequest` définissant le paramètre d'extraction des offres.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode `GetOfferRequest` appelant `setGetOfferRequests`.

```
GetOfferRequest request1 = new GetOfferRequest(5, GetOfferRequest.NO_DUPLICATION);
     request1.setIpName("IP1");
     OfferAttributeRequirements offerAttributes1 = new OfferAttributeRequirements();
     NameValuePairImpl attr1 = new NameValuePairImpl("attr1",
          NameValuePair.DATA_TYPE_NUMERIC, 1);
     NameValuePairImpl attr2 = new NameValuePairImpl("attr2",
          NameValuePair.DATA_TYPE_STRING, "value2");
     NameValuePairImpl attr3 = new NameValuePairImpl("attr3",
          NameValuePair.DATA_TYPE_STRING, "value3");
     NameValuePairImpl attr4 = new NameValuePairImpl("attr4",
          NameValuePair.DATA_TYPE_NUMERIC, 4);
     offerAttributes1.setNumberRequested(5);
     offerAttributes1.setAttributes(new NameValuePairImpl[] {attr1, attr2});
     OfferAttributeRequirements childAttributes1 = new OfferAttributeRequirements();
     childAttributes1.setNumberRequested(3);
     childAttributes1.setAttributes(new NameValuePairImpl[] {attr3});
     OfferAttributeRequirements childAttributes2 = new OfferAttributeRequirements();
     childAttributes2.setNumberRequested(3);
     childAttributes2.setAttributes(new NameValuePairImpl[] {attr4});
     offerAttributes1.setChildRequirements(Arrays.asList(childAttributes1,
          childAttributes2));
     request1.setOfferAttributes(offerAttributes1);

     GetOfferRequest request2 = new GetOfferRequest(3, GetOfferRequest.ALLOW_DUPLICATION);
     request2.setIpName("IP2");
     OfferAttributeRequirements offerAttributes2 = new OfferAttributeRequirements();
     offerAttributes2.setNumberRequested(3);
     offerAttributes2.setAttributes(new NameValuePairImpl[] {new NameValuePairImpl("attr5",
```

```
     NameValuePair.DATA_TYPE_STRING, "value5")});
request2.setOfferAttributes(offerAttributes2);

GetOfferRequest request3 = new GetOfferRequest(2, GetOfferRequest.NO_DUPLICATION);
request3.setIpName("IP3");
request3.setOfferAttributes(null);

Command getOffersMultiIPCmd = new CommandImpl();
getOffersMultiIPCmd.setGetOfferRequests(new GetOfferRequest[] {request1,
     request2, request3});
```

setInteractiveChannel

`setInteractiveChannel(`*interactiveChannel*`)`

La méthode `setInteractiveChannel` définit le nom du canal interactif utilisé par la commande `startSession`.

- **interactiveChannel** — Chaîne contenant le nom du canal interactif.

Important : *interactiveChannel* doit correspondre exactement au nom du canal interactif défini dans Campaign. Ce nom est sensible à la casse.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode `executeBatch` appelant `startSession`.

```
String interactiveChannel="Accounts Website";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setInteractiveChannel(interactiveChannel);
```

setInteractionPoint

`setInteractionPoint(`*interactionPoint*`)`

La méthode `setInteractionPoint` définit le nom du point d'interaction utilisé par les commandes `getOffers` et `postEvent`.

- **interactionPoint** — Chaîne contenant le nom du point d'interaction.

Important : *interactionPoint* doit correspondre exactement au nom du point d'interaction défini dans le canal interactif.Ce nom est sensible à la casse.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode `executeBatch` appelant `getOffers`.

```
String interactionPoint = "Overview Page Banner 1";
int numberRequested=1;

Command getOffersCommand = new CommandImpl();
```

```
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(interactionPoint);
getOffersCommand.setNumberRequested(numberRequested);
```

setMethodIdentifier

`setMethodIdentifier(`*`methodIdentifier`*`)`

La méthode `setMethodIdentifier` définit le type de commande contenue dans l'objet de commande.

- **methodIdentifier** — Chaîne contenant le type de commande.
  Valeurs admises :
  - **COMMAND_ENDSESSION** — Représente la méthode `endSession`.
  - **COMMAND_GETOFFERS** — Représente la méthode `getOffers`.
  - **COMMAND_GETPROFILE** — Représente la méthode `getProfile`.
  - **COMMAND_GETVERSION** — Représente la méthode `getVersion`.
  - **COMMAND_POSTEVENT** — Représente la méthode `postEvent`.
  - **COMMAND_SETAUDIENCE** — Représente la méthode `setAudience`.
  - **COMMAND_SETDEBUG** — Représente la méthode `setDebug`.
  - **COMMAND_STARTSESSION** — Représente la méthode `startSession`.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode `executeBatch` appelant `getVersion` et `endSession`.

```
Command getVersionCommand = new CommandImpl();
getVersionCommand.setMethodIdentifier(Command.COMMAND_GETVERSION);

Command endSessionCommand = new CommandImpl();
endSessionCommand.setMethodIdentifier(Command.COMMAND_ENDSESSION);

Command[] commands =
{
    getVersionCommand,
    endSessionCommand
};
```

setNumberRequested

`setNumberRequested(`*`numberRequested`*`)`

La méthode `setNumberRequested` définit le nombre d'offres demandées par la commande `getOffers`.

- **numberRequested** — Entier définissant le nombre d'offres demandées par la commande `getOffers`.

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode executeBatch appelant getOffers.

```
String interactionPoint = "Overview Page Banner 1";
int numberRequested=1;

Command getOffersCommand = new CommandImpl();
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(interactionPoint);
getOffersCommand.setNumberRequested(numberRequested);
```

setRelyOnExistingSession

setRelyOnExistingSession(*relyOnExistingSession*)

La méthode setRelyOnExistingSession définit une valeur booléenne qui indique si la commande startSession utilise une session existante ou non.

Si la valeur est true, l'ID de session de executeBatch doit correspondre à un ID de session existante. Si elle est false, vous devez fournir un nouvel ID de session avec la méthode executeBatch.

- **relyOnExistingSession** — Valeur booléenne (true ou false).

Valeur de retour

Aucune.

Exemple

L'exemple suivant est un extrait d'une méthode executeBatch appelant startSession.

```
boolean relyOnExistingSession=false;
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setRelyOnExistingSession(relyOnExistingSession);
```

A propos de l'interface NameValuePair

De nombreuses méthodes de l'API Interact renvoient des objets NameValuePair ou nécessitent de transmettre des objets NameValuePair en tant qu'arguments. Lors de la transmission en tant qu'arguments dans une méthode, vous devez utiliser l'implémentation par défaut NameValuePairImpl.

getName

getName()

La méthode getName renvoie le nom du composant d'un objet NameValuePair.

Valeur de retour

La méthode getName renvoie une chaîne.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite l'objet de réponse de getProfile.

```
for(NameValuePair nvp : response.getProfileRecord())
{
    System.out.println("Name:"+nvp.getName());
}
```

getValueAsDate

```
getValueAsDate()
```

La méthode `getValueAsDate` renvoie la valeur d'un objet NameValuePair.

Vous devez utiliser `getValueDataType` avant d'utiliser `getValueAsDate` pour confirmer que vous référencez le type de données correct.

Valeur de retour

La méthode `getValueAsDate` renvoie une date.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite un objet NameValuePair et imprime la valeur s'il s'agit d'une date.

```
if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATE))
{
    System.out.println("Value:"+nvp.getValueAsDate());
}
```

getValueAsNumeric

```
getValueAsNumeric()
```

La méthode `getValueAsNumeric` renvoie la valeur d'un objet NameValuePair.

Vous devez utiliser `getValueDataType` avant d'utiliser `getValueAsNumeric` pour confirmer que vous référencez le type de données correct.

Valeur de retour

La méthode `getValueAsNumeric` renvoie un double. Si, par exemple, vous extrayez une valeur initialement stockée dans votre table de profil comme un entier, `getValueAsNumeric` renvoie un double.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite un objet NameValuePair et imprime la valeur si elle est numérique.

```
if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
{
    System.out.println("Value:"+nvp.getValueAsNumeric());
}
```

getValueAsString

```
getValueAsString()
```

La méthode `getValueAsString` renvoie la valeur d'un objet NameValuePair.

Vous devez utiliser `getValueDataType` avant d'utiliser `getValueAsString` pour confirmer que vous référencez le type de données correct.

Valeur de retour

La méthode getValueAsString renvoie une chaîne. Si, par exemple, vous extrayez une valeur initialement stockée dans votre table de profil comme char, varchar, ou char[10], getValueAsString renvoie une chaîne.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite un objet NameValuePair et imprime la valeur s'il s'agit d'une chaîne.

```
if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_STRING))
{
    System.out.println("Value:"+nvp.getValueAsString());
}
```

getValueDataType

```
getValueDataType()
```

La méthode getValueDataType renvoie le type de données d'un objet NameValuePair.

Vous devez utiliser getValueDataType avant d'utiliser getValueAsDate, getValueAsNumeric, ou getValueAsString pour confirmer que vous référencez le type de données correct.

Valeur de retour

La méthode getValueDataType renvoie une chaîne indiquant si NameValuePair contient une donnée, un nombre ou une chaîne.

Valeurs admises :

- **DATA_TYPE_DATETIME** — Date contenant une valeur de date et d'heure.
- **DATA_TYPE_NUMERIC** — Double contenant une valeur numérique.
- **DATA_TYPE_STRING** — Chaîne contenant une valeur de texte.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite l'objet de réponse d'une méthode getProfile.

```
for(NameValuePair nvp : response.getProfileRecord())
{
    System.out.println("Name:"+nvp.getName());
    if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
    {
        System.out.println("Value:"+nvp.getValueAsDate());
    }
    else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
    {
        System.out.println("Value:"+nvp.getValueAsNumeric());
    }
    else
    {
        System.out.println("Value:"+nvp.getValueAsString());
    }
}
```

setName

`setName(name)`

La méthode `setName` définit le nom du composant d'un objet NameValuePair.

- **name** — Chaîne contenant le composant nom d'un objet NameValuePair.

Valeur de retour

Aucune.

Exemple

L'exemple suivant montre comment définir le nom du composant d'un NameValuePair.

```
NameValuePair custId = new NameValuePairImpl();
custId.setName("CustomerId");
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
```

setValueAsDate

`setValueAsDate(valueAsDate)`

La méthode `setValueAsDate` renvoie la valeur d'un objet NameValuePair.

- **valueAsDate** — Date contenant la valeur de date et d'heure d'un objet NameValuePair.

Valeur de retour

Aucune.

Exemple

L'exemple suivant montre comment définir la valeur de composant d'un NameValuePair si la valeur est une date.

```
NameValuePair parm2 = new NameValuePairImpl();
parm2.setName("TimeStamp");
parm2.setValueAsDate(new Date());
parm2.setValueDataType(NameValuePair.DATA_TYPE_DATETIME);
```

setValueAsNumeric

`setValueAsNumeric(valueAsNumeric)`

La méthode `setValueAsNumeric` définit la valeur d'un objet NameValuePair.

- **valueAsNumeric** — Double contenant la valeur numérique d'un objet NameValuePair.

Valeur de retour

Aucune.

Exemple

L'exemple suivant montre comment définir la valeur de composant d'un NameValuePair si la valeur est numérique.

```
NameValuePair parm4 = new NameValuePairImpl();
parm4.setName("FlashEnabled");
parm4.setValueAsNumeric(1.0);
parm4.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

setValueAsString

`setValueAsString(valueAsString)`

La méthode `setValueAsString` définit la valeur d'un objet NameValuePair.

- **valueAsString** — Chaîne contenant la valeur numérique d'un objet NameValuePair.

Valeur de retour

Aucune.

Exemple

L'exemple suivant montre comment définir la valeur de composant d'un NameValuePair si la valeur est numérique.

```
NameValuePair parm3 = new NameValuePairImpl();
parm3.setName("Browser");
parm3.setValueAsString("IE6");
parm3.setValueDataType(NameValuePair.DATA_TYPE_STRING);
```

setValueDataType

`getValueDataType(valueDataType)`

La méthode `setValueDataType` définit le type de données d'un objet NameValuePair.

Valeurs admises :

- **DATA_TYPE_DATETIME** — Date contenant une valeur de date et d'heure.
- **DATA_TYPE_NUMERIC** — Double contenant une valeur numérique.
- **DATA_TYPE_STRING** — Chaîne contenant une valeur de texte.

Valeur de retour

Aucune.

Exemple

Les exemples suivants montrent comment définir le type de données de la valeur d'un NameValuePair.

```
NameValuePair parm2 = new NameValuePairImpl();
parm2.setName("TimeStamp");
parm2.setValueAsDate(new Date());
parm2.setValueDataType(NameValuePair.DATA_TYPE_DATETIME);

NameValuePair parm3 = new NameValuePairImpl();
parm3.setName("Browser");
parm3.setValueAsString("IE6");
parm3.setValueDataType(NameValuePair.DATA_TYPE_STRING);

NameValuePair parm4 = new NameValuePairImpl();
```

```
parm4.setName("FlashEnabled");
parm4.setValueAsNumeric(1.0);
parm4.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

A propos de la classe Offer

La classe `Offer` contient des méthodes qui définissent un objet Offer. Cet objet offre contient un grand nombre de propriétés d'une offre dans Campaign. L'objet d'offre contient les attributs suivants :

- **AdditionalAttributes** — -NameValuePairs contenant les attributs d'offre personnalisée que vous avez définis dans Campaign.
- **Description** — Description de l'offre.
- **EffectiveDate** — Date d'effet de l'offre.
- **ExpirationDate** — Date d'expiration de l'offre.
- **OfferCode** — Code de l'offre.
- **OfferName** — Nom de l'offre.
- **TreatmentCode** — Code de traitement de l'offre.
- **Score** — Score marketing de l'offre, ou le score défini par les `ScoreOverrideTable` si la propriété `enableScoreOverrideLookup` est true.

getAdditionalAttributes

```
getAdditionalAttributes()
```

La méthode `getAdditionalAttributes` renvoie les attributs d'offre personnalisés définis dans Campaign.

Valeur de retour

La méthode `getAdditionalAttributes` renvoie un tableau d'objets NameValuePair.

Exemple

L'exemple suivant effectue un tri en fonction de tous les attributs supplémentaires, vérifie la date d'effet et la date d'expiration, et imprime les autres attributs.

```
for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
{
    // check to see if the effective date exists
    if(offerAttribute.getName().equalsIgnoreCase("effectiveDate"))
    {
        System.out.println("Found effective date");
    }
    // check to see if the expiration date exists
    else if(offerAttribute.getName().equalsIgnoreCase("expirationDate"))
    {
        System.out.println("Found expiration date");
    }
    printNameValuePair(offerAttribute);
    }
}
public static void printNameValuePair(NameValuePair nvp)
{
    // print out the name:
    System.out.println("Name:"+nvp.getName());

    // based on the datatype, call the appropriate method to get the value
    if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_DATETIME)
        System.out.println("DateValue:"+nvp.getValueAsDate());
```

```
        else if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_NUMERIC)
            System.out.println("NumericValue:"+nvp.getValueAsNumeric());
        else
            System.out.println("StringValue:"+nvp.getValueAsString());
    }
```

getDescription

```
getDescription()
```

La méthode getDescription renvoie la description de l'offre définie dans Campaign.

Valeur de retour

La méthode getDescription renvoie une chaîne.

Exemple

L'exemple suivant imprime la description d'une offre.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
     System.out.println("Offer Description:"+offer.getDescription());
}
```

getOfferCode

```
getOfferCode()
```

La méthode getOfferCode renvoie le code de l'offre comme défini dans Campaign.

Valeur de retour

La méthode getOfferCode renvoie un tableau de chaînes contenant le code de l'offre.

Exemple

L'exemple suivant imprime le code d'une offre.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
   System.out.println("Offer Code:"+offer.getOfferCode());
}
```

getOfferName

```
getOfferName()
```

La méthode getOfferName renvoie le nom de l'offre tel qu'il est défini dans Campaign.

Valeur de retour

La méthode getOfferName renvoie une chaîne.

Exemple

L'exemple suivant imprime le nom d'une offre.

```
for(Offer offer : offerList.getRecommendedOffers())
{
// print offer
System.out.println("Offer Name:"+offer.getOfferName());
}
```

getScore

```
getScore()
```

La méthode `getScore` renvoie l'un des résultats suivants :

- Si vous n'avez pas activé la table des offres par défaut, la table de substitution de score, ou l'apprentissage intégré, cette méthode renvoie le score marketing de l'offre, tel qu'il est défini dans l'onglet Stratégie d'interaction.
- Si vous n'avez pas activé la or des offres par défaut, la table de substitution de score, ou l'apprentissage intégré, cette méthode renvoie le score de l'offre, tel qu'il est défini par l'ordre de priorité entre la table des offres par défaut, le score du vendeur et la table de substitution de score.
- Si vous avez activé l'apprentissage intégré, cette méthode renvoie le score final utilisé par l'apprentissage intégré pour ordonner les offres.

Valeur de retour

La méthode `getScore` renvoie un entier représentant le score de l'offre.

Exemple

L'exemple suivant imprime le score d'une offre.

```
for(Offer offer : offerList.getRecommendedOffers())
{
// print offer
System.out.println("Offer Score:"+offer.getOfferScore());
}
```

getTreatmentCode

```
getTreatmentCode()
```

La méthode `getTreatmentCode` renvoie le code de traitement l'offre comme défini dans Campaign.

Etant donné que Campaign utilise le code de traitement pour identifier l'instance de l'offre proposée, ce code doit être renvoyé sous la forme d'un paramètre d'événement lors de l'utilisation de la méthode `postEvent` pour journaliser un événement de contact, d'acceptation, ou de refus de l'offre. Si vous journalisez une acceptation ou un refus d'une offre, vous devez définir la valeur du nom de NameValuePair représentant le code de traitement en tant que `UACIOfferTrackingCode`.

Valeur de retour

La méthode `getTreatmentCode` renvoie une chaîne.

Exemple

L'exemple suivant imprime le code de traitement d'une offre.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
    System.out.println("Offer Treatment Code:"+offer.getTreatmentCode());
}
```

A propos de la classe OfferList

La classe `OfferList` contient des méthodes qui définissent les résultats de la méthode `getOffers`. L'objet OfferList contient les attributs suivants :

- **DefaultString** — Chaîne par défaut définie pour le point d'interaction dans le canal interactif.
- **RecommendedOffers** — Tableau des objets Offer demandés par la méthode `getOffers`.

La classe `OfferList` fonctionnent avec des listes d'offres. Cette classe n'a pas de rapport avec les listes d'offres Campaign.

getDefaultString

```
getDefaultString()
```

La méthode `getDefaultString` renvoie la chaîne par défaut du point d'interaction, comme défini dans Campaign.

Si l'objet RecommendedOffers est vide, vous devez configurer votre point de contact afin qu'il présenter cette chaîne et assure qu'un certain contenu soit présenté. Interact renseigne l'objet DefaultString uniquement si l'objet RecommendedOffers est vide.

Valeur de retour

La méthode `getDefaultString` renvoie une chaîne.

Exemple

L'exemple suivant obtient la chaîne par défaut si l'objet offerList ne contient aucun offre.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
  System.out.println("Offer Name:"+offer.getOfferName());
    }
}
else // count on the default Offer String
    System.out.println("Default offer:"+offerList.getDefaultString());
```

getRecommendedOffers

```
getRecommendedOffers()
```

La méthode `getRecommendedOffers` renvoie un tableau d'objets Offer demandés par la méthode `getOffers`.

Si la réponse à getRecommendedOffer est vide, le point de contact doit présenter le résultat de getDefaultString.

Valeur de retour

La méthode getRecommendedOffers renvoie un objet Offer.

Exemple

L'exemple suivant traite l'objet Offerlist et imprime le nom de l'offre pour toutes les offres recommandées.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
        // print offer
        System.out.println("Offer Name:"+offer.getOfferName());
    }
}
else // count on the default Offer String
System.out.println("Default offer:"+offerList.getDefaultString());
```

A propos de la classe Response

La classe Réponse contient des méthodes qui définissent les résultats des méthodes de la classe InteractAPI. L'objet Response contient les attributs suivants :

- **AdvisoryMessages** — Tableau des messages de recommandation. Cet attribut est renseigné uniquement si il y avait des avertissements ou des erreurs lorsque la méthode a été exécutée.
- **ApiVersion** — Chaîne contenant la version de l'API. Cet attribut est renseigné par la méthode getVersion.
- **OfferList** — Objet OfferList contenant les offres demandées par la méthode getOffers.
- **ProfileRecord** — Tableau des NameValuePairs contenant les données de profil. Cet attribut est renseigné par la méthode getProfile.
- **SessionID** — Chaîne identifiant l'ID session. Elle est renvoyée par toutes les méthodes de classe InteractAPI.
- **StatusCode** — Nombre indiquant si la méthode s'est exécutée sans erreur, avec un avertissement, ou avec des erreurs. Elle est renvoyée par toutes les méthodes de classe InteractAPI.

getAdvisoryMessages

```
getAdvisoryMessages()
```

La méthode getAdvisoryMessages renvoie un tableau de messages de recommandation à partir de l'objet de réponse.

Valeur de retour

La méthode getAdvisoryMessages renvoie un tableau d'objets de Messages de recommandation.

Exemple

L'exemple suivant obtient les objets AdvisoryMessage à partir d'un objet de réponse et itère à travers eux, en imprimant les messages.

```
AdvisoryMessage[] messages = response.getAdvisoryMessages();
    for(AdvisoryMessage msg : messages)
    {
        System.out.println(msg.getMessage());
        // Some advisory messages may have additional detail:
        System.out.println(msg.getDetailMessage());
    }
```

getApiVersion

```
getApiVersion()
```

La méthode `getApiVersion` renvoie la version de l'API d'un objet de réponse.

La méthode `getVersion` renseigne l'attribut APIVersion d'un objet de réponse.

Valeur de retour

L'objet de réponse renvoie une chaîne.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite l'objet de réponse de `getVersion`.

```
if(response.getStatusCode() == Response.STATUS_SUCCESS)
{
    System.out.println("getVersion call processed with no warnings or errors");
    System.out.println("API Version:" + response.getApiVersion());
}
```

getOfferList

```
getOfferList()
```

La méthode `getOfferList` renvoie l'objet OfferList d'un objet de réponse.

La méthode `getOffers` renseigne l'objet OfferList d'un objet de réponse.

Valeur de retour

L'objet de réponse renvoie un objet OfferList.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite l'objet de réponse de `getOffers`.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
    // print offer
    System.out.println("Offer Name:"+offer.getOfferName());
    }
}
```

getAllOfferLists

```
getAllOfferLists()
```

La méthode `getAllOfferLists` renvoie un tableau de tous les Offerlists d'un objet de réponse.

Ceci est utilisé par la méthode `getOffersForMultipleInteractionPoints` qui remplit le tableau des objets OfferList d'un objet de réponse.

Valeur de retour

L'objet de réponse renvoie un tableau OfferList.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite l'objet de réponse de `getOffers`.

```
OfferList[] allOfferLists = response.getAllOfferLists();
 if (allOfferLists != null) {
  for (OfferList ol : allOfferLists) {
   System.out.println("The following offers are delivered for interaction point "
     + ol.getInteractionPointName() + ":");
   for (Offer o : ol.getRecommendedOffers()) {
    System.out.println(o.getOfferName());
   }
  }
 }
```

getProfileRecord

```
getProfileRecord()
```

La méthode `getProfileRecord` renvoie les enregistrements de profil de la session en cours sous la forme d'un tableau d'objets NameValuePair. Ces enregistrements de profil incluent également les `eventParameters` ajouté plus tôt au cours de la session d'exécution.

La méthode `getProfile` renseigne l'enregistrement du profil des objets NameValuePair d'un objet de réponse.

Valeur de retour

L'objet de réponse renvoie un tableau d'objets NameValuePair.

Exemple

L'exemple suivant est un extrait d'une méthode qui traite l'objet de réponse de `getOffers`.

```
for(NameValuePair nvp : response.getProfileRecord())
{
    System.out.println("Name:"+nvp.getName());
    if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
    {
        System.out.println("Value:"+nvp.getValueAsDate());
    }
    else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
    {
         System.out.println("Value:"+nvp.getValueAsNumeric());
    }
```

```
    else
    {
        System.out.println("Value:"+nvp.getValueAsString());
    }
}
```

getSessionID

```
getSessionID()
```

La méthode getSessionID renvoie l'ID de session.

Valeur de retour

La méthode getSessionID renvoie une chaîne.

Exemple

L'exemple suivant montre un message que vous pouvez afficher à la fin ou au début de votre traitement d'erreurs pour indiquer à quelle session appartiennent les erreurs.

```
System.out.println("This response pertains to sessionId:"+response.getSessionID());
```

getStatusCode

```
getStatusCode()
```

La méthode getStatusCode renvoie la code de statut d'un objet de réponse.

Valeur de retour

L'objet de réponse renvoie un entier.

- 0 - STATUS_SUCCESS — La méthode appelée s'est terminée sans erreurs. Il n'y a pas forcément de messages de recommandation.
- 1 - STATUS_WARNING — La méthode appelée s'est terminée avec au moins un avertissement (mais sans erreurs). Consultez les messages de recommandation pour plus de détails.
- 2 - STATUS_ERROR — La méthode appelée ne s'est pas terminée correctement et comporte au moins une erreur. Consultez les messages de recommandation pour plus de détails.

Exemple

Voici un exemple de la façon dont vous pouvez utiliser getStatusCode dans le traitement des erreurs.

```
public static void processSetDebugResponse(Response response)
{
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("setDebug call processed with no warnings or errors");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("setDebug call processed with a warning");
    }
    else
    {
        System.out.println("setDebug call processed with an error");
```

```
        }

        // For any non-successes, there should be advisory messages explaining why
        if(response.getStatusCode() != Response.STATUS_SUCCESS)
            printDetailMessageOfWarningOrError("setDebug",
        response.getAdvisoryMessages());
    }
```

Chapitre 8. A propos de l'API ExternalCallout

Interact propose une macro extensible, EXTERNALCALLOUT, qui peut être utilisée avec vos diagrammes interactifs. Cette macro vous permet d'exécuter une logique personnalisée destinée à communiquer avec des systèmes externes pendant l'exécution d'un diagramme. Par exemple, si vous souhaitez calculer le score de crédit d'un client pendant l'exécution d'un diagramme, vous pouvez créer une classe Java (un appel externe), puis utiliser la macro EXTERNALCALLOUT dans un processus Sélection dans votre diagramme interactif pour obtenir le score de crédit à partir de votre appel externe.

La configuration de EXTERNALCALLOUT comporte deux grandes étapes. Tout d'abord, vous devez créer une classe Java qui implémente l'API ExternalCallout. Deuxièmement, vous devez configurer les propriété de configuration Marketing Platform sur le serveur d'exécution dans la catégorie Interact | flowchart | ExternalCallouts.

Outre les informations contenues dans cette section, le JavaDoc de l'API ExternalCallout est disponible sur n'importe quel serveur d'exécution Interact dans le répertoire Interact/docs/externalCalloutJavaDoc.

Interface IAffiniumExternalCallout

L'API ExternalCallout est contenue dans l'interface IAffiniumExternalCallout. Vous devez implémenter l'interface IAffiniumExternalCallout pour utiliser la macro EXTERNALCALLOUT.

La classe qui implémente IAffiniumExternalCallout doit avoir un constructeur avec lequel elle peut être initialisée par le serveur d'exécution.

- En l'absence de constructeurs dans la classe, le compilateur Java crée un constructeur par défaut qui est suffisant.
- En l'absence de constructeurs avec des arguments, un constructeur public sans argument doit être fourni. Il sera utilisé par le serveur d'exécution.

Lors du développement de votre appel externe, gardez à l'esprit les points suivants :

- Chaque évaluation de l'expression utilisant un appel externe crée une nouvelle instance de la classe. Vous devez gérer les problèmes de sécurité d'unité d'exécution des membres statiques dans la classe.
- Si votre appel externe utilise des ressources système, telles que des fichiers ou une connexion de base de données, vous devez gérer les connexions. Me serveur d'exécution ne comporte pas d'utilitaire de nettoyage automatique des connexions.

Vous devez compiler votre implémentation par rapport à interact_externalcallout.jar qui se trouve dans le répertoire lib de votre installation d'environnement d'exécution IBM Unica Interact.

IAffiniumExternalCallout permet au serveur d'exécution de demander des données à partir de votre classe Java. L'interface se compose de quatre méthodes:

- getNumberOfArguments

© Copyright IBM Corp. 2001, 2012

- getValue
- initialize
- shutdown

Ajout d'un service Web à utiliser avec EXTERNALCALLOUT

La macro EXTERNALCALLOUT reconnaît les appels externes uniquement si vous avez défini les propriétés de configuration appropriées.

Dans l'environnement d'exécution Marketing Platform pour l'environnement de conception, éditez les propriétés de configuration suivantes dans la catégorie Interact > flowchart > externalCallouts.

Propriété de configuration	Paramètre
Catégorie externalCallouts	Créez une nouvelle catégorie pour votre appel externe
class	Noms de classe de votre appel externe
classpath	Chemin d'accès aux classes de vos fichiers d'appel externe
Catégorie Données de paramètre	Si votre appel externe requiert des paramètres, créez de nouvelles propriétés de configuration de paramètre et affectez à chacun une valeur.

getNumberOfArguments

```
getNumberOfArguments()
```

La méthode getNumberOfArguments renvoie le nombre d'arguments attendus par la classe Java avec laquelle vous effectuez l'intégration.

Valeur de retour

La méthode getNumberOfArguments renvoie un entier.

Exemple

L'exemple suivant montre comment imprimer le nombre d'arguments.

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

```
getValue(audienceID, configData, arguments)
```

La méthode getValue exécute la fonctionnalité de base de l'appel externe et renvoie les résultats.

La méthode getValue nécessite les paramètres suivants :

- **audienceID** — Valeur identifiant l'ID de référentiel.
- **configData** — Mappe avec des paires clé-valeur de données de configuration requises par l'appel externe.

- **arguments** — Arguments requis par l'appel externe. Chaque argument peut être une Chaîne, Double, Date, Liste Un argument Liste peut contenir des valeurs NULL, cependant, une Liste ne peut pas contenir, par exemple, une Chaîne et un Double.

  La vérification du type d'argument doit être effectuée dans votre implémentation.

Si la méthode `getValue` échoue pour une raison quelconque, elle renvoie `CalloutException`.

Valeur de retour

La méthode getValue renvoie une liste de Chaînes.

Exemple

```
public List<String> getValue(AudienceId audienceId, Map<String,
  String> configurationData, Object... arguments) throws CalloutException
{
 Long customerId = (Long) audienceId.getComponentValue("Customer");
 // now query scoreQueryUtility for the credit score of customerId
 Double score = scoreQueryUtility.query(customerId);
 String str = Double.toString(score);
 List<String> list = new LinkedList<String>();
 list.add(str);
 return list;
}
```

initialize

```
initialize(configData)
```

La méthode `initialize` est appelée lorsque le serveur d'exécution démarre. S'il existe des opérations qui qui risquent d'affecter les performances lors de l'exécution, telles que le chargement d'une table de base de données, elles doivent être exécutées par cette méthode.

La méthode `initialize` nécessite les paramètres suivants :

- **configData** — Mappe avec des paires clé-valeur de données de configuration requises par l'appel externe.

  Interact lit ces valeurs à partir des paramètres External Callout définis dans la catégorie `Interact > Flowchart > External Callouts > [External Callout] > Parameter Data`.

Si la méthode `initialize` échoue pour une raison quelconque, elle renvoie `CalloutException`.

Valeur de retour

Aucune.

Exemple

```
public void initialize(Map<String, String> configurationData) throws CalloutException
{
    // configurationData has the key-value pairs specific to the environment
    // the server is running in
    // initialize scoreQueryUtility here
}
```

shutdown

```
shutdown(configData)
```

La méthode shutdown est appelée lorsque le serveur d'exécution démarre. Si des tâches de nettoyage sont requises par votre appel externe, elles doivent exécuter à ce moment.

La méthode shutdown nécessite le paramètre suivant :

- **configData** — Mappe avec des paires clé-valeur de données de configuration requises par l'appel externe.

Si la méthode shutdown échoue pour une raison quelconque, elle renvoie CalloutException.

Valeur de retour

Aucune.

Exemple

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
    // shutdown scoreQueryUtility here
}
```

Exemple d'API ExternalCallout

1. Créez un fichier appelé GetCreditScore.java ayant le contenu suivant. Ce fichier suppose qu'il existe une classe appelée ScoreQueryUtility qui extrait un score à partir d'une application de modélisation.

```
import java.util.Map;
import com.unicacorp.interact.session.AudienceId;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.IAffiniumExternalCallout;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.CalloutException;
import java.util.Random;

public class GetCreditScore implements IAffiniumExternalCallout
{
 // the class that has the logic to query an external system for a customer's credit score
 private static ScoreQueryUtility scoreQueryUtility;
 public void initialize(Map<String, String> configurationData) throws CalloutException
 {
  // configurationData has the key- value pairs specific to the environment the server is running in
  // initialize scoreQueryUtility here
 }

 public void shutdown(Map<String, String> configurationData) throws CalloutException
 {
 // shutdown scoreQueryUtility here
 }

 public int getNumberOfArguments()
 {
  // do not expect any additional arguments other than the customer's id
  return 0;
 }

 public List<String> getValue(AudienceId audienceId, Map<String, String> configurationData,
  Object... arguments) throws CalloutException
 {
  Long customerId = (Long) audienceId.getComponentValue("Customer");
  // now query scoreQueryUtility for the credit score of customerId
```

```
  Double score = scoreQueryUtility.query(customerId);
  String str = Double.toString(score);
  List<String> list = new LinkedList<String>();
  list.add(str);
  return list;
 }
}
```

2. Compilez `GetCreditScore.java` en `GetCreditScore.class`.
3. Créez un fichier jar appelé `creditscore.jar` contenant `GetCreditScore.class` et les autres fichiers de classe qu'il utilise.
4. Copiez le fichier jar à un emplacement sur le serveur d'exécution, par exemple `/data/interact/creditscore.jar`.
5. Créez un appel externe nommé `GetCreditScore` et le chemin d'accès aux classes `/data/interact/creditscore.jar` dans la catégorie `externalCallouts` sur la page Gestion des configurations.
6. Dans un diagramme interactif, l'appel externe peut être utilisé en tant que `EXTERNALCALLOUT('GetCreditScore')`.

Interface IInteractProfileDataService

L'API Profile est contenue dans l'interface `iInteractProfileDataService`. Cette interface vous permet d'importer des données hiérarchiques dans une session Interact via une ou plusieurs sources de données externes (par exemple un fichier à plat, un service Web, etc.), lors du démarrage de la session Interact ou de la modification d l'ID de référentiel Interact.

Pour développer une importation de données hiérarchiques avec l'API Profile Data Services, vous devez écrire une classe Java qui extrait les informations depuis n'importe quelle source de données et la mappe avec un objet `ISessionDataRootNode`, puis désigne ces données mappées avec la macro EXTERNALCALLOUT.

Vous devez compiler votre implémentation par rapport à `interact_externalcallout.jar` qui se trouve dans le répertoire `lib` de votre installation d'environnement d'exécution IBM Unica Interact.

Pour consulter un ensemble complet de documentation Javadoc sur l'utilisation de cette interface, accédez aux fichiers dans *`Interact_home`*`/docs/externalCalloutJavaDoc` avec un navigateur Web.

Pour obtenir un exemple d'implémentation de l'utilisation de Profile Data Service, avec des descriptions commentées de l'implémentation de l'exemple, voir *`Interact_home`*`/samples/externalcallout/XMLProfileDataService.java`.

Ajout d'une source de données à utiliser avec Profile Data Services

La macro `EXTERNALCALLOUT` reconnaît une source de données pour l'importation des données hiérarchiques de Profile Data Services uniquement si vous avez défini les propriétés de configuration appropriées.

Dans l'environnement d'exécution Marketing Platform pour l'environnement de conception, ajoutez ou définissez les propriétés de configuration suivantes dans la catégorie `Interact > profile > Audience Levels > [AudienceLevelName] > Profile Data Services`.

Propriété de configuration	Paramètre
Catégorie `Nouveau nom de catégorie`	Nom de la source de données que vous définissez. Le nom que vous entrez ici doit être unique dans les sources de données pour le même référentiel.
`enabled`	Indique si cette source de données est activée pour le référentiel auquel elle est affectée.
`className`	Nom qualifié complet de la classe de source de données qui implémente `IInteractProfileDataService`
`classPath`	Chemin d'accès aux fichiers de classe de Profile Data Services. Si vous l'omettez, le chemin d'accès aux classes du serveur d'applications qui le contient est utilisé par défaut.
Catégorie `priorité`	Priorité de cette source de données dans le référentiel. Il doit s'agir d'une valeur unique dans toutes les sources de données de chaque référentiel. Si une priorité est définie sur 100 pour une source de données, aucune autre source de données dans le référentiel ne peut avoir de priorité de 100.

Chapitre 9. Utilitaires IBM Unica Interact

Cette section décrit les utilitaires d'administration fournis avec Interact.

Utilitaire Run Deployment (runDeployment.sh/.bat)

L'outil de ligne de commande runDeployment vous permet de déployer un canal interactif pour un groupe de serveur spécifique depuis la ligne de commande, en utilisant les paramètres fournis par un fichier deployment.properties. Ce fichier décrit tous les paramètres possibles et est disponible dans le même emplacement que l'outil runDeployment. La possibilité d'exécuter un déploiement de canal interactif depuis la ligne de commande est particulièrement utile lorsque vous utilisez la fonction OffersBySQL. Par exemple, vous pouvez configurer un diagramme par lots Campaign afin de l'exécuter à intervalles réguliers. Lorsque l'exécution du diagramme se termine, un déclencheur peut être appelé pour initialiser le déploiement des offres dans la table OffersBySQL à l'aide de cet outil de ligne de commande.

Description

Vous pouvez trouver l'outil de ligne de commande runDeployment qui est installé automatiquement sur le serveur de phase de conception Interact, à l'emplacement suivant :

Interact_home/interactDT/tools/deployment/runDeployment.sh (ou runDeployment.bat sur un serveur Windows)

Le seul argument passé à la commande est l'emplacement d'un fichier appelé deployment.properties qui décrit tous les paramètres possibles requis pour déployer la combinaison de groupe de serveur canal interactif/exécution. Un exemple de fichier est fourni à titre de référence.

Remarque : Avant d'utiliser l'utilitaire runDeployment, vous devez d'abord le modifier avec un éditeur de texte pour indiquer l'emplacement de l'environnement d'exécution Java Java sur le serveur. Par exemple, vous pouvez indiquer *Interact_home*/jre ou *Platform_home*/jre comme chemin d'accès, si l'un de ces répertoires contient l'exécution Java que l'utilitaire doit utiliser. Une autre solution consiste à indiquer le chemin d'accès à tout environnement d'exécution Java pris en charge pour cette édition des produits IBM Unica.

Utilisation de l'utilitaire runDeployment dans un environnement sécurisé (SSL)

Pour utiliser l'utilitaire runDeployment, lorsque la sécurité a été activée sur le serveur Interact (et par conséquent avec une connexion via un port SSL), vous devez ajouter la propriété Java du fichier de clés certifiées, comme suit :

1. Lorsque vous éditez le fichier deployment.properties pour votre déploiement de canal interactif, modifiez la propriété deploymentURL afin d'utiliser l'URL SSL sécurisée, comme dans cet exemple :

   deploymentURL=https://<*HOST*>.<*DOMAIN*>:<*PORT*>/Campaign/interact/InvokeDeploymentServlet

© Copyright IBM Corp. 2001, 2012

2. Editez le script runDeployment.sh ou runDeployment.bat à l'aide d'un éditeur de texte afin d'ajouter l'argument suivant à la ligne commençant par ${JAVA_HOME} :

   -Djavax.net.ssl.trustStore=<*TrustStorePath*>

   Par exemple, la ligne se présente comme suit lorsque vous avez ajouté l'argument du fichier de clés certifiées :

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

   Remplacez<*TrustStorePath*> par le chemin d'accès au fichier de clés certifiées SSL réel.

Exécution de l'utilitaire

Lorsque vous avez édité l'utilitaire pour qu'il fournisse l'environnement d'exécution Java, et avez personnalisé une copie du fichier deployment.properties afin qu'il corresponde à votre environnement, lancez cette commande pour exécuter l'utilitaire :

Interact_home/interactDT/tools/deployment/runDeployment.sh *deployment.properties*

Remplacez *Interact_home* par la valeur réelle de l'installation de la phase de conception Interact, et remplacez *deployment.properties* par le chemin et le nom réels du fichier de propriétés que vous avez personnalisé pour ce déploiement.

Exemple de fichier deployment.properties

L'exemple de fichier deployment.properties contient la liste commentée de tous les paramètres que vous devez personnaliser pour le faire correspondre à votre environnement. Cet exemple de fichier contient également des commentaires qui décrivent chaque paramètre, et explique pourquoi il peut être nécessaire de personnaliser une valeur particulière.

```
##############################################################################
#
# Les propriétés suivantes sont envoyées au programme InvokeDeploymentClient.
# Le programme recherche un paramètre deploymentURL. Ce programme publie une demande
# concernant à cette URL. Tous les autres paramètres sont publiés en tant
# que paramètres dans cette demande. Le programme vérifie alors le statut
# du déploiement et revient lorsque le déploiement est terminé
# (ou si le délai waitTime indiqué a été atteint).
#
# La sortie de ce programme a le format suivant :
# <STATE> : <Misc Detail>
#
# où état peut être l'une des valeurs suivantes :
# ERROR
# RUNNING
# SUCCESS
#
# Misc Detail sont les données qui renseignent normalement la zone de
  message d'état
# dans l'interface graphique de déploiement de la page de synthèse IC.
  REMARQUE : il peut y avoir des balises HTML
# dans Misc Detail
#
##############################################################################

##############################################################################
# deploymentURL : URL du servlet InvokeDeployment qui réside dans la phase
```

```
# de conception Interact. Elle doit avoir le format suivant :
# http://dt_host:port/Campaign/interact/InvokeDeploymentServlet
################################################################################
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet

################################################################################
# dtLogin : il s'agit de la connexion que vous devez utiliser pour vous connecter
# à la phase de conception si
# si vous vouliez déployer IC via l'interface graphique de déploiement
# dans la page de synthèse IC.
################################################################################
dtLogin=asm_admin

################################################################################
# dtPW :  PW qui accompagne le dtLogin
################################################################################
dtPW=

################################################################################
# icName : nom du canal interactif à déployer
################################################################################
icName=ic1

################################################################################
# partition :  nom de la partition
################################################################################
partition=partition1

################################################################################
# request : type de demande à exécuter par cet outil
# Actuellement, il existe deux comportements. Si la valeur est "deploy",
# le déploiement est exécuté. Avec toutes les autres valeurs, l'outil
# renvoie simplement le statut du dernier déploiement
# de l'IC indiqué.
################################################################################
request=deploy

################################################################################
# serverGroup : Nom du groupe de serveurs qui doit
# déployer l'IC.
################################################################################
serverGroup=defaultServerGroup

################################################################################
# serverGroupType : Indique si ce déploiement concerne le
# un groupe de serveur de production ou de serveurs de test. 1 indique
# la production, 2 indique le test.
################################################################################
serverGroupType=1

################################################################################
# rtLogin : Compte utilisé pour authentifier le groupe de serveurs
# sur lequel vous effectuez le déploiement.
################################################################################
rtLogin=asm_admin

################################################################################
# rtPW : Mot de passe associé au rtLogin
################################################################################
rtPW=

################################################################################
# waitTime : Lorsque l'outil envoie la demande de déploiement, l'outil
# vérifie le statut du déploiement. Si le déploiement n'est pas
# terminé (ou a échoué), l'outil continue à sonder le système
# pour connaître le statut jusqu'à la fin du déploiement, OU
# l'échéance du délai waitTime indiqué (en secondes).
```

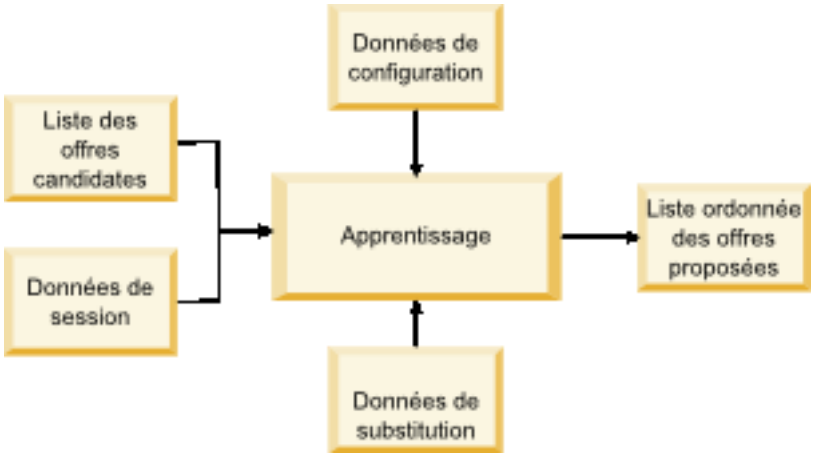
```
################################################################################
waitTime=5

################################################################################
# pollTime : Si le statut d'un déploiement est encore en cours
# d'exécution, l'outil continue à vérifier le statut. Il se met en
# veille entre les  contrôles de statut pendant un nombre de secondes
# en fonction duparamètre pollTime.
################################################################################
pollTime=3

################################################################################
# global : Le choix de la valeur false empêche le déploiement des
# paramètres globaux par l'outil. La non-disponibilité de la
# propriété permet de déployer les paramètres globaux.
################################################################################
global=true
```

Chapitre 10. A propos de l'API d'apprentissage

Interact fournit un module d'apprentissage, qui utilise un algorithme bayesien naïf pour surveiller les actions des visiteurs et proposer des offres optimale (en termes d'acceptation). Vous pouvez implémenter la même interface Java avec vos propres algorithmes à l'aide de l'API d'apprentissage pour créer votre propre module d'apprentissage.

Remarque : Si vous utilisez l'apprentissage externe, les rapports d'exemple sur l'apprentissage (rapports Détails de formation à l'offre interactive et Analyse de l'évolution de segment interactif) ne renvoient pas de données valides.

Au niveau le plus simple, l'API d'apprentissage fournit des méthodes permettant de collecter des données à partir de l'environnement d'exécution et de revenir à une liste ordonnée des offres recommandées.



Vous pouvez collecter les données suivantes à partir de Interact

- Données de contact de l'offre
- Données d'acceptation de l'offre
- Toutes les données de session
- Donnée d'offre spécifiques à Campaign
- Propriétés de configuration définies dans la catégorie `apprentissage` pour l'environnement de conception et la catégorie `offerserving` pour l'environnement d'exécution.

Vous pouvez utiliser ces données dans votre algorithmes pour créer une liste d'offres proposées. Vous pouvez ensuite renvoyer une liste des offres recommandées, par ordre de recommandation décroissante.

Bien que cela ne soit pas représenté dans le diagramme, vous pouvez également utiliser l'API d'apprentissage pour collecter des données pour votre implémentation de l'apprentissage. Vous pouvez conserver ces données en mémoire, ou les journaliser dans un fichier ou une base de données en vue d'une analyse ultérieure.

Après avoir créé vos classes Java , vous pouvez les convertir en fichiers jar. Une fois que vous avez créé vos fichiers jar, vous devez également configurer

© Copyright IBM Corp. 2001, 2012

l'environnement d'exécution pour reconnaître votre module d'apprentissage externe en éditant les propriétés de configuration. Vous devez copier vos classes Java ou les fichiers jar sur chaque serveur d'exécution à l'aide de votre module d'apprentissage externe.

Outre les informations contenues dans ce guide, le JavaDoc de l'API d'optimiseur d'apprentissage est disponible sur tous les serveurs d'exécution dans le répertoire `Interact/docs/learningOptimizerJavaDoc`.

Vous devez compiler votre implémentation par rapport à `interact_learning.jar` qui se trouve dans le répertoire `lib` de votre installation d'environnement d'exécution Interact.

Lorsque vous écrivez votre implémentation d'apprentissage personnalisée, vous devez garder à l'esprit les directives suivantes.

- Les performances sont essentielles.
- Vous devez travailler avec le traitement multitâche et assurer la sécurité des unités d'exécution.
- Vous devez gérer toutes les ressources externes en pensant aux modes d'échec et à la performance.
- Utilisez les exceptions, la journalisation (log4j) et la mémoire dans les cas appropriés.

Activation de l'apprentissage externe

Vous pouvez utiliser l'API d'apprentissage Java API pour écrire votre propre module d'apprentissage. Vous devez configurer l'environnement d'exécution afin qu'il reconnaisse votre utilitaire d'apprentissage dans Marketing Platform.

Dans Marketing Platform, pour l'environnement d'exécution, éditez les propriétés de configuration suivantes dans la catégorie `Interact > offerserving`.Les propriétés de configuration de l'API de l'optimiseur d'apprentissage existent dans la catégorie `Interact > offerserving > External Learning Config`.

Propriété de configuration	Paramètre
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Nom de classe de l'apprentissage externe
`externalLearningClassPath`	Chemin d'accès à la classe ou fichiers JAR sur le serveur d'exécution pour l'apprentissage externe. Si vous utilisez un groupe de serveurs et si tous les serveurs d'exécution référencent la même instance de Marketing Platform, une copie de la classe ou des fichiers jar doit être présente au même emplacement sur chaque serveur.

Vous devez redémarrer le serveur d'exécution Interact pour que les modifications prennent effet.

Interface ILearning

L'API d'apprentissage est bâtie autour de l'interface `ILearning`. Vous devez implémenter l'interface `ILearning` pour prendre en charge la logique personnalisée de votre module d'apprentissage.

Entre autres, l'interface `ILearning` vous permet de collecter des données à partir de l'environnement d'exécution de votre classe Java , et d'envoyer une liste des offres recommandées au serveur d'exécution.

initialize

```
initialize(ILearningConfig config, boolean debug)
```

La méthode `initialize` est appelée une fois lorsque le serveur d'exécution démarre. S'il existe des opérations qui n'ont pas besoin d'être répétées, mais qui peuvent affecter les performances lors de l'exécution, telles que le chargement des données statiques à partir d'une table de base de données, elles doivent être exécutées par cette méthode.

- **config** — Un objet `ILearningConfig` définit toutes les propriétés de configuration relatives à l'apprentissage.
- **debug** — Valeur booléenne. S'il s'agit de `true`, indique que la prolixité du niveau de journalisation du système d'environnement d'exécution est définie sur debug. Pour obtenir de meilleurs résultats, sélectionnez cette valeur avant l'écriture dans un journal.

Si la méthode `initialize` échoue pour une raison quelconque, elle envoie une `LearningException`.

Valeur de retour

Aucune.

logEvent

```
logEvent(ILearningContext context,
             IOffer offer,
             IClientArgs clientArgs,
             IInteractSession session,
             boolean debug)
```

La méthode `logEvent` est appelée par le serveur d'exécution lorsque l'API Interact envoie un événement qui est configuré pour se connecter comme un contact ou une réponse. Utilisez cette méthode pour journaliser les données de contact et de réponse dans une base de données ou un fichier à des fins de génération de rapports et d'apprentissage. Par exemple, si vous souhaitez déterminer via un algorithme la probabilité qu'un client accepte une offre en fonction de critères, utilisez cette méthode pour journaliser les données.

- **context** — Objet `ILearningContext` définissant le contexte d'apprentissage de l'événement, par exemple, contact, acceptation ou refus.

- **offer** — Objet `IOffer` de la définition de l'offre à propos de laquelle cet événement est journalisé.
- **clientArgs** — Objet `IClientArgs` définissant des paramètres. Actuellement, `logEvent` ne nécessite pas de `clientArgs`, aussi ce paramètre peut-il être vide.
- **session** — Objet `IInteractSession` définissant toutes les données de session.
- **debug** — Valeur booléenne. S'il s'agit de `true`, indique que la prolixité du niveau de journalisation du système d'environnement d'exécution est définie sur debug. Pour obtenir de meilleurs résultats, sélectionnez cette valeur avant l'écriture dans un journal.

Si la méthode `logEvent` échoue, elle envoie un `LearningException`.

Valeur de retour

Aucun.

optimizeRecommendList

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

La méthode `optimizeRecommendList` doit prendre la liste des offres recommandées et les données de session et renvoyer une liste contenant le nombre d'offres demandées. La méthode `optimizeRecommendList` doit classer les offres dans un ordre quelconque avec votre propre algorithme d'apprentissage. La liste des offres doit être ordonnée de sorte que les offres à proposer en premier soient au début de la liste. Par exemple, si votre algorithme d'apprentissage donne un score faible aux meilleures offres, les offres doivent être ordonnées comme suit : 1, 2, 3. Si votre algorithme d'apprentissage donne un score élevé aux meilleures offres, les offres doivent être ordonnées comme suit : 100, 99, 98.

La méthode `optimizeRecommendList` nécessite les paramètres suivants :

- **recList**— Liste des objets de traitement (offres) recommandés par l'environnement d'exécution.
- **clientArg** — Objet IClientArgs contenant au moins le nombre d'offres demandées par l'environnement d'exécution.
- **session** — Objet IInteractSession définissant toutes les données de session.
- **debug** — Valeur booléenne. S'il s'agit de `true`, indique que la prolixité du niveau de journalisation du système d'environnement d'exécution est définie sur debug. Pour obtenir de meilleurs résultats, sélectionnez cette valeur avant l'écriture dans un journal.

Si la méthode `optimizeRecommendList` échoue, elle envoie une `LearningException`.

Valeur de retour

La méthode `optimizeRecommendList` renvoie une liste d'objets `ITreatment`.

reinitialize

```
reinitialize(ILearningConfig config,
                  boolean debug)
```

L'environnement d'exécution appelle la méthode `reinitialize` chaque fois qu'il y a un nouveau déploiement. Cette méthode passe toutes les données de configuration d'apprentissage. Si vous avez des services requis par l'API d'apprentissage qui lisent les propriétés de configuration, cette interface doit les redémarrer.

- **config** — Objet `ILearningConfig` qui contient toutes les propriétés de configuration.
- **debug** — Valeur booléenne. S'il s'agit de `true`, indique que la prolixité du niveau de journalisation du système d'environnement d'exécution est définie sur debug. Pour obtenir de meilleurs résultats, sélectionnez cette valeur avant l'écriture dans un journal.

Si la méthode `logEvent` échoue, elle envoie une `LearningException`.

Valeur de retour

Aucun.

shutdown

```
shutdown(ILearningConfig config, boolean debug)
```

L'environnement d'exécution appelle la méthode `shutdown` lorsque le serveur d'exécution s'arrête. Si des tâches de nettoyage sont requises par votre module d'apprentissage, elles doivent exécuter à ce moment.

La méthode `shutdown` nécessite les paramètres suivants.

- **config** — Objet `ILearningConfig` qui définit toutes les propriétés de configuration.
- **debug** — Valeur booléenne. S'il s'agit de `true`, indique que la prolixité du niveau de journalisation du système d'environnement d'exécution est définie sur debug. Pour obtenir de meilleurs résultats, sélectionnez cette valeur avant l'écriture dans un journal.

Si la méthode `shutdown` échoue pour une raison quelconque, elle envoie une `LearningException`.

Valeur de retour

Aucune.

Interface IAudienceID

L'interface `IAudienceID` prend en charge l'interface `IInteractSession`. Il s'agit d'une interface pour l'ID de référentiel. Etant donné que votre ID de référentiel peut être composé de plusieurs parties, cette interface permet d'accéder à tous les éléments de l'ID de référentiel ainsi qu'à son nom.

getAudienceLevel

```
getAudienceLevel()
```

La méthode `getAudienceLevel` renvoie le référentiel.

Valeur de retour

La méthode `getAudienceLevel` renvoie une chaîne qui définit le référentiel.

getComponentNames

```
getComponentNames()
```

La méthode `getComponentNames` obtient l'ensemble de noms des composants qui constituent le référentiel. Par exemple, si votre ID de référentiel est constitué des valeurs `customerName` et `accountID`, `getComponentNames` renvoie un ensemble contenant les chaînes customerName et accountID.

Valeur de retour

Un ensemble de chaînes contenant les noms des composants de l'ID de référentiel.

getComponentValue

```
getComponentValue(String componentName)
```

La méthode `getComponentValue` renvoie la valeur du composant indiqué.

- **componentName** — Chaîne définissant le nom du composant dont vous souhaitez extraire la valeur. Cette chaîne est insensible à la casse.

Valeur de retour

La méthode `getComponentValue` renvoie un objet qui définit la valeur du composant.

IClientArgs

L'interface `IClientArgs` prend en charge l'interface `ILearning`. Cette interface est une abstraction qui englobe toutes les données transmises au serveur à partir du point de contact et qui ne sont pas déjà couvertes par les données de session. Par exemple, le nombre d'offres demandées par la méthode `getOffers` de l'API Interact. Ces données sont stockées dans une mappe.

getValue

`getValue(int clientArgKey)`

La méthode `getValue` renvoie la valeur de l'élément de mappe demandée.

Les éléments suivants sont requis dans la mappe.

- **1** — NUMBER_OF_OFFERS_REQUESTED. Nombre d'offres demandées par la méthode `getOffers` de l'API Interact. Cette constante renvoie un entier.

Valeur de retour

La méthode `getValue` renvoie un objet qui définit la valeur de la constante de mappe demandée.

IInteractSession

L'interface `IInteractSession` prend en charge l'interface `ILearning`. Il s'agit d'une interface pour la session en cours dans l'environnement d'exécution.

getAudienceId

`getAudienceId()`

La méthode `getAudienceId` renvoie un objet `AudienceID`. Utilisez l'interface `IAudienceID` pour extraire les valeurs.

Valeur de retour

La méthode `getAudienceId` renvoie un objet `AudienceID`.

getSessionData

`getSessionData()`

La méthode `getSessionData` renvoie une mappe non modifiable des données de session où le nom de la variable de session est la clé. Le nom de la variable de session est toujours en majuscules. Utilisez l'interface `IInteractSessionData` pour extraire les valeurs.

Valeur de retour

La méthode `getSessionData` renvoie un objet `IInteractSessionData`.

Interface IInteractSessionData

L'interface `IInteractSessionData` prend en charge l'interface `ILearning`. Il s'agit d'une interface pour les données de la session d'exécution du visiteur en cours. Les données de session sont stockées sous la forme d'une liste de paires nom-valeur. Vous pouvez également utiliser cette interface pour modifier la valeur des données dans la session d'exécution.

getDataType

`getDataType(string parameterName)`

La méthode `getDataType` renvoie le type de données du nom de paramètre indiqué.

Valeur de retour

La méthode `getDataType` renvoie un objet InteractDataType. ntearctDataType est une énumération Java représentée par Inconnu, String, Double, Date ou Liste.

getParameterNames

```
getParameterNames()
```

La méthode `getParameterNames` renvoie un ensemble de tous les noms des données dans la session en cours.

Valeur de retour

La méthode `getParameterNames` renvoie un ensemble de noms pour lesquels des valeurs ont été définies. Chaque nom de l'ensemble peut être transmis dans getValue(String) pour renvoyer une valeur.

getValue

```
getValue(parameterName)
```

La méthode `getValue` renvoie la valeur de l'objet correspondant au `parameterName` indiqué. L'objet peut être Chaîne, Double ou Date.

La méthode `getValue` nécessite le paramètre suivant :

- **parameterName** — Chaîne définissant la paire nom-valeur de la session de données.

Valeur de retour

La méthode `getValue` renvoie un objet contenant la valeur du paramètre nommé.

setValue

```
setValue(string parameterName, object value)
```

La méthode `setValue` permet de définir une valeur pour le `parameterName` indiqué. La valeur peut être Chaîne, Double ou Date.

La méthode `setValue` nécessite les paramètres suivants :

- **parameterName** — Chaîne définissant la paire nom-valeur de la session de données.
- **value** — Objet définissant la valeur du paramètre désigné.

Valeur de retour

Aucune.

ILearningAttribute

L'interface `ILearningAttribute` prend en charge l'interface `ILearningConfig`. Il s'agit d'une interface des attributs d'apprentissage définis dans les propriétés de configuration, dans la catégorie `learningAttributes`.

getName

```
getName()
```

La méthode `getName` renvoie le nom de l'attribut d'apprentissage.

Valeur de retour

La méthode `getName` renvoie une chaîne qui définit le nom de l'attribut d'apprentissage.

ILearningConfig

L'interface `ILearningConfig` prend en charge l'interface `ILearning`. Il s'agit d'une interface pour les propriétés de configuration d'apprentissage. Toutes ces méthodes renvoient la valeur de la propriété.

L'interface se compose de 15 méthodes :

- **getAdditionalParameters** — Renvoie une mappe des propriétés supplémentaires définies dans la catégorie `External Learning Config`
- **getAggregateStatsIntervalInMinutes** — Renvoie un entier
- **getConfidenceLevel** — Renvoie un entier
- **getDataSourceName** — Renvoie une chaîne
- **getDataSourceType** — Renvoie une chaîne
- **getInsertRawStatsIntervalInMinutes** — Renvoie un entier
- **getLearningAttributes** — Renvoie une liste des objets `ILearningAttribute`
- **getMaxAttributeNames** — Renvoie un entier
- **getMaxAttributeValues** — Renvoie un entier
- **getMinPresentCountThreshold** — Renvoie un entier
- **getOtherAttributeValue** — Renvoie une chaîne
- **getPercentRandomSelection** — Renvoie un entier
- **getRecencyWeightingFactor** — Renvoie une variable flottante
- **getRecencyWeightingPeriod** — Renvoie un entier
- **isPruningEnabled** - Renvoie une valeur booléenne

ILearningContext

L'interface `ILearningContext` prend en charge l'interface `ILearning`.

getLearningContext

```
getLearningContext()
```

La méthode `getLearningContext` renvoie la constante qui nous indique s'il s'agit d'un scénario de contact, d'acceptation ou de refus.

- **1**—LOG_AS_CONTACT
- **2**—LOG_AS_ACCEPT
- **3**—LOG_AS_REJECT

4 et 5 sont réservés en vue d'un usage ultérieur.

Valeur de retour

La méthode `getLearningContext` renvoie un entier.

getResponseCode

```
getResponseCode()
```

La méthode `getResponseCode` renvoie le code de réponse affecté à cette offre. Cette valeur doit exister dans la table `UA_UsrResponseType` dans les tables système Campaign.

Valeur de retour

La méthode `getResponseCode` renvoie une chaîne qui définit le code de réponse.

IOffer

L'interface `IOffer` prend en charge l'interface `ITreatment`. Il s'agit d'une interface pour l'objet d'offre défini dans l'environnement d'exécution. Utilisez l'interface `IOffer` pour collecter les détails de l'offre dans l'environnement d'exécution.

getCreateDate

```
getCreateDate()
```

La méthode `getCreateDate` renvoie la date de création de l'offre.

Valeur de retour

La méthode `getCreateDate` renvoie la date de création de l'offre.

getEffectiveDateFlag

```
getEffectiveDateFlag()
```

La méthode `getEffectiveDateFlag` renvoie un nombre qui définit la date effective de l'offre.

- **0** — La date effective est une date absolue, telle que le 15 mars 2012.
- **1** — La date effective est la date de la recommandation.

Valeur de retour

La méthode `getEffectiveDateFlag` renvoie un entier qui définit la date effective de l'offre.

getExpirationDateFlag

```
getExpirationDateFlag()
```

La méthode `getExpirationDateFlag` renvoie un entier qui définit la date d'expiration de l'offre.

- **0** — Une date absolue, par exemple le 15 mars 2012.
- **1** — Un certain nombre de jours après la recommandation, par exemple le 14.
- **2** — La fin du mois suivant la recommandation. Si une offre est présentée le 31 mars, l'offre expire ce jour-là.

Valeur de retour

La méthode `getExpirationDateFlag` renvoie un entier qui définit la date d'expiration de l'offre.

getOfferAttributes

```
getOfferAttributes()
```

La méthode `getOfferAttributes` renvoie les attributs de l'offre définis pour l'offre sous la forme d'un objet `IOfferAttributes`.

Valeur de retour

La méthode `getOfferAttributes` renvoie un objet `IOfferAttributes`.

getOfferCode

```
getOfferCode()
```

La méthode `getOfferCode` renvoie le code de l'offre comme défini dans Campaign.

Valeur de retour

La méthode `getOfferCode` renvoie un objet `IOfferCode`.

getOfferDescription

```
getOfferDescription()
```

La méthode `getOfferDescription` renvoie la description de l'offre définie dans Campaign.

Valeur de retour

La méthode `getOfferDescription` renvoie une chaîne.

getOfferID

```
getOfferID()
```

La méthode `getOfferID` renvoie le ID de l'offre tel qu'il est défini dans Campaign.

Valeur de retour

La méthode `getOfferID` renvoie une valeur longue qui définit le ID de l'offre.

getOfferName

```
getOfferName()
```

La méthode `getOfferName` renvoie le nom de l'offre tel qu'il est défini dans Campaign.

Valeur de retour

La méthode `getOfferName` renvoie une chaîne.

getUpdateDate

```
getUpdateDate()
```

La méthode `getUpdateDate` renvoie la date de la dernière mise à jour de l'offre.

Valeur de retour

La méthode `getUpdateDate` renvoie la date de la dernière mise à jour de l'offre.

IOfferAttributes

L'interface `IOfferAttributes` prend en charge l'interface `IOffer`. Il s'agit d'une interface pour les attributs d'offre qui sont définis pour une offre dans l'environnement de conception. Utilisez l'interface `IOfferAttributes` pour collecter les attributs de l'offre dans l'environnement d'exécution.

getParameterNames

```
getParameterNames()
```

La méthode `getParameterNames` renvoie une liste des noms de paramètres de l'offre.

Valeur de retour

La méthode `getParameterNames` renvoie un ensemble définissant la liste des noms de paramètres de l'offre.

getValue

```
getValue(String parameterName)
```

La méthode `getValue` renvoie la valeur d'un attribut d'offre donné.

Valeur de retour

La méthode `getValue` renvoie un objet qui définit la valeur de l'attribut de l'offre.

Interface IOfferCode

L'interface `IOfferCode` prend en charge l'interface `ILearning`. Il s'agit d'une interface pour le code d'offre défini pour une offre dans l'environnement de conception. Un code d'offre peut comprendre une ou plusieurs chaînes. Utilisez l'interface `IOfferCode` pour collecter le code de l'offre dans l'environnement d'exécution.

getPartCount

```
getPartCount()
```

La méthode `getPartCount` renvoie le nombre d'éléments qui constituent un code d'offre.

Valeur de retour

La méthode `getPartCount` renvoie un entier définissant le nombre d'éléments qui constituent le code de l'offre.

getParts

```
getParts()
```

La méthode `getParts` obtient une liste non modifiable des éléments du code de l'offre.

Valeur de retour

La méthode `getParts` obtient une liste non modifiable des éléments du code de l'offre.

LearningException

La classe `LearningException` prend en charge l'interface `ILearning`. Certaines méthodes dans l'interface nécessitent des implémentations en vue d'émettre une `LearningException` qui est une sous-classe simple de `java.lang.Exception`. Il est fortement recommandé à des fins de débogage que `LearningException` soit construit avec l'exception racine si elle existe.

IScoreOverride

L'interface `IScoreOverride` prend en charge l'interface `ITreatment`. Cette interface vous permet de lire les données définies dans la table de substitution de score ou la table des offres par défaut.

getOfferCode

```
getOfferCode()
```

La méthode `getOfferCode` renvoie la valeur des colonnes de code de l'offre dans la table de substitution de score pour ce membre de référentiel.

Valeur de retour

La méthode `getOfferCode` renvoie un objet `IOfferCode` qui définit la valeur des colonnes de code de l'offre dans la table de substitution de score.

getParameterNames

```
getParameterNames()
```

La méthode `getParameterNames` renvoie la liste des paramètres.

Valeur de retour

La méthode `getParameterNames` renvoie un ensemble définissant la liste des paramètres.

La méthode `IScoreOverride` contient les paramètres suivants. Sauf mention contraire, ces paramètres sont les mêmes que la table de substitution de score.

- ADJ_EXPLORE_SCORE_COLUMN
- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT — Permet de substituer le nombre estimatif actuel (lors du calcul de la pondération de l'offre)

- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN — Permet de créer une expression booléenne pour déterminer l'admissibilité de l'offre
- PREDICATE_SCORE — Permet de créer une expression qui mène à un score numérique
- SCORE_COLUMN
- ZONE_COLUMN

Vous pouvez également référencer une colonne que vous ajoutez à la substitution du score ou de la table des offres par défaut en utilisant le même nom que la colonne.

getValue

```
getValue(String parameterName)
```

La méthode `getValue` renvoie la valeur de la colonne de zone dans la table de substitution de score pour ce membre de référentiel.

- **parameterName** — Chaîne définissant le nom du paramètre dont vous souhaitez connaître la valeur.

Valeur de retour

La méthode `getValue` renvoie un objet contenant la valeur du paramètre demandé.

ISelectionMethod

L'interface `ISelection` indique la méthode utilisée pour arriver à la liste recommandée. La valeur par défaut de l'objet de traitement est EXTERNAL_LEARNING. Il n'est donc pas nécessaire de définir cette valeur. La valeur est stockée dans l'historique détaillé des contacts à des fins de génération de rapports.

Vous pouvez étendre cette interface au-delà des constantes existantes si vous voulez stocker les données en vue d'une analyse ultérieure. Par exemple, vous pouvez créer deux modules d'apprentissage différents et les implémenter sur des groupes de serveurs distincts. Vous pouvez étendre l'interface `ISelection` pour inclure SERVER_GROUP_1 et SERVER_GROUP_2. Vous pouvez alors comparer les résultats de vos deux modules d'apprentissage.

Interface ITreatment

L'interface `ITreatment` prend en charge l'interface `ILearning` comme interface des informations de traitement. Un traitement représente l'offre affectée à une cellule particulière telle qu'elle est définie dans l'environnement de conception. A partir de cette interface, vous pouvez obtenir des informations de cellule et d'offre ainsi que le score de marketing affecté.

getCellCode

```
getCellCode()
```

La méthode getCellCode renvoie le code de cible tel qu'il est défini dans Campaign. La cible est la cible affectée au segment dynamique associé à cette offre.

Valeur de retour

La méthode getCellCode renvoie une chaîne qui définit le code de cible.

getCellId

```
getOfferName()
```

La méthode getCellId renvoie l'ID interne de la cible tel qu'il est défini dans Campaign. La cible est la cible affectée au segment dynamique associé à cette offre.

Valeur de retour

La méthode getCellId renvoie une valeur longue qui définit le ID de cible.

getCellName

```
getCellName()
```

La méthode getCellName renvoie le nom de la cible tel qu'il est défini dans Campaign. La cible est la cible affectée au segment dynamique associé à cette offre.

Valeur de retour

La méthode getCellName renvoie une chaîne qui définit le nom de la cible.

getLearningScore

```
getLearningScore()
```

La méthode getLearningScore renvoie le score de ce traitement. La priorité est la suivante.

1. Renvoie la valeur de substitution, si elle est présente dans la mappe des valeurs de substitution indexée par IScoreoveride.PREDICATE_SCORE_COLUMN
2. Renvoie le score de prédicat si la valeur n'est pas NULL
3. Renvoie la valeur des vendeurs, si elle est présente dans la mappe des valeurs de substitution indexée par IScoreoveride.PREDICATE_SCORE_COLUMN
4. Renvoie la valeur des vendeurs

Valeur de retour

La méthode getLearningScore renvoie un entier qui définit le score déterminé par l'algorithme d'apprentissage.

getMarketerScore

```
getMarketerScore()
```

La méthode getMarketerScore renvoie le score du vendeur défini par le curseur dans l'onglet Stratégie d'interaction de l'offre.

Pour extraire le score d'un vendeur défini par les options avancées de l'onglet Stratégie d'interaction, utilisez getPredicateScore.

Pour extraire le score du vendeur réellement utilisé par le traitement, utilisez `getLearningScore`.

Valeur de retour

La méthode `getMarketerScore` renvoie un entier qui définit le score du vendeur.

getOffer

```
getOffer()
```

La méthode `getOffer` renvoie l'offre de ce traitement.

Valeur de retour

La méthode `getOffer` renvoie un objet `IOffer` qui définit l'offre de ce traitement.

getOverrideValues

```
getOverrideValues()
```

La méthode `getOverrideValues` renvoie les substitutions définies dans les offres par défaut ou la table de substitution des scores.

Valeur de retour

La méthode `getOverrideValues` renvoie un objet `IScoreOverride`.

getPredicate

```
getPredicate()
```

La méthode `getPredicate` renvoie le prédicat défini par la colonne prédicat de la table des offres par défaut, la table de substitution de score ou les options avancées de traitement des règles.

Valeur de retour

La méthode `getPredicate` renvoie une chaîne qui définit le prédicat par la colonne de prédicat de la table des offres par défaut, la table de substitution de score ou les options avancées de traitement des règles.

getPredicateScore

```
getPredicateScore()
```

La méthode `getPredicateScore` renvoie le score défini par la colonne prédicat de la table des offres par défaut, de la table de substitution de score ou des options avancées de traitement des règles.

Valeur de retour

La méthode `getPredicateScore` renvoie un double qui définit le score fixé par la colonne prédicat de la table des offres par défaut, de la table de substitution de score ou des options avancées de traitement des règles.

getScore

```
getScore()
```

La méthode `getScore` renvoie l'un des résultats suivants :

- Le score marketing de l'offre, tel qu'il est défini dans l'onglet Stratégie d'interaction dans Campaign si la propriété `enableScoreOverrideLookup` est définie sur false.
- Le score de l'offre, tel qu'il est défini par `scoreOverrideTable` si la propriété `enableScoreOverrideLookup` est définie sur true.

Valeur de retour

La méthode `getScore` renvoie un entier représentant le score de l'offre.

getTreatmentCode

```
getTreatmentCode()
```

La méthode `getTreatmentCode` renvoie le code de traitement.

Valeur de retour

La méthode `getTreatmentCode` renvoie une chaîne qui définit le code de traitement.

setActualValueUsed

```
setActualValueUsed(string parmName, object value)
```

Utilisez la méthode `setActualValueUsed` pour définir les valeurs qui sont utilisés à différentes étapes de l'exécution de l'algorithme d'apprentissage.

Par exemple, si vous utilisez cette méthode pour écrire dans les tables de l'historique des contacts et des réponses, et modifiez les rapports d'exemple existants, vous pouvez inclure des données de votre algorithme d'apprentissage dans la génération de rapports.

- **parmName** — Chaîne définissant le nom du paramètre que vous définissez.
- **value** — Objet définissant la valeur du paramètre que vous définissez.

Valeur de retour

Aucune.

Exemple d'API d'apprentissage

Cette section contient un exemple d'implémentation d'ILearningInterface. Notez que cette implémentation est un simple exemple et n'est pas conçue pour être utilisée dans un environnement de production.

Cet exemple effectue le suivi des décomptes des acceptations et des contacts et utilise le ratio acceptations/contacts correspondant à une offre particulière pour calculer le taux de probabilité d'acceptation de l'offre. Les offres non présentées bénéficient d'une priorité de recommandation plus élevée. Les offres ayant au moins un contact doivent être classées en fonction du taux décroissant de probabilité d'acceptation.

Dans cet exemple, tous les décomptes sont conservés en mémoire. Il ne s'agit pas d'un scénario réaliste, car le serveur d'exécution n'aura pas assez de mémoire. Dans un scénario de production réel, les décomptes doivent être conservés dans une base de données.

```
package com.unicacorp.interact.samples.learning.v2;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.unicacorp.interact.samples.learning.SampleOptimizer.MyOfferSorter;
import com.unicacorp.interact.treatment.optimization.IClientArgs;
import com.unicacorp.interact.treatment.optimization.IInteractSession;
import com.unicacorp.interact.treatment.optimization.ILearningConfig;
import com.unicacorp.interact.treatment.optimization.ILearningContext;
import com.unicacorp.interact.treatment.optimization.IOffer;
import com.unicacorp.interact.treatment.optimization.LearningException;
import com.unicacorp.interact.treatment.optimization.v2.ILearning;
import com.unicacorp.interact.treatment.optimization.v2.ITreatment;

/**
 * Ceci est un exemple d'implémentation de l'optimiseur d'apprentissage.
 * L'interface ILearning se trouve dans la bibliothèque interact.jar.
 *
 * Pour utiliser cette implémentation, sélectionnez ExternalLearning comme optimizationType dans le noeud offerServing
 * de l'application Interact dans la configuration Platform. Le noeud offerserving comporte aussi
 * une catégorie de configuration External Learning dans laquelle vous devez définir le nom de la classe comme suit :
 * com.unicacorp.interact.samples.learning.v2.SampleLearning.  Veuillez noter toutefois que cette implémentation
 * est un simple exemple n'est pas conçue pour être utilisée dans un environnement de production.
 *
 *
 * Cet exemple effectue le suivi des décomptes des acceptations et des contacts et utilise le ratio acceptations/contacts
 * correspondant à une offre particulière pour calculer le taux de probabilité d'acceptation de l'offre.
 *
 *
 * Les offres non présentées bénéficient d'une priorité de recommandation plus élevée.
 * Les offres ayant au moins un contact doivent être classées en fonction du taux décroissant de probabilité
 * d'acceptation.
 * Remarque : tous les décomptes sont conservés en mémoire. Il ne s'agit pas d'un scénario réaliste car la mémoire
   sera insuffisante
 * à un moment donné.  Dans un scénario de production réel, les décomptes doivent être conservés dans une
 * base de données.
 */

public class SampleLearning implements ILearning
{

    // A map of offer ids to contact count for the offer id
    private Map<Long,Integer> _offerToContactCount = new HashMap<Long, Integer>();

    // A map of offer ids to contact count for the offer id
    private Map<Long,Integer> _offerToAcceptCount = new HashMap<Long, Integer>();


    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#initialize
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void initialize(ILearningConfig config, boolean debug) throws LearningException
    {
        // If any remote connections are required, this is a good place to initialize those connections as this
        // method is called once at the start of the interact runtime webapp.
        // This example does not have any remote connections and prints for debugging purposes that this method will
        // be called
        System.out.println("Calling initialize for SampleLearning");
    }

    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#reinitialize
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void reinitialize(ILearningConfig config, boolean debug) throws LearningException
    {
        // If an IC is deployed, this reinitialize method is called to allow the implementation to
```

```
    // refresh any updated configuration settings
    System.out.println("Calling reinitialize for SampleLearning");
}


/* (non-Javadoc)
 * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#logEvent
 *   (com.unicacorp.interact.treatment.optimization.v2.ILearningContext,
 *   com.unicacorp.interact.treatment.optimization.v2.IOffer,
 *   com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
 *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
 */
public void logEvent(ILearningContext context, IOffer offer, IClientArgs clientArgs,
IInteractSession session, boolean debug) throws LearningException
{
    System.out.println("Calling logEvent for SampleLearning");


    if(context.getLearningContext()==ILearningContext.LOG_AS_CONTACT)
    {
        System.out.println("adding contact");

        // Keep track of all contacts in memory
        synchronized(_offerToAcceptCount)
        {
           Integer count = _offerToAcceptCount.get(offer.getOfferId());
           if(count == null)
               count = new Integer(1);
           else
               count++;
           _offerToAcceptCount.put(offer.getOfferId(), ++count);
        }

    }
    else if(context.getLearningContext()==ILearningContext.LOG_AS_ACCEPT)
    {
        System.out.println("adding accept");
        // Keep track of all accept counts in memory by adding to the map
        synchronized(_offerToAcceptCount)
        {
           Integer count = _offerToAcceptCount.get(offer.getOfferId());
           if(count == null)
               count = new Integer(1);
           else
               count++;
           _offerToAcceptCount.put(offer.getOfferId(), ++count);
        }
        }

}

/* (non-Javadoc)
 * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#optimizeRecommendList
 *   (java.util.List, com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
 *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
 */
public List<ITreatment> optimizeRecommendList(List<ITreatment> recList,
    IClientArgs clientArgs, IInteractSession session, boolean debug)
    throws LearningException
{
    System.out.println("Calling optimizeRecommendList for SampleLearning");

    // Sort the candidate treatments by calling the sorter defined in this class and return the sorted list
    Collections.sort(recList,new MyOfferSorter());

    // now just return what was asked for via "numberRequested" variable
    List<ITreatment> result = new ArrayList<ITreatment>();

    for(int x=0;x<(Integer)clientArgs.getValue(IClientArgs.NUMBER_OF_OFFERS_REQUESTED) && x<recList.size();x++)
    {
        result.add(recList.get(x));
    }
    return result;
}

/* (non-Javadoc)
 * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#shutdown
 *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
 */
```

```
    public void shutdown(ILearningConfig config, boolean debug) throws LearningException
    {
        // If any remote connections exist, this would be a good place to gracefully
        // disconnect from them as this method is called at the shutdown of the Interact runtime
        // webapp.   For this example, there is nothing really to do
        // except print out a statement for debugging.
        System.out.println("Calling shutdown for SampleLearning");

    }
    // Sort by:
    // 1. offers with zero contacts  - for ties, order is based on original input
    // 2. descending accept probability rate - for ties, order is based on original input

    public class MyOfferSorter implements Comparator<ITreatment>
    {
        private static final long serialVersionUID = 1L;

        /* (non-Javadoc)
         * @see java.lang.Comparable#compareTo(java.lang.Object)
         */
        public int compare(ITreatment treatment1, ITreatment treatment2)
        {

            // get contact count for both treatments
            Integer contactCount1 = _offerToContactCount.get(treatment1.getOffer().getOfferId());
            Integer contactCount2 = _offerToContactCount.get(treatment2.getOffer().getOfferId());

            // if treatment hasn't been contacted, then that wins
            if(contactCount1 == null || contactCount1 == 0)
                return -1;

            if(contactCount2 == null || contactCount2 == 0)
                return 1;

            // get accept counts
            Integer acceptCount1 = _offerToAcceptCount.get(treatment1.getOffer().getOfferId());
            Integer acceptCount2 = _offerToAcceptCount.get(treatment2.getOffer().getOfferId());

            float acceptProbability1 = (float) acceptCount1 / (float) contactCount1;
            float acceptProbability2 = (float) acceptCount2 / (float) contactCount2;

            // descending order
            return (int) (acceptProbability2 - acceptProbability1);

        }
    }

}
```

Annexe A. IBM Unica Interact WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://soap.api.interact.unicacorp.com" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" bloop="http://api.interact.unicacorp.com/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" targetNamespace="http://soap.api.interact.unicacorp.com">
 <wsdl:types>
  <xs:schema xmlns:ns="http://soap.api.interact.unicacorp.com" attributeFormDefault="qualified"
   elementFormDefault="qualified" targetNamespace="http://soap.api.interact.unicacorp.com">
   <xs:element name="executeBatch">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="commands" nillable="false" type="ns1:CommandImpl"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="executeBatchResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:BatchResponse"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="endSession">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="endSessionResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getOffers">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
      <xs:element minOccurs="1" name="iPoint" nillable="false" type="xs:string"/>
      <xs:element minOccurs="1" name="numberRequested" type="xs:int"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getOffersResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getProfile">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getProfileResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getVersionResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
     </xs:sequence>
```

© Copyright IBM Corp. 2001, 2012

```
  </xs:complexType>
 </xs:element>
 <xs:element name="postEvent">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
    <xs:element minOccurs="1" name="eventName" nillable="false" type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="eventParameters"
     nillable="true" type="ns1:NameValuePairImpl"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="postEventResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="setAudience">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID" nillable="false" type="ns1:NameValuePairImpl"/>
    <xs:element minOccurs="1" name="audienceLevel" nillable="false" type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="parameters" nillable="true" type="ns1:NameValuePairImpl"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="setAudienceResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="setDebug">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
    <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="setDebugResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="startSession">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
    <xs:element minOccurs="1" name="relyOnExistingSession" type="xs:boolean"/>
    <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
    <xs:element minOccurs="1" name="interactiveChannel" nillable="false" type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID" nillable="false" type="ns1:NameValuePairImpl"/>
    <xs:element minOccurs="1" name="audienceLevel" nillable="false" type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="parameters" nillable="true" type="ns1:NameValuePairImpl"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="startSessionResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
<xs:schema xmlns:ax21="http://api.interact.unicacorp.com/xsd" attributeFormDefault="qualified"
 elementFormDefault="qualified" targetNamespace="http://api.interact.unicacorp.com/xsd">
 <xs:complexType name="Command">
  <xs:sequence>
   <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID" nillable="true" type="ax21:NameValuePair"/>
   <xs:element minOccurs="1" name="audienceLevel" nillable="true" type="xs:string"/>
   <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
```

```
  <xs:element minOccurs="1" name="event" nillable="true" type="xs:string"/>
  <xs:element maxOccurs="unbounded" minOccurs="1" name="eventParameters" nillable="true" type="ax21:NameValuePair"/>
  <xs:element minOccurs="1" name="interactionPoint" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="interactiveChannel" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="methodIdentifier" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="numberRequested" type="xs:int"/>
  <xs:element minOccurs="1" name="relyOnExistingSession" type="xs:boolean"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="NameValuePair">
 <xs:sequence>
  <xs:element minOccurs="1" name="name" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="valueAsDate" nillable="true" type="xs:dateTime"/>
  <xs:element minOccurs="1" name="valueAsNumeric" nillable="true" type="xs:double"/>
  <xs:element minOccurs="1" name="valueAsString" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="valueDataType" nillable="true" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="CommandImpl">
 <xs:sequence>
  <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID" nillable="true" type="ax21:NameValuePairImpl"/>
  <xs:element minOccurs="1" name="audienceLevel" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
  <xs:element minOccurs="1" name="event" nillable="true" type="xs:string"/>
  <xs:element maxOccurs="unbounded" minOccurs="1" name="eventParameters" nillable="true" type="ax21:NameValuePairImpl"/>
  <xs:element minOccurs="1" name="interactionPoint" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="interactiveChannel" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="methodIdentifier" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="numberRequested" type="xs:int"/>
  <xs:element minOccurs="1" name="relyOnExistingSession" type="xs:boolean"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="NameValuePairImpl">
 <xs:sequence>
  <xs:element minOccurs="1" name="name" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="valueAsDate" nillable="true" type="xs:dateTime"/>
  <xs:element minOccurs="1" name="valueAsNumeric" nillable="true" type="xs:double"/>
  <xs:element minOccurs="1" name="valueAsString" nillable="true" type="xs:string"/>
  <xs:element minOccurs="1" name="valueDataType" nillable="true" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="BatchResponse">
 <xs:sequence>
  <xs:element minOccurs="0" name="batchStatusCode" type="xs:int"/>
  <xs:element maxOccurs="unbounded" minOccurs="0" name="responses" nillable="false" type="ax21:Response"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="Response">
 <xs:sequence>
  <xs:element maxOccurs="unbounded" minOccurs="0" name="advisoryMessages" nillable="true" type="ax21:AdvisoryMessage"/>
  <xs:element minOccurs="0" name="apiVersion" nillable="false" type="xs:string"/>
  <xs:element minOccurs="0" name="offerList" nillable="true" type="ax21:OfferList"/>
  <xs:element maxOccurs="unbounded" minOccurs="0" name="profileRecord" nillable="true" type="ax21:NameValuePair"/>
  <xs:element minOccurs="0" name="sessionID" nillable="true" type="xs:string"/>
  <xs:element minOccurs="0" name="statusCode" type="xs:int"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="AdvisoryMessage">
 <xs:sequence>
  <xs:element minOccurs="0" name="detailMessage" nillable="true" type="xs:string"/>
  <xs:element minOccurs="0" name="message" nillable="true" type="xs:string"/>
  <xs:element minOccurs="0" name="messageCode" type="xs:int"/>
  <xs:element minOccurs="0" name="statusLevel" type="xs:int"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="OfferList">
 <xs:sequence>
  <xs:element minOccurs="0" name="defaultString" nillable="true" type="xs:string"/>
  <xs:element maxOccurs="unbounded" minOccurs="0" name="recommendedOffers" nillable="true" type="ax21:Offer"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="Offer">
 <xs:sequence>
  <xs:element maxOccurs="unbounded" minOccurs="0" name="additionalAttributes" nillable="true" type="ax21:NameValuePair"/>
  <xs:element minOccurs="0" name="description" nillable="true" type="xs:string"/>
  <xs:element maxOccurs="unbounded" minOccurs="0" name="offerCode" nillable="true" type="xs:string"/>
  <xs:element minOccurs="0" name="offerName" nillable="true" type="xs:string"/>
  <xs:element minOccurs="0" name="score" type="xs:int"/>
  <xs:element minOccurs="0" name="treatmentCode" nillable="true" type="xs:string"/>
```

```
   </xs:sequence>
  </xs:complexType>
 </xs:schema>
</wsdl:types>
<wsdl:message name="setAudienceRequest">
 <wsdl:part name="parameters" element="ns0:setAudience"/>
</wsdl:message>
<wsdl:message name="setAudienceResponse">
 <wsdl:part name="parameters" element="ns0:setAudienceResponse"/>
</wsdl:message>
<wsdl:message name="postEventRequest">
 <wsdl:part name="parameters" element="ns0:postEvent"/>
</wsdl:message>
<wsdl:message name="postEventResponse">
 <wsdl:part name="parameters" element="ns0:postEventResponse"/>
</wsdl:message>
<wsdl:message name="getOffersRequest">
 <wsdl:part name="parameters" element="ns0:getOffers"/>
</wsdl:message>
<wsdl:message name="getOffersResponse">
 <wsdl:part name="parameters" element="ns0:getOffersResponse"/>
</wsdl:message>
<wsdl:message name="startSessionRequest">
 <wsdl:part name="parameters" element="ns0:startSession"/>
</wsdl:message>
<wsdl:message name="startSessionResponse">
 <wsdl:part name="parameters" element="ns0:startSessionResponse"/>
</wsdl:message>
<wsdl:message name="getVersionRequest"/>
<wsdl:message name="getVersionResponse">
 <wsdl:part name="parameters" element="ns0:getVersionResponse"/>
</wsdl:message>
<wsdl:message name="setDebugRequest">
 <wsdl:part name="parameters" element="ns0:setDebug"/>
</wsdl:message>
<wsdl:message name="setDebugResponse">
 <wsdl:part name="parameters" element="ns0:setDebugResponse"/>
</wsdl:message>
<wsdl:message name="executeBatchRequest">
 <wsdl:part name="parameters" element="ns0:executeBatch"/>
</wsdl:message>
<wsdl:message name="executeBatchResponse">
 <wsdl:part name="parameters" element="ns0:executeBatchResponse"/>
</wsdl:message>
<wsdl:message name="getProfileRequest">
 <wsdl:part name="parameters" element="ns0:getProfile"/>
</wsdl:message>
<wsdl:message name="getProfileResponse">
 <wsdl:part name="parameters" element="ns0:getProfileResponse"/>
</wsdl:message>
<wsdl:message name="endSessionRequest">
 <wsdl:part name="parameters" element="ns0:endSession"/>
</wsdl:message>
<wsdl:message name="endSessionResponse">
 <wsdl:part name="parameters" element="ns0:endSessionResponse"/>
</wsdl:message>
<wsdl:portType name="InteractServicePortType">
 <wsdl:operation name="setAudience">
  <wsdl:input message="ns0:setAudienceRequest" wsaw:Action="urn:setAudience"/>
  <wsdl:output message="ns0:setAudienceResponse" wsaw:Action="urn:setAudienceResponse"/>
 </wsdl:operation>
 <wsdl:operation name="postEvent">
  <wsdl:input message="ns0:postEventRequest" wsaw:Action="urn:postEvent"/>
  <wsdl:output message="ns0:postEventResponse" wsaw:Action="urn:postEventResponse"/>
 </wsdl:operation>
 <wsdl:operation name="getOffers">
  <wsdl:input message="ns0:getOffersRequest" wsaw:Action="urn:getOffers"/>
  <wsdl:output message="ns0:getOffersResponse" wsaw:Action="urn:getOffersResponse"/>
 </wsdl:operation>
 <wsdl:operation name="startSession">
  <wsdl:input message="ns0:startSessionRequest" wsaw:Action="urn:startSession"/>
  <wsdl:output message="ns0:startSessionResponse" wsaw:Action="urn:startSessionResponse"/>
 </wsdl:operation>
 <wsdl:operation name="getVersion">
  <wsdl:input message="ns0:getVersionRequest" wsaw:Action="urn:getVersion"/>
  <wsdl:output message="ns0:getVersionResponse" wsaw:Action="urn:getVersionResponse"/>
 </wsdl:operation>
 <wsdl:operation name="setDebug">
  <wsdl:input message="ns0:setDebugRequest" wsaw:Action="urn:setDebug"/>
```

```
  <wsdl:output message="ns0:setDebugResponse" wsaw:Action="urn:setDebugResponse"/>
 </wsdl:operation>
 <wsdl:operation name="executeBatch">
  <wsdl:input message="ns0:executeBatchRequest" wsaw:Action="urn:executeBatch"/>
  <wsdl:output message="ns0:executeBatchResponse" wsaw:Action="urn:executeBatchResponse"/>
 </wsdl:operation>
 <wsdl:operation name="getProfile">
  <wsdl:input message="ns0:getProfileRequest" wsaw:Action="urn:getProfile"/>
  <wsdl:output message="ns0:getProfileResponse" wsaw:Action="urn:getProfileResponse"/>
 </wsdl:operation>
 <wsdl:operation name="endSession">
  <wsdl:input message="ns0:endSessionRequest" wsaw:Action="urn:endSession"/>
  <wsdl:output message="ns0:endSessionResponse" wsaw:Action="urn:endSessionResponse"/>
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="InteractServiceSOAP11Binding" type="ns0:InteractServicePortType">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <wsdl:operation name="setAudience">
  <soap:operation soapAction="urn:setAudience" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="postEvent">
  <soap:operation soapAction="urn:postEvent" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="getOffers">
  <soap:operation soapAction="urn:getOffers" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="startSession">
  <soap:operation soapAction="urn:startSession" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="getVersion">
  <soap:operation soapAction="urn:getVersion" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="setDebug">
  <soap:operation soapAction="urn:setDebug" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="executeBatch">
  <soap:operation soapAction="urn:executeBatch" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
```

```
 </wsdl:operation>
 <wsdl:operation name="getProfile">
  <soap:operation soapAction="urn:getProfile" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="endSession">
  <soap:operation soapAction="urn:endSession" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="InteractServiceSOAP12Binding" type="ns0:InteractServicePortType">
 <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
 <wsdl:operation name="setAudience">
  <soap12:operation soapAction="urn:setAudience" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap12:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="postEvent">
  <soap12:operation soapAction="urn:postEvent" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap12:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="getOffers">
  <soap12:operation soapAction="urn:getOffers" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap12:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="startSession">
  <soap12:operation soapAction="urn:startSession" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap12:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="getVersion">
  <soap12:operation soapAction="urn:getVersion" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap12:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="setDebug">
  <soap12:operation soapAction="urn:setDebug" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap12:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="executeBatch">
  <soap12:operation soapAction="urn:executeBatch" style="document"/>
  <wsdl:input>
```

```
    <soap12:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap12:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getProfile">
   <soap12:operation soapAction="urn:getProfile" style="document"/>
   <wsdl:input>
    <soap12:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap12:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="endSession">
   <soap12:operation soapAction="urn:endSession" style="document"/>
   <wsdl:input>
    <soap12:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap12:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:binding name="InteractServiceHttpBinding" type="ns0:InteractServicePortType">
  <http:binding verb="POST"/>
  <wsdl:operation name="setAudience">
   <http:operation location="InteractService/setAudience"/>
   <wsdl:input>
    <mime:content part="setAudience" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="setAudience" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="postEvent">
   <http:operation location="InteractService/postEvent"/>
   <wsdl:input>
    <mime:content part="postEvent" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="postEvent" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getOffers">
   <http:operation location="InteractService/getOffers"/>
   <wsdl:input>
    <mime:content part="getOffers" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="getOffers" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="startSession">
   <http:operation location="InteractService/startSession"/>
   <wsdl:input>
    <mime:content part="startSession" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="startSession" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getVersion">
   <http:operation location="InteractService/getVersion"/>
   <wsdl:input>
    <mime:content part="getVersion" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="getVersion" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="setDebug">
   <http:operation location="InteractService/setDebug"/>
   <wsdl:input>
    <mime:content part="setDebug" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="setDebug" type="text/xml"/>
```

```
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="executeBatch">
   <http:operation location="InteractService/executeBatch"/>
   <wsdl:input>
    <mime:content part="executeBatch" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="executeBatch" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getProfile">
   <http:operation location="InteractService/getProfile"/>
   <wsdl:input>
    <mime:content part="getProfile" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="getProfile" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="endSession">
   <http:operation location="InteractService/endSession"/>
   <wsdl:input>
    <mime:content part="endSession" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="endSession" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="InteractService">
  <wsdl:port name="InteractServiceSOAP11port_http" binding="ns0:InteractServiceSOAP11Binding">
   <soap:address location="http://localhost:7001/interact/services/InteractService"/>
  </wsdl:port>
  <wsdl:port name="InteractServiceSOAP12port_http" binding="ns0:InteractServiceSOAP12Binding">
   <soap12:address location="http://localhost:7001/interact/services/InteractService"/>
  </wsdl:port>
  <wsdl:port name="InteractServiceHttpport" binding="ns0:InteractServiceHttpBinding">
   <http:address location="http://localhost:7001/interact/services/InteractService"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Annexe B. Interact propriétés de configuration de l'environnement d'exécution

Cette section décrit toutes les propriétés de configuration de l'environnement d'exécution d'Interact.

Interact | general

Ces propriétés de configuration définissent les paramètres généraux de votre environnement d'exécution, notamment le niveau de traces par défaut et les paramètres régionaux.

log4jConfig

Description

Emplacement du fichier qui contient les propriétés log4j. Ce chemin doit se rapporter à la variable d'environnement `INTERACT_HOME`. `INTERACT_HOME` est l'emplacement du répertoire d'installation d'Interact.

Valeur par défaut

`./conf/interact_log4j.properties`

asmUserForDefaultLocale

Description

La propriété `asmUserForDefaultLocale` définit l'utilisateur IBM Unica Marketing duquel Interact dérive ses paramètres locaux.

Les paramètres régionaux définissent la langue d'affichage de la conception et celle des messages de conseils dans lesquels se trouvent les API Interact. Si les paramètres locaux ne correspondent pas à ceux du système d'exploitation de vos machines, Interact continuera de s'exécuter, mais il se peut que la langue d'affichage de la conception et des messages de conseils soit différente.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | general | learningTablesDataSource

Ces propriétés de configuration définissent les paramètres de la source de données pour les tables d'apprentissage intégrées. Vous devez définir cette source de données si vous utilisez l'apprentissage intégré d'Interact.

Si vous créez votre propre implémentation d'apprentissage via l'API d'apprentissage, vous pouvez configurer votre implémentation d'apprentissage personnalisée pour lire ces valeurs à l'aide de l'interface `ILearningConfig`.

jndiName

Description

© Copyright IBM Corp. 2001, 2012

Utilisez la propriété jndiName pour identifier les sources de données JNDI (Java Naming and Directory Interface) définies sur le serveur d'applications (WebSphere ou WebLogic) pour les tables d'apprentissage auxquelles les serveurs d'exécution d'Interact accèdent.

Les tables d'apprentissage sont créées par le fichier ddl aci_lrntab et comportent les tables suivantes (entre autres) : UACI_AttributeValue et UACI_OfferStats.

Valeur par défaut

Aucune valeur par défaut définie.

type

Description

Type de la base de données associée à la source de données utilisée par les tables d'apprentissage auxquelles les serveurs d'exécution d'Interact accèdent.

Les tables d'apprentissage sont créées par le fichier ddl aci_lrntab et comportent les tables suivantes (entre autres) : UACI_AttributeValue et UACI_OfferStats.

Valeur par défaut

SQLServer

Valeurs valides

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Description

La propriété ConnectionRetryPeriod spécifie la durée, en secondes, pendant laquelle Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables d'apprentissage. Interact essaie automatiquement de se reconnecter à la base de données pendant ce laps de temps avant de signaler une erreur ou un échec au niveau de la base de données. Si la valeur est définie sur 0, Interact essaiera indéfiniment ; si elle est définie sur -1, il n'y aura aucune tentative.

Les tables d'apprentissage sont créées par le fichier ddl aci_lrntab et comportent les tables suivantes (entre autres) : UACI_AttributeValue et UACI_OfferStats.

Valeur par défaut

-1

connectionRetryDelay

Description

La propriété ConnectionRetryDelay spécifie la durée, en secondes, pendant laquelle Interact attend avant toute tentative de reconnexion à la base de données pour les tables d'apprentissage après une défaillance. Si la valeur est définie sur -1, il n'y aura pas de tentative.

Les tables d'apprentissage sont créées par le fichier ddl `aci_lrntab` et comportent les tables suivantes (entre autres) : `UACI_AttributeValue` et `UACI_OfferStats`.

Valeur par défaut

-1

schéma

Description

Nom du schéma qui contient les tables associées au module d'apprentissage intégré. Interact insère la valeur de cette propriété devant tous les noms de table. Par exemple, `UACI_IntChannel` devient `schema.UACI_IntChannel`.

Il n'est pas nécessaire de définir un schéma. Si vous ne définissez pas de schéma, Interact part du principe que le propriétaire des tables est le même que celui du schéma. Vous devez définir cette valeur pour supprimer toute ambiguïté.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | general | prodUserDataSource

Ces propriétés de configuration définissent les paramètres de la source de données pour les tables de profils de production. Vous devez définir cette source de données. Il s'agit de la source de données référencée par l'environnement d'exécution lorsque des diagrammes interactifs sont exécutés après le déploiement.

jndiName

Description

Utilisez la propriété `jndiName` pour identifier les sources de données JNDI (Java Naming and Directory Interface) définies sur le serveur d'applications (WebSphere ou WebLogic) pour les tables client auxquelles les serveurs d'exécution d'Interact accèdent.

Valeur par défaut

Aucune valeur par défaut définie.

type

Description

Type de la base de données pour les tables client accessibles via les serveurs d'exécution d'Interact.

Valeur par défaut

SQLServer

Valeurs valides

SQLServer | DB2 | ORACLE

aliasPrefix

Description

La propriété AliasPrefix spécifie la manière dont Interact génère le nom d'alias qu'Interact crée automatiquement en cas d'utilisation d'une table de dimension et d'écriture dans une nouvelle table accessible par les serveurs d'exécution d'Interact.

Notez que chaque base de données dispose d'une longueur d'identifiant maximale. Vérifiez la documentation associée à la base de données utilisée et assurez-vous que la valeur définie ne dépasse pas la longueur d'identifiant maximale de votre base.

Valeur par défaut

A

connectionRetryPeriod

Description

La propriété ConnectionRetryPeriod spécifie la durée, en secondes, pendant laquelle Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables client d'exécution. Interact essaie automatiquement de se reconnecter à la base de données pendant ce laps de temps avant de signaler une erreur ou un échec au niveau de la base de données. Si la valeur est définie sur 0, Interact essaiera indéfiniment ; si elle est définie sur -1, il n'y aura aucune tentative.

Valeur par défaut

-1

connectionRetryDelay

Description

La propriété ConnectionRetryDelay spécifie la durée, en secondes, pendant laquelle Interact attend avant toute tentative de reconnexion à la base de données pour les tables client de l'environnement d'exécution d'Interact après une défaillance. Si la valeur est définie sur -1, il n'y aura pas de tentative.

Valeur par défaut

-1

schema

Description

Nom du schéma qui contient vos tables de données de profil. Interact insère la valeur de cette propriété devant tous les noms de table. Par exemple, UACI_IntChannel devient schema.UACI_IntChannel.

Il n'est pas nécessaire de définir un schéma. Si vous ne définissez pas de schéma, Interact part du principe que le propriétaire des tables est le même que celui du schéma. Vous devez définir cette valeur pour supprimer toute ambiguïté.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | general | systemTablesDataSource

Ces propriétés de configuration définissent les paramètres de la source de données pour les tables système de l'environnement d'exécution. Vous devez définir cette source de données.

jndiName

Description

Utilisez la propriété `jndiName` pour identifier la source de données Java Naming and Directory Interface (JNDI) définie sur le serveur d'application (WebSphere ou WebLogic) pour les tables de l'environnement d'exécution.

La base de données de l'environnement d'exécution est renseignée avec les scripts dll `aci_runtime` et `aci_populate_runtime`. Elle contient également les tables suivantes (entre autres) : `UACI_CHOfferAttrib` et `UACI_DefaultedStat`.

Valeur par défaut

Aucune valeur par défaut définie.

type

Description

Type de la base de données pour les tables système de l'environnement d'exécution.

La base de données de l'environnement d'exécution est renseignée avec les scripts dll `aci_runtime` et `aci_populate_runtime`. Elle contient également les tables suivantes (entre autres) : `UACI_CHOfferAttrib` et `UACI_DefaultedStat`.

Valeur par défaut

SQLServer

Valeurs valides

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables système d'exécution. Interact essaie automatiquement de se reconnecter à la base de données pendant ce laps de temps avant de signaler une erreur ou un échec au niveau de la base de données. Si la valeur est définie sur 0, Interact essaiera indéfiniment ; si elle est définie sur -1, il n'y aura aucune tentative.

La base de données de l'environnement d'exécution est renseignée avec les scripts dll `aci_runtime` et `aci_populate_runtime`. Elle contient également les tables suivantes (entre autres) : `UACI_CHOfferAttrib` et `UACI_DefaultedStat`.

Valeur par défaut

-1

connectionRetryDelay

Description

La propriété ConnectionRetryDelay spécifie la durée, en secondes, pendant laquelle Interact attend avant toute tentative de reconnexion à la base de données pour les tables système de l'environnement d'exécution d'Interact après une défaillance. Si la valeur est définie sur -1, il n'y aura pas de tentative.

La base de données de l'environnement d'exécution est renseignée avec les scripts dll aci_runtime et aci_populate_runtime. Elle contient également les tables suivantes (entre autres) : UACI_CHOfferAttrib et UACI_DefaultedStat.

Valeur par défaut

-1

schema

Description

Nom du schéma qui contient les tables destinées à l'environnement d'exécution. Interact insère la valeur de cette propriété devant tous les noms de table. Par exemple, UACI_IntChannel devient schema.UACI_IntChannel.

Il n'est pas nécessaire de définir un schéma. Si vous ne définissez pas de schéma, Interact part du principe que le propriétaire des tables est le même que celui du schéma. Vous devez définir cette valeur pour supprimer toute ambiguïté.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | general | systemTablesDataSource | loaderProperties

Ces propriétés de configuration définissent les paramètres de l'utilitaire de chargement de base de données pour les tables système de l'environnement d'exécution. Vous devez définir ces propriétés uniquement si vous utilisez un utilitaire de chargement.

databaseName

Description

Nom de la base de données à laquelle l'utilitaire se connecte.

Valeur par défaut

Aucune valeur par défaut définie.

LoaderCommandForAppend

Description

Le paramètre LoaderCommandForAppend spécifie la commande émise pour appeler votre utilitaire de chargement de base de données afin d'ajouter des enregistrements aux tables de intermédiaires de l'historique des contacts et des réponses dans Interact. Vous devez définir ce paramètre pour que l'utilitaire de chargement puisse prendre en charge les données d'historique des réponses et des contacts.

Ce paramètre est spécifié par le nom de chemin complet de l'exécutable d'un utilitaire de chargement ou de celui du script de lancement d'un tel utilitaire. L'utilisation d'un script vous permet d'effectuer d'autres opérations de configuration avant d'appeler l'utilitaire de chargement.

La plupart des utilitaires de chargement nécessitent plusieurs arguments afin d'être lancés. Ils peuvent notamment inclure la spécification des fichiers de données et de contrôle (qui forment la base du chargement) ainsi que la base de données et la table de destination du chargement. Les jetons sont remplacés par les éléments spécifiés lorsque la commande est exécutée.

Consultez la documentation associée à l'utilitaire de chargement de votre base de données pour voir la syntaxe à utiliser lorsqu'il est appelé.

Par défaut, ce paramètre n'est pas défini.

Les marques disponibles pour `LoaderCommandForAppend` sont décrites dans le tableau suivant :

Jeton	**Description**
<CONTROLFILE>	Cette marque est remplacée par le chemin et le nom de fichier complets du fichier de contrôle temporaire généré par Interact conformément au modèle spécifié dans le paramètre `LoaderControlFileTemplate`.
<DATABASE>	Ce jeton est remplacé par le nom de la source de données dans laquelle Interact charge des données. Le nom de la source de données est le même que celui appliqué à la catégorie de cette source de données.
<DATAFILE>	Ce jeton est remplacé par le chemin d'accès complet et le nom de fichier vers le fichier de données temporaires créé par Interact pendant le processus de chargement. Ce fichier se trouve dans `UNICA_ACTMPDIR`, le répertoire temporaire de Interact.
<DBCOLUMNNUMBER>	Ce jeton est remplacé par l'ordinal de colonne de la base de données.
<FIELDLENGTH>	Ce jeton est remplacé par la longueur du champ en cours de chargement dans la base de données.
<FIELDNAME>	Ce jeton est remplacé par le nom du champ en cours de chargement dans la base de données.
<FIELDNUMBER>	Ce jeton est remplacé par le numéro du champ en cours de chargement dans la base de données.

Jeton	Description
<FIELDTYPE>	Ce jeton est remplacé par le libellé "CHAR( )". La longueur de ce champ est spécifiée entre les parenthèses (). Si votre base de données ne comprend pas le type du champ, CHAR, vous pouvez spécifier manuellement le texte approprié pour le type et utiliser le jeton <FIELDLENGTH>. Par exemple, pour SQLSVR et SQL2000, vous devriez utiliser "SQLCHAR(<FIELDLENGTH>)".
<NATIVETYPE>	Ce jeton est remplacé par le type de base de données dans laquelle ce champ a été chargé.
<NUMFIELDS>	Ce jeton est remplacé par le nombre de champs contenus dans la table.
<PASSWORD>	Ce jeton est remplacé par le mot de passe de la base de données utilisé lors de la connexion du diagramme actuel à la source de données.
<TABLENAME>	Ce jeton est remplacé par le nom de la table de la base de données dans laquelle Interact charge des données.
<USER>	Ce jeton est remplacé par le nom d'utilisateur de la base de données utilisé lors de la connexion du diagramme actuel à la source de données.

Valeur par défaut

Aucune valeur par défaut définie.

LoaderControlFileTemplateForAppend

Description

La propriété `LoaderControlFileTemplateForAppend` indique le chemin complet et le nom du modèle de fichier de contrôle préalablement configuré dans Interact. Si ce paramètre est configuré, Interact construit dynamiquement un fichier contrôle temporaire basé sur le modèle qui est spécifié ici. Le chemin et le nom de ce fichier de contrôle temporaire sont associés au jeton `<CONTROLFILE>`, lui-même associé à la propriété `LoaderCommandForAppend`.

Avant d'utiliser Interact en mode utilitaire de chargement de base de données, vous devez configurer le modèle de fichier contrôle qui est spécifié par ce paramètre. Le modèle de fichier contrôle prend en charge les jetons suivants, qui sont dynamiquement remplacés à la création du fichier contrôle temporaire par Interact.

Pour vérifier la syntaxe requise pour le fichier de contrôle, veuillez consulter la documentation relative à l'utilitaire de chargement de votre

base de données. Les marques associées à votre modèle de fichier de contrôle sont les mêmes que celles associées à la propriété LoaderControlFileTemplate.

Par défaut, ce paramètre n'est pas défini.

Valeur par défaut

Aucune valeur par défaut définie.

LoaderDelimiterForAppend

Description

La propriété LoaderDelimiterForAppend indique si le fichier de données temporaire de Interact est un fichier plat délimité ou de longueur fixe, ainsi que, s'il est délimité, le ou les caractères délimiteurs utilisés.

Si la valeur n'est pas définie, Interact crée le fichier de données temporaire sous la forme d'un fichier à plat de largeur fixe.

Si vous spécifiez une valeur, celle-ci est utilisée lorsque l'utilitaire de chargement est appelé pour renseigner une table potentiellement renseignée. Interact crée le fichier de données temporaire sous la forme d'un fichier plat délimité et utilise la valeur de cette propriété en tant que délimiteur.

Par défaut, cette propriété n'est pas définie.

Valeur par défaut

Valeurs valides

Tout caractère (entouré de guillemets si vous le souhaitez).

LoaderDelimiterAtEndForAppend

Description

Pour certains utilitaires de chargement externes, le fichier de données doit être délimité et un délimiteur doit être présent à chaque fin de ligne. Pour satisfaire cette condition, définissez la valeur LoaderDelimiterAtEndForAppend sur TRUE. Ainsi, lorsque l'utilitaire de chargement est appelé pour renseigner une table potentiellement renseignée, Interact applique des délimiteurs à la fin de chaque ligne.

Valeur par défaut

FALSE

Valeurs valides

TRUE | FALSE

LoaderUseLocaleDP

Description

La propriété LoaderUseLocaleDP indique, lorsque Interact écrit des valeurs numériques sur des fichiers qui doivent être chargés par un utilitaire de chargement, si le symbole spécifique à chaque région est utilisé en tant que point décimal.

Définissez cette valeur sur FALSE pour indiquer que le point (.) est utilisé en tant que point décimal.

Définissez-la sur TRUE pour indiquer que le symbole propre à votre région est utilisé.

Valeur par défaut

FALSE

Valeurs valides

TRUE | FALSE

Interact | general | testRunDataSource

Ces propriétés de configuration définissent les paramètres de la source de données pour les tables d'exécution en mode test pour l'environnement de conception Interact. Vous devez définir cette source de données pour au moins l'un de vos environnements d'exécution. Ces tables sont utilisées lorsque vous exécutez un diagramme temps réel en mode test.

jndiName

Description

Utilisez la propriété jndiName pour identifier la source de données Java Naming and Directory Interface (JNDI) définie sur le serveur d'application (WebSphere or WebLogic) et associée aux tables client accessibles à l'environnement de conception lors de l'exécution de diagrammes temps réel en mode test.

Valeur par défaut

Aucune valeur par défaut définie.

type

Description

Type de la base de données associée aux tables client accessibles à l'environnement de conception lors de l'exécution de diagrammes temps réel en mode test.

Valeur par défaut

SQLServer

Valeurs valides

SQLServer | DB2 | ORACLE

aliasPrefix

Description

La propriété AliasPrefix spécifie la manière dont Interact génère le nom d'alias que Interact crée automatiquement en cas d'utilisation d'une table de dimension et d'écriture dans une nouvelle table pour les tables client accessibles par l'environnement de conception quand il exécute des tests de diagrammes interactifs.

Notez que chaque base de données dispose d'une longueur d'identifiant maximale. Vérifiez la documentation associée à la base de données utilisée et assurez-vous que la valeur définie ne dépasse pas la longueur d'identifiant maximale de votre base.

Valeur par défaut

A

connectionRetryPeriod

Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables d'exécution en mode test. Interact essaie automatiquement de se reconnecter à la base de données pendant ce laps de temps avant de signaler une erreur ou un échec au niveau de la base de données. Si la valeur est définie sur 0, Interact essaiera indéfiniment ; si elle est définie sur -1, il n'y aura aucune tentative.

Valeur par défaut

-1

connectionRetryDelay

Description

La propriété `ConnectionRetryDelay` spécifie la durée, en secondes, pendant laquelle Interact attend avant toute tentative de reconnexion à la base de données pour les tables d'exécution en mode test après une défaillance. Si la valeur est définie sur -1, il n'y aura pas de tentative.

Valeur par défaut

-1

schéma

Description

Nom du schéma qui contient les tables destinées à l'exécution de diagrammes temps réel en mode test. Interact insère la valeur de cette propriété devant tous les noms de table. Par exemple, `UACI_IntChannel` devient `schema.UACI_IntChannel`.

Il n'est pas nécessaire de définir un schéma. Si vous ne définissez pas de schéma, Interact part du principe que le propriétaire des tables est le même que celui du schéma. Vous devez définir cette valeur pour supprimer toute ambiguïté.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | general | contactAndResponseHistoryDataSource

Ces propriétés de configuration définissent les paramètres de connexion pour la source de données de l'historique des réponses et des contacts nécessaires au suivi des réponses entre les sessions d'Interact.

Ces paramètres ne concernent pas le module d'historique des réponses et des contacts.

jndiName

Description

Utilisez cette propriété jndiName pour identifier les sources de données Java Naming and Directory Interface (JNDI) définies sur le serveur d'application (WebSphere ou WebLogic) de la source de données de l'historique des réponses et des contacts requises pour le suivi des réponses inter-sessions Interact.

Valeur par défaut

type

Description

Type de la base de données associé à la source utilisée par la source de l'historique des réponses entre les sessions Interact.

Valeur par défaut

SQLServer

Valeurs valides

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Description

La propriété ConnectionRetryPeriod spécifie la durée, en secondes, pendant laquelle Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour le suivi des réponses entre les sessions Interact. Interact essaie automatiquement de se reconnecter à la base de données pendant ce laps de temps avant de signaler une erreur ou un échec au niveau de la base de données. Si la valeur est définie sur 0, Interact essaiera indéfiniment ; si elle est définie sur -1, il n'y aura aucune tentative.

Valeur par défaut

-1

connectionRetryDelay

Description

La propriété ConnectionRetryDelay spécifie la durée, en secondes, pendant laquelle Interact attend avant toute tentative de reconnexion à la base de données pour le suivi des réponses entre les sessions Interact après une défaillance. Si la valeur est définie sur -1, il n'y aura pas de tentative.

Valeur par défaut

-1

schéma

Description

Nom du schéma qui contient les tables du suivi des réponses entre les sessions Interact. Interact insère la valeur de cette propriété devant tous les noms de table. Par exemple, UACI_IntChannel devient schema.UACI_IntChannel.

Il n'est pas nécessaire de définir un schéma. Si vous ne définissez pas de schéma, Interact part du principe que le propriétaire des tables est le même que celui du schéma. Vous devez définir cette valeur pour supprimer toute ambiguïté.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | general | idsByType

Ces propriétés de configuration définissent les paramètres des numéros d'ID utilisés par le module d'historique des réponses et des contacts.

initialValue

Description

Valeur d'ID initiale utilisée lors de la génération d'ID à l'aide de la table UACI_IDsByType.

Valeur par défaut

1

Valeurs valides

N'importe quelle valeur supérieure à 0.

retries

Description

Nombre de tentatives avant qu'une exception soit générée lors de la génération d'ID à l'aide de la table UACI_IDsByType.

Valeur par défaut

20

Valeurs valides

N'importe quel nombre entier supérieur à 0.

Interact | flowchart

Cette section définit les paramètres de configuration des diagrammes temps réel.

defaultDateFormat

Description

Format de date par défaut utilisé par Interact pour convertir la date en chaîne et inversement.

Valeur par défaut

jj/mm/aa

idleFlowchartThreadTimeoutInMinutes

Description

Nombre de minutes pendant lesquelles Interact autorise le thread dédié à un processus de diagramme temps réel à être inactif avant de libérer le thread.

Valeur par défaut

5

idleProcessBoxThreadTimeoutInMinutes

Description

Nombre de minutes pendant lesquelles Interact autorise le thread dédié à un diagramme temps réel à être inactif avant de le libérer.

Valeur par défaut

5

maxSizeOfFlowchartEngineInboundQueue

Description

Nombre maximum de requêtes de diagramme qu'Interact maintient dans la file d'attente. Si ce nombre de requêtes est atteint, Interact n'acceptera plus de requêtes.

Valeur par défaut

1000

maxNumberOfFlowchartThreads

Description

Nombre maximum de threads dédiés aux requêtes de diagrammes temps réel.

Valeur par défaut

25

maxNumberOfProcessBoxThreads

Description

Nombre maximum de threads dédiés aux processus de diagrammes temps réel.

Valeur par défaut

50

maxNumberOfProcessBoxThreadsPerFlowchart

Description

Nombre maximum de threads dédiés aux processus de diagrammes temps réel pour chaque instance de diagramme.

Valeur par défaut

3

minNumberOfFlowchartThreads

Description

Nombre minimum de threads dédiés aux requêtes de diagrammes temps réel.

Valeur par défaut

10

minNumberOfProcessBoxThreads

Description

Nombre minimum de threads dédiés aux processus de diagrammes temps réel.

Valeur par défaut

20

sessionVarPrefix

Description

Préfixe des variables de session.

Valeur par défaut

SessionVar

Interact | flowchart | ExternalCallouts | [ExternalCalloutName]

Cette section définit les paramètres de classe pour les appels externes personnalisés écrits à l'aide de l'API des appels externes.

class

Description

Nom de la classe Java représentée par cet appel externe.

Il s'agit de la classe Java à laquelle vous pouvez accéder avec IBM Unica via la macro `EXTERNALCALLOUT`.

Valeur par défaut

Aucune valeur par défaut définie.

classpath

Description

Nom du chemin d'accès de la classe Java représentée par l'appel externe. Le chemin d'accès aux classes doit correspondre aux fichiers jar sur le serveur d'environnement d'exécution. Si vous utilisez un groupe de serveurs et que tous les serveurs d'exécution utilisent Marketing Platform, une copie du fichier jar doit être présente au même emplacement de chaque serveur. Le chemin d'accès aux classes indique l'emplacement absolu des fichiers jar, séparés par le délimiteur du chemin d'accès du système d'exploitation du serveur d'environnement d'exécution, par exemple un point-virgule (;) pour Windows et deux points (:) pour UNIX. Les répertoires qui comportent des fichiers de classe ne sont pas acceptés. Par exemple, pour un système Unix : `/path1/file1.jar:/path2/file2.jar`.

Ce chemin d'accès aux classes doit comporter moins de 1 024 caractères. Vous pouvez utiliser le fichier manifeste dans un fichier .jar pour spécifier d'autres fichiers .jar. Ainsi, un seul fichier .jar est visible dans votre chemin d'accès à la classe

Il s'agit de la classe Java à laquelle vous pouvez accéder avec IBM Unica via la macro `EXTERNALCALLOUT`.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | flowchart | ExternalCallouts | [ExternalCalloutName] | Parameter Data | [parameterName]

Cette section définit le réglage des paramètres d'un appel externe personnalisé écrit à l'aide de l'API des appels externes.

valeur

Description

Valeur d'un paramètre requis par la classe de l'appel externe.

Valeur par défaut

Aucune valeur par défaut définie.

Exemple

Si le nom d'hôte d'un serveur externe est requis par l'appel externe, créez une catégorie de paramètre nommée `host` et définissez la propriété `value` en tant que nom du serveur.

Interact | monitoring

Ces propriétés de configuration vous permettent de définir les paramètres de suivi JMX. Vous ne devez configurer ces propriétés que lorsque vous utilisez le suivi JMX.

Des propriétés de suivi JMX séparées sont à définir pour le module d'historique des contacts et des réponses. Elles sont disponibles dans les propriétés de configuration pour l'environnement de conception d'Interact.

protocole

Description

Définissez le protocole du service de messagerie d'Interact.

Si vous choisissez JMXMP, vous devez inclure les fichiers JAR suivants dans votre classpath (dans l'ordre) :

`Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar`

Valeur par défaut

JMXMP

Valeurs valides

JMXMP | RMI

port

Description

Numéro de port du service de messagerie.

Valeur par défaut

9998

enableSecurity

Description

Opérateur booléen qui active ou désactive la sécurité du serveur d'exécution Interact. Si la valeur est réglée sur `True`, vous devez fournir un nom d'utilisateur et un mot de passe pour accéder au service JMX d'Interact. Ces données d'identification sont authentifiées par Marketing Platform pour le serveur d'exécution. Il n'est pas possible de se connecter sans mot de passe à la Jconsole.

Si le protocole utilisé est RMI, cette propriété n'a aucun effet. Cette propriété n'a aucun effet sur JMX pour Campaign (la phase de conception Interact).

Valeur par défaut

True

Valeurs valides

True | False

Interact | profile

Ces propriétés de configuration contrôlent plusieurs fonctions de service d'offres en option, notamment la suppression des offres et le remplacement de score.

enableScoreOverrideLookup

Description

Si la propriété est définie sur `True`, Interact charge les données de remplacement de score à partir de `scoreOverrideTable` lors de la création d'une session. Si la propriété est définie sur `False`, Interact ne charge pas les données de remplacement du score marketing lors de la création d'une session.

Si la valeur est True, vous devez également configurer la propriété `Unica > Interact > profile > Audience Levels > (Audience Level) > scoreOverrideTable`. Vous devez définir la propriété `scoreOverrideTable` uniquement pour les référentiels dont vous avez besoin. Si la propriété `scoreOverrideTable` est vide pour un référentiel, la table de remplacement de score correspondante est désactivée.

Valeur par défaut

`False`

Valeurs valides

True | False

enableOfferSuppressionLookup

Description

Si la propriété est définie sur `True`, Interact charge les données de suppression des offres à partir de la propriété `offerSuppressionTable` lors de la création d'une session. Si la propriété est définie sur `False`, Interact ne charge pas les données de suppression des offres lors de la création d'une session.

Si la valeur est True, vous devez également configurer la propriété `Unica > Interact > profile > Audience Levels > (Audience Level) > offerSuppressionTable`. Vous devez définir la propriété `enableOfferSuppressionLookup` uniquement pour les référentiels dont vous avez besoin.

Valeur par défaut

`False`

Valeurs valides

True | False

enableProfileLookup

Description

En cas de nouvelle installation d'Interact, cette propriété n'est pas autorisée. En cas de mise à jour de l'installation d'Interact, cette propriété est valide jusqu'à ce que le premier déploiement soit effectué.

Le comportement de chargement d'une table utilisée dans un diagramme temps réel mais non mappée dans le canal interactif. Si la propriété est définie sur `True`, Interact charge les données de profil à partir de `profileTable` lors de la création d'une session.

Si la valeur est True, vous devez également configurer la propriété `Unica > Interact > profile > Audience Levels > (Audience Level) > profileTable`.

Le paramètre **Charger ces données dans la mémoire au démarrage d'une session de visite** de l'assistant de mapping des tables de canal interactif remplace cette propriété de configuration.

Valeur par défaut

`False`

Valeurs valides

True | False

defaultOfferUpdatePollPeriod

Description

Durée, en secondes, pendant laquelle le système attend avant de mettre à jour les offres par défaut dans la mémoire cache à partir de la table des offres par défaut. Si la valeur est définie sur -1, le système ne procède pas à la mise à jour des offres par défaut dans la mémoire cache après chargement de la liste initiale dans la mémoire cache lors du démarrage du serveur d'exécution.

Valeur par défaut

-1

Interact | profile | Audience Levels | [AudienceLevelName]

Ces propriétés de configuration vous permettent de définir les noms de tables requis pour les fonctionnalités additionnelles d'Interact. Vous ne devez définir le nom de la table que lorsque vous utilisez la fonction correspondante.

scoreOverrideTable

Description

Nom de la table qui contient les informations de remplacement de score associées à ce référentiel. Cette propriété s'applique si vous avez défini la propriété `enableScoreOverrideLookup` sur vrai. Vous devez définir cette propriété pour les référentiels pour lesquels vous souhaitez qu'une table de remplacement de score soit activée. Si vous ne disposez pas de table de remplacement de score pour ce référentiel, vous pouvez laisser cette propriété non définie, même si `enableScoreOverrideLookup` est défini sur vrai.

Interact recherche cette table dans les tables client accessibles aux serveurs d'exécution Interact, définis par les propriétés `prodUserDataSource`.

Si vous avez défini la propriété `schema` pour cette source de données, Interact ajoutera ce nom de table au schéma, par exemple, `schema.UACI_ScoreOverride`. Si vous saisissez un nom complet, tel que `mySchema.UACI_ScoreOverride`, Interact n'ajoute pas le nom du schéma.

Valeur par défaut

UACI_ScoreOverride

offerSuppressionTable

Description

Nom de la table qui contient les informations de suppression d'offres associées à ce référentiel. Vous devez définir cette propriété pour les référentiels pour lesquels vous souhaitez qu'une table de suppression d'offres soit activée. Si vous ne disposez pas de table de suppression d'offres pour ce référentiel, vous pouvez laisser cette propriété non définie, même si `enableOfferSuppressionLookup` est défini sur vrai.

Interact recherche cette table dans les tables client accessibles aux serveurs d'exécution, définis par la propriété `prodUserDataSource`.

Valeur par défaut

UACI_BlackList

profileTable

Description

En cas de nouvelle installation d'Interact, cette propriété n'est pas autorisée. En cas de mise à jour de l'installation d'Interact, cette propriété est valide jusqu'à ce que le premier déploiement soit effectué.

Le nom de la table qui contient les données d'analyse associées à ce référentiel.

Interact recherche cette table dans les tables client accessibles aux serveurs d'exécution, définis par la propriété prodUserDataSource.

Si vous avez défini la propriété schema pour cette source de données, Interact ajoute ce nom de table au schéma, par exemple, schema.UACI_usrProd. Si vous saisissez un nom complet, tel que mySchema.UACI_usrProd, Interact n'ajoute pas le nom du schéma.

Valeur par défaut

Aucune valeur par défaut définie.

contactHistoryTable

Description

Nom de la table intermédiaire qui contient les données d'historique des contacts associées à ce référentiel.

Cette table est enregistrée dans les tables de l'environnement d'exécution (systemTablesDataSource).

Si vous avez défini la propriété schema pour cette source de données, Interact ajoute ce nom de table au schéma, par exemple, schema.UACI_CHStaging. Si vous saisissez un nom complet, tel que mySchema.UACI_CHStaging, Interact n'ajoute pas le nom du schéma.

Valeur par défaut

UACI_CHStaging

chOfferAttribTable

Description

Nom de la table qui contient les attributs d'offres de l'historique des contacts associés à ce référentiel.

Cette table est enregistrée dans les tables de l'environnement d'exécution (systemTablesDataSource).

Si vous avez défini la propriété schema pour cette source de données, Interact ajoutera ce nom de table au schéma, par exemple, schema.UACI_CHOfferAttrib. Si vous saisissez un nom complet, tel que mySchema.UACI_CHOfferAttrib, Interact n'ajoute pas le nom du schéma.

Valeur par défaut

UACI_CHOfferAttrib

responseHistoryTable

Description

Nom de la table intermédiaire qui contient les données d'historique des réponses associées à ce référentiel.

Cette table est enregistrée dans les tables de l'environnement d'exécution (systemTablesDataSource).

Si vous avez défini la propriété schema pour cette source de données, Interact ajoute ce nom de table au schéma, par exemple, schema.UACI_RHStaging. Si vous saisissez un nom complet, tel que mySchema.UACI_RHStaging, Interact n'ajoute pas le nom du schéma.

Valeur par défaut

UACI_RHStaging

crossSessionResponseTable

Description

Nom de la table associée à ce référentiel requis pour le suivi des réponses intersessions dans les tables d'historique des contacts et des réponses accessibles à la fonction de suivi des réponses.

Si vous avez défini la propriété `schema` pour cette source de données, Interact ajoute ce nom de table au schéma, par exemple, `schema.UACI_XSessResponse`. Si vous saisissez un nom complet, tel que `mySchema.UACI_XSessResponse`, Interact n'ajoute pas le nom du schéma.

Valeur par défaut

`UACI_XSessResponse`

Interact | profil | Référentiels | [AudienceLevelName] | Offers by Raw SQL

Ces propriétés de configuration vous permettent de définir les noms de table requis pour des fonctions Interact supplémentaires. Vous ne devez définir le nom de la table que lorsque vous utilisez la fonction correspondante.

enableOffersByRawSQL

Description

S'il est défini sur `True`, Interact active la fonction offersBySQL pour ce référentiel. Cela vous permet de configurer le code SQL à exécuter pour créer l'ensemble d'offres candidates souhaité lors de l'exécution. Si elle a la valeur `False`, Interact n'utilise pas la fonction offersBySQL.

Si vous définissez cette propriété sur true, vous pouvez également configurer la propriété `Unica | Interact | profil | Référentiels | (Audience Level) | Offers by Raw SQL | SQL Template` afin de définir un ou plusieurs modèles SQL.

Valeur par défaut

`False`

Valeurs valides

True | False

cacheSize

Description

Taille du cache utilisé pour stocker les résultats des requêtes OfferBySQL. Notez que l'utilisation d'un cache peut avoir un impact négatif si les résultats de la requête sont uniques pour la plupart des sessions.

Valeur par défaut

`-1 (off)`

Valeurs valides

-1 | Valeur

cacheLifeInMinutes

Description

Si le cache est activé, cette option indique le nombre de minutes d'attente avant que le système n'efface le cache pour éviter qu'il ne devienne périmé.

Valeur par défaut

`-1 (off)`

Valeurs valides

-1 | Valeur

defaultSQLTemplate

Description

Nom du modèle SQL à utiliser si aucun n'est spécifié via les appels d'API.

Valeur par défaut

`Aucune`

Valeurs valides

Nom du modèle SQL

Interact | profil | Référentiels | [AudienceLevelName] | SQL Template

Ces propriétés de configuration vous permettent de définir un ou plusieurs modèles de requête SQL utilisés par la fonction offersBySQL de Interact.

name

Description

Nom que vous souhaitez affecter à ce modèle de requête SQL. Entrez un nom descriptif ayant une signification assez claire pour être facilement utilisé avec ce modèle SQL dans les appels d'API. Notez que si vous utilisez ici un nom *identique* à un nom défini dans le processus de la zone Liste de processus. d'Interact pour un traitement offerBySQL, l 'instruction SQL de la zone du processus sera utilisée au lieu de l'instruction SQL que vous entrez ici.

Valeur par défaut

Aucune

SQL

Description

Contient la requête SQL devant être appelée par ce modèle. La requête SQL peut contenir des références à des noms de variables faisant partie des données de la session du visiteur (profil). Par exemple, `select * from MyOffers where category = ${preferredCategory}` utilisera la session contenant la variable appelée `preferredCategory`.

Vous devez configurer l'instruction SQL afin qu'elle interroge les tables d'offre spécifiques que vous avez créées lors de la phase de conception en vue d'une utilisation par cette fonction. Les procédures mémorisées ne sont pas prises en charge ici.

Valeur par défaut

Aucune

Interact | profil | Référentiels | [AudienceLevelName | Profile Data Services | [DataSource]

Ces propriétés de configuration vous permettent de définir les noms de table requis pour des fonctions Interact supplémentaires. Vous ne devez définir le nom de la table que lorsque vous utilisez la fonction correspondante. La catégorie Profile Data Services fournit des informations sur une source de données intégrée (appelée Base de données) qui est créée pour tous les référentiels, et qui est préconfiguré avec une priorité de 100. Toutefois, vous pouvez décider de la modifier ou de la désactiver. La catégorie contient également un modèle destiné à des sources de données externes supplémentaires. Lorsque vous cliquez sur le modèle appelé **External Data Services**, vous pouvez définir les paramètres de configuration décrits ici.

Nouveau nom de la catégorie

Description

(indisponible pour l'entrée de Base de données par défaut). Nom de la source de données que vous définissez. Le nom que vous entrez ici doit être unique dans les sources de données pour le même référentiel.

Valeur par défaut

Aucune

Valeurs valides

N'importe quelle chaîne de texte est autorisée.

enabled

Description

Si elle est définie sur `True`, Interact, cette source de données est activée pour le référentiel auquel elle est affectée. Si elle a la valeur `False`, Interact, elle n'utilise pas cette source de données pour ce référentiel.

Valeur par défaut

`True`

Valeurs valides

True | False

className

Description

(indisponible pour l'entrée de Base de données par défaut). Nom qualifié complet de la classe de source de données qui implémente `IInteractProfileDataService`.

Valeur par défaut

Aucun.

Valeurs valides

Chaîne fournissant un nom de classe qualifié complet.

classPath

Description

(indisponible pour l'entrée de Base de données par défaut). Un paramètre de configuration facultatif qui fournit le chemin permettant de charger cette classe d'implémentation de la source de données. Si vous l'omettez, le chemin d'accès aux classes du serveur d'applications qui le contient est utilisé par défaut.

Valeur par défaut

Non indiqué, mais le chemin d'accès aux classes du serveur d'applications qui le contient est utilisé par défaut.

Valeurs valides

Chaîne fournissant le chemin de classe.

priority

Description

Priorité de cette source de données dans le référentiel. Il doit s'agir d'une valeur unique dans toutes les sources de données de chaque référentiel. Si une priorité est définie sur 100 pour une source de données, aucune autre source de données dans le référentiel ne peut avoir de priorité de 100.

Valeur par défaut

`100` pour la Base de données par défaut, `200` pour la source de données définie par l'utilisateur

Valeurs valides

Tout entier non négatif est autorisé.

Interact | offerserving

Ces propriétés de configuration définissent les propriétés de configuration d'apprentissage génériques.

Pour régler votre implémentation d'apprentissage lorsque vous utilisez l'apprentissage intégré, utilisez les propriétés de configuration associées à l'environnement de conception.

optimizationType

Description

La propriété `optimizationType` permet de déterminer si Interact utilise un moteur d'apprentissage pour aider dans les affectations d'offres. Si la propriété est définie sur `NoLearning`, Interact n'utilise pas l'apprentissage. Si la propriété est définie sur `BuiltInLearning`, Interact utilise le moteur d'apprentissage intégré à Interact. Si la propriété est définie sur `ExternalLearning`, Interact utilise votre moteur d'apprentissage. Si vous sélectionnez `ExternalLearning`, vous devez définir les propriétés `externalLearningClass` et `externalLearningClassPath`.

Valeur par défaut

NoLearning

Valeurs valides

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

Description

Durée maximale, en millisecondes, pendant laquelle le serveur d'exécution attend qu'un diagramme temps réel prenne fin avant de récupérer les offres.

Valeur par défaut

5000

treatmentCodePrefix

Description

Préfixe ajouté aux codes de traitement.

Valeur par défaut

Aucune valeur par défaut définie.

Interact | offerserving | Built-in Learning Config

Ces propriétés de configuration définissent les paramètres d'écriture de la base de données pour les tables d'apprentissage intégrées.

Pour régler votre implémentation d'apprentissage, utilisez les propriétés de configuration associées à l'environnement de conception.

insertRawStatsIntervalInMinutes

Description

Le nombre de minutes pendant lesquelles le mode d'apprentissage Interact attend avant d'insérer plus de lignes dans les tables intermédiaires d'apprentissage. Vous devrez peut-être modifier cette durée en fonction de la quantité de données traitées par le module d'apprentissage dans votre environnement.

Valeur par défaut

5

aggregateStatsIntervalInMinutes

Description

Le nombre de minutes pendant lesquelles le mode d'apprentissage Interact attend entre l'agrégation des données dans les tables intermédiaires d'apprentissage. Vous devrez peut-être modifier cette durée en fonction de la quantité de données traitées par le module d'apprentissage dans votre environnement.

Valeur par défaut

15

Valeurs valides

Un nombre entier supérieur à zéro.

Interact | offerserving | External Learning Config

Ces propriétés de configuration définissent les paramètres de classe associés à un module d'apprentissage externe écrit à l'aide de l'API d'apprentissage.

class

Description

Si `optimizationType` est définie sur `ExternalLearning`, définissez `externalLearningClass` sur le nom de classe associé au moteur d'apprentissage externe.

Valeur par défaut

Aucune valeur par défaut définie.

Disponibilité

Cette propriété s'applique uniquement si la propriété `optimizationType` est définie sur `ExternalLearning`.

classPath

Description

Si `optimizationType` est définie sur `ExternalLearning`, définissez `externalLearningClass` sur le chemin d'accès associé au moteur d'apprentissage externe.

Le chemin d'accès à la classe doit correspondre aux fichiers jar sur le serveur d'environnement d'exécution. Si vous utilisez un groupe de serveurs et que tous les serveurs d'exécution utilisent Marketing Platform, une copie du fichier jar doit être présente au même emplacement de chaque serveur. Le chemin d'accès aux classes indique l'emplacement absolu des fichiers jar, séparés par le délimiteur du chemin d'accès du système d'exploitation du serveur d'environnement d'exécution, par exemple un point-virgule (;) pour Windows et deux points (:) pour UNIX. Les répertoires qui comportent des fichiers de classe ne sont pas acceptés. Par exemple, pour un système Unix : `/path1/file1.jar:/path2/file2.jar`.

Ce chemin d'accès doit comporter moins de 1 024 caractères. Vous pouvez utiliser le fichier manifeste dans un fichier .jar pour spécifier d'autres fichiers .jar. Ainsi, un seul fichier .jar est visible dans votre chemin d'accès aux classes

Valeur par défaut

Aucune valeur par défaut définie.

Disponibilité

Cette propriété s'applique uniquement si la propriété `optimizationType` est définie sur `ExternalLearning`.

Interact | offerserving | External Learning Config | Parameter Data | [parameterName]

Ces propriétés de configuration définissent les paramètres associés à votre module d'apprentissage externe.

valeur

Description

Valeur d'un paramètre requis par la classe d'un module d'apprentissage externe.

Valeur par défaut

Aucune valeur par défaut définie.

Exemple

Lorsque le module d'apprentissage externe requiert le chemin d'une application de résolution des algorithmes, il est nécessaire de créer une catégorie de paramètres appelée `solverPath` et de définir la propriété `value` en tant que chemin de l'application.

Interact | services

Les propriétés de configuration de cette catégorie définissent les paramètres associés à tous les services qui collectent les données et les statistiques relatives à l'historique des contacts et des réponses à des fins de création de rapports et d'écriture dans les tables système de l'environnement d'exécution.

externalLoaderStagingDirectory

Description

Cette propriété définit l'emplacement du répertoire intermédiaire d'un utilitaire de chargement de base de données.

Valeur par défaut

Aucune valeur par défaut définie.

Valeurs valides

Un chemin d'accès associé au répertoire d'installation d'Interact ou le chemin d'accès absolu d'un répertoire intermédiaire.

Si vous activez un utilitaire de chargement de base de données, vous devez définir la propriété `cacheType` des catégories `contactHist` et `responstHist` sur `External Loader File`.

Interact | services | contactHist

Les propriétés de configuration de cette catégorie définissent les paramètres associés au service de collecte des données pour les tables intermédiaires d'historique des contacts.

enableLog

Description

Si la valeur est définie sur `vrai`, le service qui collecte les données en vue d'enregistrer les données de l'historique des contacts est activé. Si la valeur est définie sur `faux`, aucune donnée n'est collectée.

Valeur par défaut

True

Valeurs valides

True | False

cacheType

Description

Cette propriété indique si les données collectées pour l'historique des contacts sont gardées en mémoire (`Memory Cache`) ou dans un fichier (`External Loader file`). Vous pouvez uniquement utiliser `External Loader File` si vous avez configuré Interact pour employer un utilitaire de chargement de base de données.

Si vous sélectionnez `Memory Cache`, utilisez les paramètres de la catégorie `cache`. Si vous sélectionnez `External Loader File`, utilisez les paramètres de la catégorie `fileCache`.

Valeur par défaut

Mémoire cache

Valeurs valides

Memory Cache | External Loader File

Interact | services | contactHist | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service de collecte des données pour la table intermédiaire d'historique des contacts.

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les données collectées d'historique des contacts dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | contactHist | fileCache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service de collecte des données d'historique des contacts en cas d'utilisation d'un utilitaire de chargement.

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les données collectées d'historique des contacts dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | defaultedStats

Les propriétés de configuration de cette catégorie définissent les paramètres associés au service qui collecte les statistiques relatives au nombre d'utilisations de la chaîne par défaut du point d'interaction.

enableLog

Description

Si la valeur est définie sur `vrai`, le service qui collecte les statistiques relatives au nombre d'utilisations de la chaîne par défaut du point d'interaction dans la table `UACI_DefaultedStat` est activé. Si la valeur est définie sur `faux`, aucune statistique n'est collectée.

La collecte de données n'étant pas requise, vous pouvez définir cette propriété sur `faux` si vous n'utilisez pas la création de rapports d'IBM.

Valeur par défaut

True

Valeurs valides

True | False

Interact | services | defaultedStats | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service qui collecte les statistiques relatives au nombre d'utilisations de la chaîne par défaut du point d'interaction.

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les statistiques collectées relatives à la chaîne par défaut dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | eligOpsStats

Les propriétés de configuration de cette catégorie définissent les paramètres associés au service qui écrit les statistiques relatives aux offres éligibles.

enableLog

Description

Si la valeur est réglée sur `vrai`, le service qui collecte les statistiques relatives aux offres éligibles est activé. Si la valeur est définie sur `faux`, aucune statistique n'est collectée.

La collecte de données n'étant pas requise, vous pouvez définir cette propriété sur `faux` si vous n'utilisez pas la création de rapports d'IBM.

Valeur par défaut

True

Valeurs valides

True | False

Interact | services | eligOpsStats | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service qui collecte les statistiques relatives aux offres éligibles.

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les statistiques collectées relatives aux offres éligibles dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | eventActivity

Les propriétés de configuration de cette catégorie définissent les paramètres associés au service qui collecte les statistiques relatives à l'activité des événements.

enableLog

Description

Si la valeur est réglée sur `vrai`, le service qui collecte les statistiques relatives à l'activité des événements est activé. Si la valeur est définie sur `faux`. aucune statistique n'est collectée.

La collecte de données n'étant pas requise, vous pouvez définir cette propriété sur `faux` si vous n'utilisez pas la création de rapports d'IBM.

Valeur par défaut

True

Valeurs valides

True | False

Interact | services | eventActivity | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service qui collecte les statistiques relatives à l'activité des événements.

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les statistiques collectées relatives à l'activité des événements dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | customLogger

Les propriétés de configuration de cette catégorie définissent les paramètres associés au service qui collecte des données personnalisées en vue de les écrire dans une table (événement qui utilise le paramètre d'événement `UACICustomLoggerTableName`).

enableLog

Description

Si la valeur est définie sur vrai, la fonction de conversion du journal personnalisé en table est activée. Si la valeur est définie sur faux, le paramètre d'événement UACICustomLoggerTableName n'a aucun effet.

Valeur par défaut

True

Valeurs valides

True | False

Interact | services | customLogger | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service qui collecte des données personnalisées en vue de les convertir en table (événement qui utilise le paramètre d'événement UACICustomLoggerTableName).

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les données personnalisées collectées dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | responseHist

Les propriétés de configuration de cette catégorie définissent les paramètres associés au service qui écrit dans les tables intermédiaires d'historique des réponses.

enableLog

Description

Si la valeur est définie sur vrai, le service qui écrit dans les tables intermédiaires d'historique des réponses est activé. Si la valeur est définie sur faux, aucune donnée n'est écrite.

La table intermédiaire d'historique des réponses est définie par la propriété de référentiel responseHistoryTable. La valeur par défaut est UACI_RHStaging.

Valeur par défaut

True

Valeurs valides

True | False

cacheType

Description

Définit si les données de cache sont gardées en mémoire ou dans un fichier. Vous pouvez uniquement utiliser `External Loader File` si vous avez configuré Interact pour employer un utilitaire de chargement de base de données.

Si vous sélectionnez `Memory Cache`, utilisez les paramètres de la catégorie `cache`. Si vous sélectionnez `External Loader File`, utilisez les paramètres de la catégorie `fileCache`.

Valeur par défaut

Mémoire cache

Valeurs valides

Memory Cache | External Loader File

Interact | services | responseHist | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service qui collecte les données d'historique des réponses.

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les données collectées d'historique des réponses dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | responseHist | fileCache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service qui collecte les données d'historique des réponses en cas d'emploi d'un utilitaire de chargement.

seuil

Description

Nombre d'enregistrements accumulés avant que le Interact les écrive dans la base de données.

`responseHist` - La table définie par la propriété de référentiel `responseHistoryTable`. La valeur par défaut est `UACI_RHStaging`.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la base de données.

Valeur par défaut

3600

Interact | services | crossSessionResponse

Les propriétés de configuration de cette catégorie définissent les paramètres généraux associés au service crossSessionResponse et au processus intersession. Vous devez configurer ces paramètres uniquement si vous utilisez le suivi des réponses entre les sessions d'Interact.

enableLog

Description

Si la valeur est définie sur `true`, le service crossSessionResponse est activé et Interact écrit des données dans les tables intermédiaires de suivi des réponses intersessions. Si la valeur est définie sur `false`, le service crossSessionResponse est désactivé.

Valeur par défaut

False

xsessionProcessIntervalInSecs

Description

Nombre de secondes entre chaque exécution du processus intersession. Ce processus déplace les données des tables intermédiaires de suivi des réponses intersessions dans la table intermédiaire d'historique des réponses et dans le module d'apprentissage intégré.

Valeur par défaut

180

Valeurs valides

Un nombre entier supérieur à zéro

purgeOrphanResponseThresholdInMinutes

Description

Durée, en minutes, pendant laquelle le service crossSessionResponse attend avant de baliser les réponses qui ne correspondent pas aux contacts dans les tables d'historique des contacts et des réponses.

Si une réponse ne concorde pas avec les tables d'historique des réponses et contacts, après un délai de `purgeOrphanResponseThresholdInMinutes` minutes, Interact associe la valeur -1 à la réponse dans la colonne `Mark` de la table intermédiaire xSessResponse. Vous pouvez alors faire correspondre ou supprimer ces réponses manuellement.

Valeur par défaut

180

Interact | services | crossSessionResponse | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de mémoire cache associés au service de collecte des données de réponses intersessions.

seuil

Description

Nombre d'enregistrements accumulés avant que le service flushCacheToDB écrive les données collectées de réponses intersessions dans la base de données.

Valeur par défaut

100

insertPeriodInSecs

Description

Nombre de secondes entre chaque écriture forcée dans la table `XSessResponse`.

Valeur par défaut

3600

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

Les propriétés de cette section définissent la méthode appliquée par le suivi des réponses intersessions pour faire correspondre des codes de traitement à l'historique des contacts et des réponses.

SQL

Description

Cette propriété définit si Interact doit utiliser l'instruction System Generated SQL ou l'instruction SQL personnalisée définie dans la propriété `OverrideSQL`.

Valeur par défaut

Utiliser System Generated SQL

Valeurs valides

Utiliser System Generated SQL | Effacer SQL

OverrideSQL

Description

Si vous n'utilisez pas la commande SQL par défaut pour faire correspondre le code de traitement à l'historique des contacts et des réponses, entrez l'instruction SQL ou la procédure enregistrée ici.

Cette valeur est ignorée si SQL est défini sur Use System Generated SQL.

Valeur par défaut

useStoredProcedure

Description

Si la propriété est définie sur true, OverrideSQL doit comporter une référence vers une procédure enregistrée qui fait correspondre le code de traitement à l'historique des contacts et des réponses.

Si elle est définie sur false et qu'elle est utilisée, la propriété OverrideSQL doit correspondre à une requête SQL.

Valeur par défaut

false

Valeurs valides

true | false

Type

Description

Le TrackingCodeType associé défini dans la table UACI_TrackingType des tables de l'environnement d'exécution. Sauf si la table UACI_TrackingType est modifiée, le Type doit être défini sur 1.

Valeur par défaut

1

Valeurs valides

Nombre entier défini dans la table UACI_TrackingType.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

Dans cette section, les propriétés définissent la méthode appliquée par le suivi des réponses intersessions pour faire correspondre des codes d'offre à l'historique des contacts et des réponses.

SQL

Description

Cette propriété définit si Interact doit utiliser l'instruction System Generated SQL ou l'instruction SQL personnalisée définie dans la propriété OverrideSQL.

Valeur par défaut

Utiliser System Generated SQL

Valeurs valides

Utiliser System Generated SQL | Effacer SQL

OverrideSQL

Description

Si vous n'utilisez pas la commande SQL par défaut pour faire correspondre le code d'offre à l'historique des contacts et des réponses, entrez l'instruction SQL ou la procédure enregistrée ici.

Cette valeur est ignorée si SQL est défini sur Use System Generated SQL.

Valeur par défaut

useStoredProcedure

Description

Si la valeur est définie sur vrai, la propriété OverrideSQL doit comporter une référence vers une procédure enregistrée qui fait correspondre le code d'offre à l'historique des contacts et des réponses.

Si elle est définie sur false et qu'elle est utilisée, la propriété OverrideSQL doit correspondre à une requête SQL.

Valeur par défaut

false

Valeurs valides

true | false

Type

Description

Le TrackingCodeType associé défini dans la table UACI_TrackingType des tables de l'environnement d'exécution. Sauf si la table UACI_TrackingType est modifiée, le Type doit être défini sur 2.

Valeur par défaut

2

Valeurs valides

Nombre entier défini dans la table UACI_TrackingType.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

Dans cette section, les propriétés définissent la méthode appliquée par le suivi des réponses intersessions pour faire correspondre un code alternatif défini par l'utilisateur à l'historique des contacts et des réponses.

Name

Description

Cette propriété définit le nom du code alternatif. Il doit correspondre à la valeur Name de la table UACI_TrackingType des tables de l'environnement d'exécution.

Valeur par défaut

OverrideSQL

Description

Commande SQL ou procédure enregistrée appliquée pour faire correspondre le code alternatif à l'historique des contacts et des réponses par code d'offre ou de traitement.

Valeur par défaut

useStoredProcedure

Description

Si la propriété est définie sur true, `OverrideSQL` doit comporter une référence vers une procédure enregistrée faisant correspondre le code alternatif à l'historique des contacts et des réponses.

Si elle est définie sur false et qu'elle est utilisée, la propriété `OverrideSQL` doit correspondre à une requête SQL.

Valeur par défaut

false

Valeurs valides

true | false

Type

Description

Le TrackingCodeType associé défini dans la table `UACI_TrackingType` des tables de l'environnement d'exécution.

Valeur par défaut

3

Valeurs valides

Nombre entier défini dans la table `UACI_TrackingType.`

Interact | services | threadManagement | contactAndResponseHist

Les propriétés de configuration de cette catégorie définissent les paramètres de gestion des threads associés aux services qui collectent les données pour les tables intermédiaires d'historique des contacts et des réponses.

corePoolSize

Description

Nombre de threads à conserver dans le pool, qu'ils soient actifs ou non, pour la collecte des données d'historique des contacts et des réponses.

Valeur par défaut

5

maxPoolSize

Description

Nombre maximum de threads à conserver dans le pool pour la collecte des données d'historique des contacts et des réponses.

Valeur par défaut

5

keepAliveTimeSecs

Description

Lorsque le nombre de threads est supérieur au nombre principal, il s'agit du laps de temps maximum pendant lequel les threads inactifs en trop attendent que les nouvelles tâches prennent fin en vue de collecter les données d'historique des contacts et des réponses.

Valeur par défaut

5

queueCapacity

Description

Taille de la file d'attente utilisée par le pool de threads pour collecter les données d'historique des contacts et des réponses.

Valeur par défaut

1000

termWaitSecs

Description

Lors de la fermeture du serveur d'exécution, laps de temps à attendre, en secondes, pour que les threads de service terminent de collecter les données d'historique des contacts et des réponses.

Valeur par défaut

5

Interact | services | threadManagement | allOtherServices

Les propriétés de configuration de cette catégorie définissent les paramètres de gestion du thread associés aux services de collecte des statistiques relatives à l'offre d'éligibilité, à l'activité des événements, à l'utilisation de la chaîne par défaut ainsi que les données de conversion du journal personnalisé en table.

corePoolSize

Description

Nombre de threads à conserver dans le pool, qu'ils soient actifs ou non, associés aux services qui collectent les statistiques relatives à l'offre d'éligibilité, à l'activité des événements, à l'utilisation de la chaîne par défaut ainsi que les données de conversion du journal personnalisé en table.

Valeur par défaut

5

maxPoolSize

Description

Nombre maximum de threads à conserver dans le pool et associés aux services de collecte des statistiques relatives à l'offre d'éligibilité, à l'activité des événements, à l'utilisation de la chaîne par défaut ainsi que les données de conversion du journal personnalisé en table.

Valeur par défaut

5

keepAliveTimeSecs

Description

Lorsque le nombre de threads est supérieur au nombre principal, il s'agit du laps de temps maximum pendant lequel les threads inactifs en trop associés aux services de collecte des statistiques relatives à l'offre d'éligibilité, à l'activité des événements, à l'utilisation de la chaîne par défaut ainsi que les données de conversion du journal personnalisé en table attendent de nouvelles tâches avant de prendre fin.

Valeur par défaut

5

queueCapacity

Description

Taille de la file d'attente utilisée par le pool de threads associé aux services de collecte des statistiques relatives à l'offre d'éligibilité, à l'activité des événements, à l'utilisation de la chaîne par défaut ainsi que les données de conversion du journal personnalisé en table.

Valeur par défaut

1000

termWaitSecs

Description

Lors de la fermeture du serveur d'exécution, laps de temps à attendre, en secondes, pour que les threads de service associés aux services de collecte des statistiques relatives à l'offre d'éligibilité, à l'activité des événements, à l'utilisation de la chaîne par défaut ainsi que les données de conversion du journal personnalisé en table prennent fin.

Valeur par défaut

5

Interact | services | threadManagement | flushCacheToDB

Les propriétés de configuration de cette catégorie définissent les paramètres de gestion des threads associés aux threads qui écrivent les données de cache collectées dans les tables des bases de données de l'environnement d'exécution.

corePoolSize

Description

Nombre de threads à conserver dans le pool pour les threads planifiés qui écrivent les données de cache dans le magasin de données.

Valeur par défaut

5

maxPoolSize

Description

Nombre maximum de threads à conserver dans le pool pour les threads planifiés qui écrivent les données de cache dans le magasin de données.

Valeur par défaut

5

keepAliveTimeSecs

Description

Lorsque le nombre de threads est supérieur au nombre principal, il s'agit du laps de temps maximum pendant lequel les threads inactifs en trop destinés aux threads planifiés qui écrivent les données de cache dans le magasin de données attendent de nouvelles tâches avant de prendre fin.

Valeur par défaut

5

queueCapacity

Description

Taille de la file d'attente utilisée par le pool de threads destiné aux threads planifiés qui écrivent les données de cache dans le magasin de données.

Valeur par défaut

1000

termWaitSecs

Description

Lors de la fermeture du serveur d'exécution, laps de temps à attendre, en secondes, pour que les threads de service associés aux threads planifiés qui écrivent les données de cache dans le magasin de données prennent fin.

Valeur par défaut

5

Interact | sessionManagement

Ces propriétés de configuration définissent les paramètres associés aux sessions d'exécution.

cacheType

Description

Définit le type de mémoire cache destinée aux serveurs d'exécution.

Valeur par défaut

`Local(e)`

Valeurs valides

Distributed | Local

maxNumberOfSessions

Description

Nombre maximum de sessions d'exécution conservées en permanence dans la mémoire cache. Si une requête vous invitant à ajouter une nouvelle session d'exécution s'affiche lorsque la mémoire cache atteint ce nombre maximum, la session d'exécution inactive la moins récente est supprimée.

Valeur par défaut

999999999

Valeurs valides

Un nombre entier supérieur à 0.

multicastIPAddress

Description

Si `cacheType` est défini sur `Distributed`, saisissez l'adresse IP utilisée par la mémoire cache distribuée. Vous devez également définir la propriété `multicastPort`.

Si `cacheType` est défini sur `Local`, vous pouvez laisser `multicastIPAddress` non défini.

Valeur par défaut

230.0.0.1

Valeurs valides

N'importe quelle adresse IP valide.

multicastPort

Description

Si `cacheType` est défini sur `Distributed`, saisissez le numéro de port utilisé par la mémoire cache distribuée. Vous devez également définir la propriété `multicastIPAddress`.

Si `cacheType` est défini sur `Local`, vous pouvez laisser `multicastPort` non défini.

Valeur par défaut

6363

Valeurs valides

1024 – 49151

sessionTimeoutInSecs

Description

Laps de temps, en secondes, pendant lequel une session peut rester inactive. Une fois que le laps de temps spécifié dans `sessionTimeout` est écoulé, Interact met fin à la session.

Valeur par défaut

`300`

Valeurs valides

N'importe quel nombre entier supérieur à zéro.

Annexe C. Interact propriétés de configuration de l'environnement de conception

Cette section décrit toutes les propriétés de configuration de l'environnement de conception d'Interact.

Campaign | partitions | partition[n] | reports

Ces propriétés de configuration définissent les dossiers de rapports.

offerAnalysisTabCachedFolder

Description

La propriété `offerAnalysisTabCachedFolder` indique l'emplacement du dossier qui contient la spécification des rapports d'offre transmis en une fois et répertoriés dans l'onglet Analyse, accessible via le lien Analyse du volet de navigation. Le chemin d'accès est spécifié via la notation XPath.

Valeur par défaut

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

Description

La propriété `segmentAnalysisTabOnDemandFolder` indique l'emplacement du dossier qui contient les rapports de segmentation répertoriés dans l'onglet Analyse d'un segment. Le chemin d'accès est spécifié via la notation XPath.

Valeur par défaut

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

Description

La propriété `offerAnalysisTabOnDemandFolder` indique l'emplacement du dossier qui contient les rapports d'offre répertoriés dans l'onglet Analyse d'une offre. Le chemin d'accès est spécifié via la notation XPath.

Valeur par défaut

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

segmentAnalysisTabCachedFolder

Description

La propriété `segmentAnalysisTabCachedFolder` indique l'emplacement du dossier qui contient la spécification des rapports de segment transmis en une fois et répertoriés dans l'onglet Analyse, accessible via le lien Analyse du volet de navigation. Le chemin d'accès est spécifié via la notation XPath.

© Copyright IBM Corp. 2001, 2012

Valeur par défaut

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']
```

analysisSectionFolder

Description

La propriété `analysisSectionFolder` indique l'emplacement du dossier racine dans lequel les spécifications des rapports sont enregistrées. Le chemin d'accès est spécifié via la notation XPath.

Valeur par défaut

```
/content/folder[@name='Affinium Campaign']
```

campaignAnalysisTabOnDemandFolder

Description

La propriété `campaignAnalysisTabOnDemandFolder` indique l'emplacement du dossier qui contient les rapports de campagne répertoriés dans l'onglet Analyse d'une campagne. Le chemin d'accès est spécifié via la notation XPath.

Valeur par défaut

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']
```

campaignAnalysisTabCachedFolder

Description

La propriété `campaignAnalysisTabCachedFolder` indique l'emplacement du dossier qui contient la spécification des rapports de campagne transmis en une fois et répertoriés dans l'onglet Analyse, accessible via le lien Analyse du volet de navigation. Le chemin d'accès est spécifié via la notation XPath.

Valeur par défaut

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']/folder[@name='cached']
```

campaignAnalysisTabEmessageOnDemandFolder

Description

La propriété `campaignAnalysisTabEmessageOnDemandFolder` indique l'emplacement du dossier qui contient les rapports d'eMessage répertoriés dans l'onglet Analyse d'une campagne. Le chemin d'accès est spécifié via la notation XPath.

Valeur par défaut

```
/content/folder[@name='Affinium Campaign']/folder[@name='eMessage
Reports']
```

campaignAnalysisTabInteractOnDemandFolder

Description

Chaîne du dossier de serveur de rapports pour les rapports Interact.

Valeur par défaut

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

Disponibilité

Cette propriété est applicable uniquement si vous avez installé Interact.

interactiveChannelAnalysisTabOnDemandFolder

Description

Chaîne du dossier de serveur de rapports pour les rapports de l'onglet Analyse du canal interactif

Valeur par défaut

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

Disponibilité

Cette propriété est applicable uniquement si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking

Ces propriétés de configuration définissent les paramètres du module d'historique des réponses et des contacts d'Interact.

isEnabled

Description

Si la valeur est définie sur `oui`, cela active le module d'historique des réponses et des contacts d'Interact qui copie l'historique des réponses et des contacts d'Interact des tables intermédiaire de l'environnement d'exécution d'Interact dans les tables de l'historique des réponses et des contacts de Campaign. La propriété `interactInstalled` doit également être définie sur `oui`.

Valeur par défaut

non

Valeurs valides

oui | non

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

runOnceADay

Description

Spécifie s'il faut exécuter l'historique des contacts et des réponses ETL une fois par jour. Si vous définissez cette propriété sur `Yes`, l'ETL est exécuté dans l'intervalle planifié spécifié par `preferredStartTime` et `preferredEndTime`.

Si ETL prend plus de 24 heures pour s'exécuter, et manque par conséquent l'heure de début pour le lendemain, il ignorera ce jour-là et s'exécutera à

l'heure planifiée le lendemain. Par exemple, si ETL est configuré pour s'exécuter entre 01h00 et 03h00, et si le processus démarre à 01h00 le lundi et s'achève à 02h00 le mardi, la prochaine exécution, planifiée à l'origine pour 01h00 le mardi, sera ignorée, et le prochain ETL démarrera à 01h00 le mercredi.

La planification ETL ne tient pas compte du passage à l'heure d'été. Par exemple, s'il est planifié qu'ETL s'exécute entre 01h00 et 03h00, il pourrait s'exécuter à 00h00 ou à 02h00 lors du passage à l'heure d'été.

Valeur par défaut

Non

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

processSleepIntervalInMinutes

Description

Le nombre de minutes pendant lesquelles le module de l'historique des réponses et des contacts d'Interact attend entre les copies des données des tables intermédiaires d'exécution d'Interact dans les tables de l'historique des réponses et des contacts d'Campaign.

Valeur par défaut

60

Valeurs valides

N'importe quel nombre entier supérieur à zéro.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

preferredStartTime

Description

L'heure préférée pour démarrer le processus ETL quotidien. Cette propriété, si utilisée conjointement avec la propriété preferredEndTime, définira l'intervalle de temps préféré pendant lequel vous souhaitez exécuter l'ETL. L'ETL démarrera pendant l'intervalle de temps spécifié et traitera le nombre d'enregistrements spécifié à l'aide de `maxJDBCFetchBatchSize`. Le format est HH:mm:ss (horloge de 24 heures).

Valeur par défaut

00:00:00

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

preferredEndTime

Description

L'heure préférée pour achever le processus ETL quotidien. Cette propriété, si utilisée conjointement avec la propriété preferredStartTime, définira l'intervalle de temps préféré pendant lequel vous souhaitez exécuter l'ETL. L'ETL démarrera pendant l'intervalle de temps spécifié et traitera le

nombre d'enregistrements spécifié à l'aide de maxJDBCFetchBatchSize. Le format est HH:mm:ss (horloge de 24 heures).

Valeur par défaut

02:00:00

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

purgeOrphanResponseThresholdInMinutes

Description

Le nombre de minutes pendant lesquelles Interact attend avant d'éliminer les réponses qui ne correspondent à aucun contact. Cette opération permet d'éviter d'enregistrer des réponses sans enregistrer les contacts.

Valeur par défaut

180

Valeurs valides

N'importe quel nombre entier supérieur à zéro.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

maxJDBCInsertBatchSize

Description

Nombre maximum d'enregistrements d'un lot JDBC avant de soumettre la requête. Ce n'est pas le nombre maximum d'enregistrements traités en une itération par le module d'historique des réponses et des contacts d'Interact. Pendant chaque itération, le module d'historique des réponses et des contacts d'Interact traite tous les enregistrements disponibles des tables intermédiaires. Cependant, tous ces enregistrements sont fragmentés en maxJDBCInsertSize.

Valeur par défaut

1000

Valeurs valides

N'importe quel nombre entier supérieur à zéro.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

maxJDBCFetchBatchSize

Description

Nombre maximum d'enregistrements d'un lot JDBC à sélectionner dans la base de données intermédiaire. Vous devrez peut-être augmenter cette valeur pour ajuster les performances du module d'historique des réponses et des contacts.

Par exemple, pour traiter deux millions et demi d'enregistrements d'historique des contacts par jour, vous devrez définir

maxJDBCFetchBatchSize sur un nombre supérieur à deux millions et demi afin que tous les enregistrements de la journée puissent être traités.

Vous pourrez alors affecter à maxJDBCFetchChunkSize et maxJDBCInsertBatchSize des valeurs inférieures (dans cet exemple, peut-être 50 000 et 10 000, respectivement). Il est possible que certains enregistrements du lendemain soient également traités, mais ils seront retenus jusqu'au lendemain.

Valeur par défaut

1000

Valeurs valides

N'importe quel nombre entier supérieur à zéro.

maxJDBCFetchChunkSize

Description

La taille maximum d'une tranche de données JDBC lue pendant ETL (extraire, transformer, charger). Dans certains cas, une taille de tranche supérieure à celle d'une insertion peut améliorer la vitesse du processus ETL.

Valeur par défaut

1000

Valeurs valides

N'importe quel nombre entier supérieur à zéro.

deleteProcessedRecords

Description

Spécifie s'ils faut conserver les enregistrements de l'historique des contacts et des réponses après qu'ils aient été traités.

Valeur par défaut

Oui

completionNotificationScript

Description

Spécifie le chemin absolu vers un script à exécuter à l'achèvement du processus ETL. Si vous spécifiez un script, quatre arguments seront transmis au script de notification de l'achèvement : heure de début, heure de fin, nombre total d'enregistrements CH traités, et nombre total d'enregistrements RH traités. L'heure de début et l'heure de fin sont des valeurs numériques représentant le nombre de millisecondes écoulées depuis 1970.

Valeur par défaut

Aucune

fetchSize

Description

Vous permet de définir la fetchSize JDBC lors de la récupération d'enregistrements dans les tables intermédiaires.

Pour ce qui concerne plus particulièrement les bases de données Oracle, définissez le paramètre sur le nombre d'enregistrements que JDBC devrait récupérer lors de chaque parcours sur le réseau. Pour des lots de 100 000 ou plus, essayez 10 000. Prenez garde à ne pas utiliser une valeur trop grande ici, car cela aura un impact sur l'espace mémoire et les gains seront alors insignifiants, voire néfastes.

Valeur par défaut

Aucune

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

Ces propriétés de configuration définissent la source de données du module d'historique des réponses et des contacts d'Interact.

jndiName

Description

Utilisez la propriété `systemTablesDataSource` property pour identifier la source de données Java Naming and Directory Interface (JNDI) définie sur le serveur d'application (WebSphere ou WebLogic) pour les tables d'exécution Interact.

La base de données d'exécution d'Interact est renseignée avec les scripts dll `aci_runtime` et `aci_populate_runtime`. Elle contient également les tables suivantes (entre autres) : `UACI_CHOfferAttrib` et `UACI_DefaultedStat`.

Valeur par défaut

Aucune valeur par défaut définie.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

databaseType

Description

Type de base de données pour la source de données d'exécution d'Interact.

Valeur par défaut

SQLServer

Valeurs valides

SQLServer | Oracle | DB2

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

schemaName

Description

Nom du schéma qui contient les tables intermédiaires du module d'historique des réponses et des contacts. Il doit être identique aux tables de l'environnement d'exécution.

Il n'est pas nécessaire de définir un schéma.

Valeur par défaut

Aucune valeur par défaut définie.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

Ces propriétés de configuration définissent le type de contact d'une campagne qui correspond à un "contact", et ce à des fins de création de rapports ou d'apprentissage.

contacté

Description

Valeur attribuée à la colonne `ContactStatusID` de la table `UA_DtlContactHist` (tables système de Campaign) pour un contact d'offre. La valeur doit être une entrée valide de la table `UA_ContactStatus.` Pour plus de détails sur l'ajout de types de contact, consultez le document *Campaign Administrator's Guide*.

Valeur par défaut

2

Valeurs valides

Un nombre entier supérieur à zéro.

Disponibilité

Cette propriété est applicable uniquement si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

Ces propriétés de configuration définissent les réponses (acceptation ou refus) associées à la création de rapports et à l'apprentissage.

accepter

Description

Valeur attribuée à la colonne `ResponseTypeID` de la table `UA_ResponseHistory` (tables système de Campaign) pour une offre acceptée. La valeur doit être une entrée valide de la table `UA_UsrResponseType.` Vous devez attribuer la valeur `1`(réponse) à la colonne `CountsAsResponse.`

Pour plus de détails sur l'ajout de types de réponse, consultez le document *Campaign - Guide d'administration*.

Valeur par défaut

3

Valeurs valides

Un nombre entier supérieur à zéro.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

rejet

Description

Valeur attribuée à la colonne `ResponseTypeID` de la table `UA_ResponseHistory` (tables système de Campaign) pour une offre refusée. La valeur doit être une entrée valide de la table `UA_UsrResponseType.` Vous devez attribuer la valeur `2` (refus) à la colonne `CountsAsResponse.` Pour plus de détails sur l'ajout de types de réponse, consultez le document *Campaign - Guide d'administration*.

Valeur par défaut

8

Valeurs valides

N'importe quel nombre entier supérieur à zéro.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | report

Ces propriétés de configuration définissent les noms de rapport lors de l'intégration avec Cognos.

interactiveCellPerformanceByOfferReportName

Description

Nom du rapport Performances des cibles interactives par offre. Ce nom doit correspondre au nom du rapport sur le serveur Cognos.

Valeur par défaut

Performances des cibles interactives par offre

treatmentRuleInventoryReportName

Description

Nom du rapport Inventaire des règles de traitement. Ce nom doit correspondre au nom du rapport sur le serveur Cognos.

Valeur par défaut

Inventaire des règles de traitement du canal

deploymentHistoryReportName

Description

Nom du rapport Historique de déploiement. Ce nom doit correspondre au nom du rapport sur le serveur Cognos

Valeur par défaut

Historique de déploiement des canaux

Campaign | partitions | partition[n] | Interact | learning

Ces propriétés de configuration vous permettent de régler le module d'apprentissage intégré.

confidenceLevel

Description

Pourcentage qui indique le degré de confiance que vous souhaitez accorder à l'utilitaire d'apprentissage avant de passer du mode d'exploration au mode d'exploitation. Lorsque la valeur est égale à 0, l'exploration s'arrête.

Cette propriété s'applique uniquement si la propriété `Interact > offerserving > optimizationType` associée à l'environnement d'exécution d'Interact possède la valeur `BuiltInLearning`.

Valeur par défaut

95

Valeurs valides

Nombre entier compris entre 0 et 95, divisible par 5 ou 99.

enableLearning

Description

Si cette propriété a la valeur `Yes`, Interact part du principe que le module d'apprentissage est activé. Si vous affectez à `enableLearning` la valeur `yes`, vous devez configurer `Interact > offerserving > optimizationType` de sorte qu'il ait la valeur `BuiltInLearning` ou `ExternalLearning`.

Si cette propriété a la valeur `No`, Interact part du principe que le module d'apprentissage est désactivé. Si vous affectez la valeur `no` à la propriété `enableLearning`, vous devez configurer `Interact > offerserving > optimizationType` sur `NoLearning`.

Valeur par défaut

Non

maxAttributeNames

Description

Nombre maximum d'attributs d'apprentissage que l'utilitaire d'apprentissage d'Interact doit surveiller.

Cette propriété s'applique uniquement si la propriété `Interact > offerserving > optimizationType` associée à l'environnement d'exécution d'Interact possède la valeur `BuiltInLearning`.

Valeur par défaut

`10`

Valeurs valides

N'importe quel nombre entier.

maxAttributeValues

Description

Nombre maximum de valeurs distinctes que le module d'apprentissage d'Interact suit pour chaque attribut d'apprentissage.

Cette propriété s'applique uniquement si la propriété `Interact > offerserving > optimizationType` associée à l'environnement d'exécution d'Interact possède la valeur `BuiltInLearning`.

Valeur par défaut

5

otherAttributeValue

Description

Nom par défaut de la valeur d'attribut utilisée pour représenter toutes les valeurs d'attributs supérieures à la valeur `maxAttributeValues.`

Cette propriété s'applique uniquement si la propriété `Interact > offerserving > optimizationType` associée à l'environnement d'exécution d'Interact possède la valeur `BuiltInLearning`.

Valeur par défaut

`Autre`

Valeurs valides

Une chaîne ou un nombre.

Exemple

Si la propriété maxAttributeValues est définie sur 3 et que otherAttributeValue est définie sur une autre valeur, le module d'apprentissage suit les trois premières valeurs. Toutes les autres valeurs sont attribuées à l'autre catégorie. Par exemple, vous suivez l'attribut visiteur qui correspond à la couleur de cheveux du visiteur. Les cinq premiers visiteurs ont des cheveux noirs, marron, blonds, roux et gris. L'utilitaire d'apprentissage suit alors les visiteurs aux cheveux noirs, marron et blonds. Les cheveux roux et gris sont regroupés sous la valeur Autre de la propriété `otherAttributeValue.`

percentRandomSelection

Description

Pourcentage qui représente l'intervalle de temps après lequel le module d'apprentissage présente une offre aléatoire. Par exemple, si vous affectez la valeur 5 à la propriété percentRandomSelection, cela signifie que 5 % du temps (5 recommandations sur 100), le module d'apprentissage présente une offre aléatoire.

Valeur par défaut

5

Valeurs valides

N'importe quel nombre entier compris entre 0 et 100.

recencyWeightingFactor

Description

Représentation décimale d'un pourcentage de l'ensemble des données défini par la propriété `recencyWeightingPeriod.` Par exemple une valeur

de 0,15 signifie que 15 % des données utilisées par l'utilitaire d'apprentissage sont issus de la propriété recencyWeightingPeriod.

Cette propriété s'applique uniquement si la propriété Interact > offerserving > optimizationType associée à l'environnement d'exécution d'Interact possède la valeur BuiltInLearning.

Valeur par défaut

0.15

Valeurs valides

Une valeur décimale inférieure à 1.

recencyWeightingPeriod

Description

Taille, exprimée en heures, des données appliquées par le module d'apprentissage au pourcentage de pondération recencyWeightingFactor. Par exemple, la valeur par défaut, 120, signifie que la propriété recencyWeightingFactor des données utilisées par le module d'apprentissage est basée sur les 120 dernières heures.

Cette propriété s'applique uniquement si la propriété optimizationType est définie sur builtInLearning.

Valeur par défaut

120

minPresentCountThreshold

Description

Nombre de fois minimum où une offre doit être présentée avant que ses données soient utilisées dans des calculs et que le module d'apprentissage passe en mode d'exploration.

Valeur par défaut

0

Valeurs valides

Un nombre entier supérieur ou égal à zéro.

enablePruning

Description

Si la valeur est définie sur Yes, le module d'apprentissage d'Interact détermine, via un algorithme, lorsqu'un attribut d'apprentissage (standard ou dynamique) n'est pas prévisible. Quand un attribut n'est pas prévisible, le module d'apprentissage ne le prend pas en compte lorsqu'il pondère une offre. Cela continue jusqu'à ce que le module d'apprentissage regroupe les données d'apprentissage.

Si Non est défini, le module d'apprentissage utilise en permanence tous les attributs d'apprentissage. En n'élaguant pas les attributs non prévisibles, le module d'apprentissage risque de ne pas être aussi précis qu'il pourrait l'être.

Valeur par défaut

Oui

Valeurs valides

Oui | Non

Campaign | partitions | partition[n] | Interact | learning | learningAttributes | [learningAttribute]

Ces propriétés de configuration définissent les attributs d'apprentissage.

attributeName

Description

Chaque propriété `attributeName` correspond au nom d'un attribut visiteur que le module d'apprentissage doit surveiller. Il doit être identique au nom d'une paire nom-valeur de vos données de session.

Cette propriété s'applique uniquement si la propriété `Interact > offerserving > optimizationType` associée à l'environnement d'exécution d'Interact possède la valeur `BuiltInLearning`.

Valeur par défaut

Aucune valeur par défaut définie.

Campaign | partitions | partition[n] | Interact | deployment

Ces propriétés de configuration définissent les paramètres de déploiement.

chunkSize

Description

Taille maximum de fragmentation en Ko pour chaque ensemble de déploiement d'Interact.

Valeur par défaut

500

Disponibilité

Cette propriété est applicable uniquement si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup]

Ces propriétés de configuration définissent les réglages du groupe de serveurs.

serverGroupName

Description

Nom du groupe de serveurs d'exécution d'Interact. Ce nom s'affiche sur l'onglet Synthèse du canal interactif.

Valeur par défaut

Aucune valeur par défaut définie.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

Ces propriétés de configuration définissent les serveurs d'exécution d'Interact.

instanceURL

Description

Adresse URL du serveur d'exécution d'Interact. Un groupe de serveurs peut contenir plusieurs serveurs d'exécution d'Interact ; cependant, chaque serveur doit être créé sous une nouvelle catégorie.

Valeur par défaut

Aucune valeur par défaut définie.

Exemple

`http://serveur:port/interact`

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | flowchart

Ces propriétés de configuration définissent l'environnement d'exécution d'Interact utilisé pour l'exécution en mode test des diagrammes temps réel.

serverGroup

Description

Nom du groupe de serveurs d'Interact utilisé par Campaign pour une exécution en mode test. Ce nom doit correspondre au nom de catégorie créé sous `serverGroups.`

Valeur par défaut

Aucune valeur par défaut définie.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

dataSource

Description

Utilisez la propriété `dataSource` pour identifier la source de données physique utilisée par Campaign lors de l'exécution en mode test des diagrammes temps réel. Cette propriété doit correspondre à la source de données définie par la propriété `Campaign > partitions > partitionN > dataSources` pour la source de données exécutée en mode test et définie pour la phase de conception d'Interact.

Valeur par défaut

Aucune valeur par défaut définie.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

Ces propriétés de configuration définissent le code de cible par défaut de la table des offres par défaut. Vous ne devez configurer ces propriétés que si vous définissez des attributions d'offres globales.

DefaultCellCode

Description

Le code de cible par défaut utilisé par Interact si vous ne définissez aucun code de cible dans la table des offres par défaut.

Valeur par défaut

Aucune valeur par défaut définie.

Valeurs valides

Chaîne qui correspond au format du code de cible défini dans Campaign

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

Ces propriétés de configuration définissent le code de cible par défaut de la table offersBySQL. Vous ne devez configurer ces propriétés que si vous utilisez des requêtes SQL pour obtenir l'ensemble souhaité d'offres candidates.

DefaultCellCode

Description

Interact utilise le code de cible par défaut pour toute offre présente dans le ou les tables OffersBySQL comportant une valeur Null dans la colonne du code de cible (ou si la colonne du code de cible est manquante). Cette valeur doit être un code de cible valide.

Valeur par défaut

Aucune valeur par défaut définie.

Valeurs valides

Chaîne qui correspond au format du code de cible défini dans Campaign

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

Ces propriétés de configuration définissent le code de cible par défaut de la table de remplacement de score. Vous ne devez configurer ces propriétés que si vous définissez des attributions d'offres individuelles.

DefaultCellCode

Description

Le code de cible par défaut utilisé par Interact si vous ne définissez aucun code de cible dans la table de remplacement des scores.

Valeur par défaut

Aucune valeur par défaut définie.

Valeurs valides

Chaîne qui correspond au format du code de cible défini dans Campaign

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Campaign | partitions | partition[n] | server | internal

Les propriétés de cette catégorie spécifient des paramètres d'intégration et les limites internalID de la partition Campaign sélectionnée. Si votre installation de Campaign comporte plusieurs partitions, définissez ces propriétés pour chaque partition que vous souhaitez affecter.

internalIdLowerLimit

Description

Les propriétés `internalIdUpperLimit` et `internalIdLowerLimit` permettent de limiter les ID internes Campaign à une plage spécifiée. Notez que les valeurs sont inclusives : Campaign peut utiliser les limites inférieure et supérieure.

Valeur par défaut

`0` (zéro)

internalIdUpperLimit

Description

Les propriétés `internalIdUpperLimit` et `internalIdLowerLimit` permettent de limiter les ID internes Campaign à une plage spécifiée. Notez que les valeurs sont inclusives : Campaign peut utiliser les limites inférieure et supérieure.

Valeur par défaut

`4294967295`

eMessageInstalled

Description

Indique qu'eMessage est installé. Lorsque vous sélectionnez yes, les fonctions d'eMessage sont disponibles dans l'interface Campaign.

Le programme d'installation d'IBM définit cette propriété sur yes pour la partition par défaut de votre installation eMessage. Pour les autres partitions sur lesquelles vous avez installé eMessage, vous devez configurer manuellement cette propriété.

Valeur par défaut

no

Valeurs valides

yes | no

interactInstalled

Description

Après l'installation de l'environnement de conception Interact, cette propriété de configuration doit être définie sur yes pour activer l'environnement de conception Interact dans Campaign.

Si Interact n'est pas installé, indiquez no. Le fait de définir cette propriété sur no ne supprime pas les menus et options Interact de l'interface utilisateur. Pour supprimer les menus et les options, vous devez désenregistrer manuellement Interact à l'aide de l'utilitaire configTool.

Valeur par défaut

no

Valeurs valides

yes | no

Availability

Cette propriété est applicable uniquement si vous avez installé Interact.

MO_UC_integration

Description

Permet l'intégration à Opérations marketing pour cette partition. Si vous envisagez de paramétrer l'une des trois options ci-dessous sur Oui, vous devez paramétrer **MO_UC_integration** sur Oui. Pour plus d'informations sur la configuration de cette intégration, voir *IBM Unica Opérations marketing and Campaign Integration Guide*.

Valeur par défaut

no

Valeurs valides

yes | no

MO_UC_BottomUpTargetCells

Description

Autorise les cellules ascendantes la liste des populations ciblées sur cette partition. Lorsque cette propriété a pour valeur Yes, les populations ciblées descendantes et ascendantes sont visibles, mais les populations ciblées ascendantes sont en lecture seule. Il est à noter que **MO_UC_integration**

doit être activé. Pour plus d'informations sur la configuration de cette intégration, voir *IBM Unica Opérations marketing and Campaign Integration Guide*.

Valeur par défaut

no

Valeurs valides

yes | no

Legacy_campaigns

Description

Si la propriété **MO_UC_integration** est définie sur **Yes**, la propriété **Legacy_campaigns** permet d'accéder aux campagnes créées avant que l'intégration soit rendue possible, notamment celles créées dans Campaign 7.x et associées aux projets Plan 7.x. Pour plus d'informations sur la configuration de cette intégration, voir *IBM Unica Opérations marketing and Campaign Integration Guide*.

Valeur par défaut

no

Valeurs valides

yes | no

IBM Unica Opérations marketing - Intégration à Offer

Description

Permet d'utiliser Opérations marketing pour effectuer des tâches de gestion de cycle de vie d'offre sur cette partition. (**MO_UC_integration** doit être activé. **Campaign integration** doit également être activé dans **Paramètres > Configuration > Unica > Plate-forme**.) Pour plus d'informations sur la configuration de cette intégration, voir *IBM Unica Opérations marketing and Campaign Integration Guide*.

Valeur par défaut

no

Valeurs valides

yes | no

UC_CM_integration

Description

Permet l'intégration de segment en ligne IBM Coremetrics pour une partition Campaign. Si vous définissez cette option sur yes, la zone du Processus Sélection d'un diagramme fournit la possibilité de sélectionner **IBM Coremetrics Segments** en entrée. Pour configurer l'intégration de chaque partition, choisissez **Paramètres > Configuration > Campaign | partitions | partition[n] | Coremetrics**.

Valeur par défaut

no

Valeurs valides

yes | no

Campaign | monitoring

Les propriétés de cette catégorie indiquent si la fonction Suivi opérationnel est activée et spécifient l'URL du serveur Operational Monitoring ainsi que le comportement de la mémoire cache. Le suivi opérationnel affiche les diagrammes actifs et permet de les contrôler.

cacheCleanupInterval

Description

La propriété `cacheCleanupInterval` spécifie l'intervalle, en secondes, entre chaque nettoyage automatique, dans la mémoire cache, des données associées à l'état du diagramme.

Cette propriété n'est pas disponible dans les versions antérieures à Campaign 7.0.

Valeur par défaut

`600` (10 minutes)

cacheRunCompleteTime

Description

La propriété `cacheRunCompleteTime` indique combien de temps, en minutes, il faut aux sessions pour être enregistrées dans la mémoire cache et affichées sur la page Suivi.

Cette propriété n'est pas disponible dans les versions antérieures à Campaign 7.0.

Valeur par défaut

`4320`

monitorEnabled

Description

La propriété `monitorEnabled` indique si le suivi est activé.

Cette propriété n'est pas disponible dans les versions antérieures à Campaign 7.0.

Valeur par défaut

`oui`

serverURL

Description

La propriété `Campaign > monitoring > serverURL` spécifie l'URL du serveur Operational Monitoring. Ce réglage est obligatoire. Modifiez la valeur si l'URL du serveur Operational Monitoring n'est pas celui par défaut.

Si Campaign est configuré afin d'utiliser les communications SSL (Secure Sockets Layer), affectez à cette propriété la valeur HTTPS. Par exemple : `serverURL=https://`*`host:SSL_port`*`/Campaign/OperationMonitor`, où :

- *host* est le nom ou l'adresse IP de la machine sur laquelle l'application Web est installée.
- *port_SSL* représente le port SSL de l'application Web.

Notez que l'URL commence par `https`.

Valeur par défaut

`http://localhost:7001/Campaign/OperationMonitor`

monitorEnabledForInteract

Description

Si la valeur est définie sur `oui`, le serveur de connecteurs JMX Campaign d'Interact est activé. Campaign n'a pas de sécurité JMX.

Si la valeur est définie sur `non`, vous ne pouvez pas vous connecter au serveur de connecteurs JMX de Campaign.

Cette surveillance JMX est destinée uniquement au module d'historique des réponses et des contacts d'Interact.

Valeur par défaut

False

Valeurs valides

True | False

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

protocole

Description

Protocole d'écoute du serveur de connecteurs JMX de Campaign, si la propriété `monitorEnabledForInteract` est définie sur oui.

Cette surveillance JMX est destinée uniquement au module d'historique des réponses et des contacts d'Interact.

Valeur par défaut

JMXMP

Valeurs valides

JMXMP | RMI

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

port

Description

Port d'écoute du serveur de connecteurs JMX de Campaign, si la propriété `monitorEnabledForInteract` est définie sur oui.

Cette surveillance JMX est destinée uniquement au module d'historique des réponses et des contacts d'Interact.

Valeur par défaut

2004

Valeurs valides

Un nombre entier entre 1 025 et 65 535.

Disponibilité

Cette propriété ne s'applique que si vous avez installé Interact.

Annexe D. Personnalisation d'offre en temps réel côté client

Dans certaines situations, il peut être souhaitable de fournir une personnalisation d'offre en temps réel sans implémentation de code Java détaillé ni d'appels SOAP au serveur Interact. Par exemple, lorsqu'un visiteur charge initialement une page Web dans laquelle le contenu Javascript est votre seule programmation étendue disponible, ou lorsqu'un visiteur ouvre un message email dans lequel seul du contenu HTML est possible. IBM Unica Interact fournit plusieurs connecteurs qui fournissent une gestion en temps réel dans les cas où vous contrôlez uniquement le contenu Web chargé côté client, ou lorsque vous voulez simplifier l'interface Interact.

Votre installation Interact inclut deux connecteurs pour la personnalisation offre lancée côté client :

- «A propos de Interact Message Connector». Avec Message Connector, le contenu Web des messages électroniques (par exemple) ou des autres médias électroniques peut comporter des balises d'image et de lien pour effectuer des appels au serveur Interact pour la présentation d'offres lors d'un chargement de page et les pages d'arrivée atteintes par une série de clics.
- «A propos de Interact Web Connector», à la page 233. Avec Message Connector, (également appelé JS Avec Message Connector), les pages Web peuvent utiliser duJavaScript côté client pour gérer l'arbitrage et la présentation d'offres, et l'historique des contacts et des réponses via présentation d'offres lors d'un chargement de page et les pages d'arrivée atteintes par une série de clics.

A propos de Interact Message Connector

Interact Message Connector permet aux messages électroniques et aux autres médias électroniques d'appeler IBM Unica Interact afin qu'il autorise la présentation d'offres personnalisées lors de l'ouverture, et lorsque le client clique sur le message pour accéder au site indiqué. Cette suppression est réalisée avec deux balises principales : La balise image (`IMG`), qui charge les offres personnalisées lors de l'ouverture, et la balise de lien (`A`), qui capture des informations sur les clics et redirige le client vers une page d'arrivée spécifique.

Exemple

L'exemple suivant montre une partie du code HTML que vous pouvez inclure dans un spot marketing (par exemple, dans un message électronique). Il contient à la fois une URL de balise `IMG` qui transmet les informations lorsque le document s'ouvre sur le serveur Interact et extrait l'image de l'offre appropriée en réponse. Il contient aussi une URL de balise `A` qui détermine quelles informations sont transmises au serveur Interact suite aux clics :

```
<a href="http://www.example.com/MessageConnector/
offerClickthru.jsp?msgId=1234&linkId=1&userid=1&referral=test"><img
src="http:// www.example.com/MessageConnector/offerImage.jsp?msgId=1234
&linkId=1&userid=1&incomeLevel=5&incomeType=numeric"/></a>
```

Dans l'exemple suivant, une balise `IMG` est imbriquée dans une balise `A`, et génère le comportement suivant :

© Copyright IBM Corp. 2001, 2012

1. Lorsque le message électronique est ouvert, Message Connector reçoit une requête contenant les informations codées dans la balise IMG : le msgID et le linkID de ce message, et les paramètres de client qui incluent l'ID utilisateur, le niveau de revenu, et le type de revenu.
2. Ces informations sont transmises via un appel d'API au serveur d'exécution Interact.
3. Le serveur d'exécution renvoie une offre à Message Connector,qui extrait l'adresse URL de l'image de l'offre, et fournit cette URL (avec les paramètres supplémentaires inclus) puis redirige la demande d'image à l'URL de cette offre.
4. Le client voit l'offre comme une image.

A ce stade, le client peut cliquer sur cette image pour répondre à l'offre d'une certaine manière. Ce clic utilisant la balise A et son attribut HREF spécifié (qui indique l'URL de destination) envoie une autre requête au Message Connector en lui demandant une page d'accueil liée à l'adresse URL de cette offre. Le navigateur du client est alors redirigé vers la page d'arrivée telle qu'elle est configurée dans l'offre.

Notez qu'un clic sur une balise A n'est pas strictement nécessaire : l'offre peut contenir une image uniquement, tel qu'un bon de réduction que le client peut imprimer.

Installation de Message Connector

Les fichiers dont vous avez besoin pour installer, déployer et exécuter Message Connector ont été fournis automatiquement avec votre installation de serveur d'exécution IBM Unica Interact. Cette section résume les étapes nécessaires pour que le Message Connector soit prêt à être utilisé.

L'installation et déploiement de Message Connector impliquent les tâches suivantes :

- Le cas échéant, configurez les paramètres par défaut pour Message Connector comme décrit dans «Configuration de Message Connector».
- Création des tables de base de données nécessaires pour stocker les données de transaction de Message Connector, comme décrit dans «Création des tables de Message Connector», à la page 229.
- Installation de l'application Web Message Connector comme indiqué dans «Déploiement et exécution de Message Connector», à la page 230.
- Création des balises IMG et A dans vos spots marketing (emails ou pages Web, par exemple) nécessaires pour appeler les offres Message Connector lors de l'ouverture et des clics, comme décrit dans «Création des liens de Message Connector», à la page 231.

Configuration de Message Connector

Avant de déployer Message Connector, vous devez personnaliser le fichier de configuration fourni avec votre installation pour qu'il corresponde à votre environnement spécifique. Vous pouvez modifier le fichier XML appelé MessageConnectorConfig.xml qui se trouve dans le répertoire Message Connector sur le serveur d'exécution Interact, comme pour <Interact_home>/msgconnector/config/MessageConnectorConfig.xml.

Le fichier MessageConnectorConfig.xml contient certains paramètres de configuration qui sont obligatoires, d'autres qui sont facultatifs. Les paramètres que

vous utilisez doivent être personnalisés pour votre installation spécifique. Procédez comme indiqué ici pour modifier la configuration.

1. Si Message Connector est déployé et en cours d'exécution sur votre serveur d'applications Web, annulez son déploiement avant de poursuivre.
2. Sur le serveur d'exécution Interact, ouvrez le fichier `MessageConnectorConfig.xml` dans n'importe quel éditeur de texte ou éditeur XML.
3. Modifiez les paramètres de configuration comme il convient, en vérifiant que les paramètres *obligatoires* suivants sont corrects pour votre installation.
   - `<interactUrl>`, l'URL du serveur d'exécution Interact auquel les balises de la page Message Connector doivent se connecter, et sur lequel Message Connector s'exécute.
   - `<imageErrorLink>`, l'URL vers laquelle Message Connector se redirige si une erreur se produit lors du traitement de la demande d'une image d'offre.
   - `<landingPageErrorLink>`, l'URL vers laquelle Message Connector se redirige si une erreur se produit lors du traitement de la demande de la page d'arrivée d'une offre.
   - `<audienceLevels>`, une section du fichier de configuration qui contient un ou plusieurs ensembles de référentiel, et qui indique le référentiel par défaut si aucun n'est spécifié par le lien de Message Connector. Au moins un référentiel doit être configuré.

   Tous les paramètres de configuration sont décrits plus en détail dans «Paramètres de configuration de Message Connector».
4. Lorsque vous avez terminé les modifications de configuration, enregistrez et fermez le fichier `MessageConnectorConfig.xml`.
5. Poursuivez la configuration et le déploiement de Message Connector.

Paramètres de configuration de Message Connector :

Pour configurer Message Connector, vous pouvez modifier le fichier XML appelé `MessageConnectorConfig.xml` qui se trouve dans votre répertoire Message Connector sur le serveur d'exécution Interact, en général `<Interact_home>/msgconnector/config/MessageConnectorConfig.xml`. Chacune des configurations de ce fichier XML sont décrites ici. N'oubliez pas que si vous modifiez ce fichier une fois que Message Connector est déployé et en cours d'exécution, vous devez annuler le déploiement puis redéployer Message Connector ou redémarrer le serveur d'applications pour recharger ces paramètres lorsque vous avez fini de modifier le fichier.

Paramètres généraux

Le tableau suivant contient la liste des paramètres facultatifs et obligatoires contenus dans la section `generalSettings` du fichier `MessageConnectorConfig.xml`.

Tableau 20. Paramètres généraux de Message Connector

Elément	Description	Valeur par défaut
<interactURL>	Adresse URL du serveur d'exécution Interact qui gère les appels à partir des balises de page de Message Connector, tels que le serveur d'exécution sur lequel Message Connector s'exécute. Cet élément est obligatoire.	http://localhost:7001/interact
<defaultDateTimeFormat>	Format de date par défaut.	JJ/MM/aaaa
<log4jConfigFileLocation>	Emplacement du fichier de propriétés Log4j. Il est relatif à la variable d'environnement *$MESSAGE_CONNECTOR_HOME* si elle est définie ; sinon, cette valeur est relative au chemin de la racine de l'application Web Message Connector.	config/ MessageConnectorLog4j.properties

Valeurs de paramètre par défaut

Le tableau suivant contient la liste des paramètres facultatifs et obligatoires contenus dans la section defaultParameterValues du fichier MessageConnectorConfig.xml.

Tableau 21. Paramètres par défaut de Message Connector

Elément	Description	Valeur par défaut
<interactiveChannel>	Nom du canal interactif par défaut.	
<interactionPoint>	Nom du point d'interaction par défaut.	
<debugFlag>	Détermine si le débogage est activé. Les valeurs admises sont true et false.	false
<contactEventName>	Nom par défaut de l'événement de contact publié.	
<acceptEventName>	Nom par défaut de l'événement d'acceptation publié.	
<imageUrlAttribute>	Nom d'attribut de l'offre par défaut qui contient l'adresse URL de l'image de l'offre, si aucun n'est indiqué dans le lien Message Connector.	
<landingPageUrlAttribute>	Adresse URL par défaut de la page d'accueil de clics si aucune n'est spécifiée dans le lien de Message Connector.	

Paramètres de comportement

Le tableau suivant contient la liste des paramètres facultatifs et obligatoires contenus dans la section behaviorSettings du fichier MessageConnectorConfig.xml.

Tableau 22. Paramètres de comportement de Message Connector

Elément	Description	Valeur par défaut
<imageErrorLink>	URL vers laquelle le connecteur redirige si une erreur se produit lors du traitement d'une requête d'une image d'offre. Ce paramètre est obligatoire.	/images/default.jpg
<landingPageErrorLink>	URL vers laquelle le connecteur redirige si une erreur se produit lors du traitement d'une requête d'une page d'accueil de clics. Ce paramètre est obligatoire.	/jsp/default.jsp
<alwaysUseExistingOffer>	Détermine si l'offre en mémoire cache doit être renvoyée, même si elle a déjà expiré. Les valeurs admises sont true et false.	false
<offerExpireAction>	L'action à entreprendre si l'offre initiale est trouvé, mais a déjà expiré. Les valeurs autorisées sont les suivantes : • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

Paramètres de stockage

Le tableau suivant contient la liste des paramètres facultatifs et obligatoires contenus dans la section storageSettings du fichier MessageConnectorConfig.xml.

Tableau 23. Paramètres de stockage de MessageConnector

Elément	Description	Valeur par défaut
<persistenceMode>	Lorsque le cache persiste de nouvelles entrées dans la base de données. Les valeurs autorisées sont WRITE-DERRIERE (où les données sont écrites initialement dans la mémoire cache et mises à jour ultérieurement dans la base de données) et WRITE-THROUGH (où les données sont écrites dans la mémoire cache et dans la base de données en même temps).	WRITE-THROUGH
<maxCacheSize>	Nombre maximal d'entrées dans le cache mémoire.	5000
<maxPersistenceBatchSize>	Taille de lot maximale lors de la conservation des entrées dans la base de données.	200
<macCachePersistInterval>	Durée maximale en secondes pendant laquelle une entrée demeure dans la mémoire cache avant d'être conservée dans la base de données.	3
<maxElementOnDisk>	Nombre maximal d'entrées dans le cache disque.	5000

Tableau 23. Paramètres de stockage de MessageConnector (suite)

Elément	Description	Valeur par défaut
<cacheEntryTimeToExpireInSeconds>	Durée maximale pendant laquelle les entrées présentes dans le cache disque sont conservées avant expiration.	60000
<jdbcSettings>	Section du fichier XML contenant des informations spécifiques si une connexion JDBC est utilisée. Elle s'exclut mutuellement avec la section <dataSourceSettings>.	Configuré par défaut pour se connecter à une base de données SQL Server configurée sur le serveur local, mais si vous activez cette section, vous devez fournir les paramètres JDBC réels et les données d'identification pour la connexion.
<dataSourceSettings>	Section du fichier XML contenant des informations spécifiques si une source de données est utilisée. Elle s'exclut mutuellement avec la section <jdbcSettings>.	Configuré par défaut pour se connecter à la source de données InteractDS définie sur le serveur d'applications Web local.

Référentiels

Le tableau suivant contient la liste des paramètres facultatifs et obligatoires contenus dans la section audienceLevels du fichier MessageConnectorConfig.xml.

Notez que l'élément audienceLevels est utilisé facultativement pour spécifier le référentiel par défaut à utiliser si aucun n'est spécifié dans le lien du connecteur de Message Connector, comme dans l'exemple suivant :

```
<audienceLevels default="Customer">
```

Dans cet exemple, la valeur de l'attribut par défaut correspond au nom d'un audienceLevel défini dans cette section. Il doit y avoir au moins un référentiel défini dans ce fichier de configuration.

Tableau 24. Paramètres de référentiel MessageConnector

Elément	Elément	Description	Valeur par défaut
<audienceLevel>		Elément contenant la configuration du référentiel. Fournissez un attribut de nom, comme dans <audienceLevel name="Customer">	
	<messageLogTable>	Nom de la table de journal. Cette valeur est obligatoire.	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<field>	Définition d'une ou de plusieurs zones d'ID référentiel pour ce audienceLevel.	
	<name>	Nom de la zone d'ID référentiel, comme indiqué dans l'exécution Interact.	
	<httpParameterName>	Nom du paramètre correspondant à cette zone d'ID référentiel.	
	<dbColumnName>	Nom de la colonne correspondante dans la base de données pour cette zone d'ID référentiel.	

Tableau 24. Paramètres de référentiel MessageConnector (suite)

Elément	Elément	Description	Valeur par défaut
	<type>	Type de la zone d'ID référentiel, tel qu'il est indiqué dans l'exécution Interact. Les valeurs peuvent être string ou numeric.	

Création des tables de Message Connector

Avant de déployer IBM Unica Interact Message Connector, vous devez d'abord créer les tables dans la base de données dans lequel sont stockées les données d'exécution Interact. Vous devez créer une table pour chaque référentiel que vous avez défini. Pour chaque référentiel, Interact utilise les tables que vous créez pour enregistrer des informations sur les transactions de Message Connector.

Utilisez le client de base de données pour exécuter le script SQL de Message Connector sur la base de données ou le schéma approprié et créer les tables utilisateur requises. Les scripts SQL de votre base de données prise en charge sont installés automatiquement lorsque vous installez le serveur d'exécution Interact. Consultez les feuilles de travail que vous avez complétées dans le *Guide d'installation IBM Unica Interact* pour plus de détails sur la connexion à la base de données contenant les tables d'exécution Interact.

1. Lancez votre client de base de données et connectez-vous à la base de données dans laquelle vos tables d'exécution Interact sont stockées.
2. Exécutez le script approprié dans le répertoire <*Interact_home*>/msgconnector/scripts/ddl (où <*Interact_home*> est le répertoire dans lequel vous avez installé l'exécution Interact, par exemple C:\Unica\Interact ou /Unica/Interact). Le tableau suivant répertorie les exemples de scripts SQL que vous pouvez utiliser pour créer manuellement les tables Message Connector suivantes :

Tableau 25. Scripts de création des tables Message Connector

Type de la source de données	Nom du script
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

Notez que ces scripts sont fournis comme exemples. Si vous utilisez une autre convention de dénomination ou une autre structure pour les valeurs d'ID de référentiel, il se peut que vous deviez modifier le script avant de l'exécuter. En règle générale, il est conseillé d'avoir une seule table dédiée à chaque référentiel.

Les tables créées doivent contenir les informations suivantes :

Tableau 26. Informations créées par les exemple de scripts SQL

Nom de la colonne	Description
LogId	Clé primaire de cette entrée.
MessageId	Identificateur unique de chaque instance de messagerie.
LinkId	Identificateur unique de chaque lien dans le support électronique (tel qu'un message électronique).
OfferImageUrl	Adresse URL d'accès à l'image associée à l'offre renvoyée.

Tableau 26. Informations créées par les exemple de scripts SQL (suite)

Nom de la colonne	Description
OfferLandingPageUrl	Adresse URL d'accès à la page d'accueil associée à l'offre renvoyée.
TreatmentCode	Code de traitement de l'offre renvoyée.
OfferExpirationDate	Date et heure d'expiration de l'offre renvoyée.
OfferContactDate	Date et l'heure du renvoi de l'offre au client.
AudienceId	ID de référentiel du support électronique.

Notez ce qui suit concernant cette table :

- Selon le référentiel, il existe une seule colonne AudienceId pour chaque composant de la clé de référentiel.
- La combinaison des formulaires MessageId, LinkId et AudienceId(s) forme la clé unique de cette table.

Lorsque l'exécution du script est terminée, vous avez créé les tables nécessaires pour Message Connector.

Vous êtes maintenant prêt à déployer l'application Web Message Connector.

Déploiement et exécution de Message Connector

IBM Unica Interact Message Connector est déployé sous la forme d'une application Web autonome sur un serveur d'applications Web pris en charge.

Avant de déployer Message Connector, vérifiez que les tâches suivantes ont été exécutées :

- Vous devez avoir installé le serveur d'exécution IBM Unica Interact. L'application déployable Message Connector est automatiquement installée avec le serveur d'exécution, et est prête à être déployée à partir de votre répertoire de base Interact.
- Vous devez également avoir exécuté les scripts SQL fournis avec votre installation pour créer les tables nécessaires dans la base de données d'exécution Interact, en vue d'une utilisation par Message Connector comme indiqué dans «Création des tables de Message Connector», à la page 229

De la même manière que vous déployez d'autres applications IBM Unica sur un serveur d'applications Web avant de pouvoir les exécuter, vous devez déployer l'application Message Connector pour la rendre disponible et lui permettre de proposer les offres.

1. Connectez-vous à l'interface de gestion de votre serveur d'applications Web avec les droits nécessaires pour déployer une application.
2. Suivez les instructions de votre serveur d'applications Web pour déployer et exécuter le fichier *<Interact_home>*/msgconnector/MessageConnector.war Remplacez *<Interact_home>* par le répertoire réel sur lequel le serveur d'exécution Interact est installé, par exemple C:\Unica\Interact, ou/Unica/Interact.

Message Connector est désormais prêt à être utilisé. Lorsque vous avez configuré votre installation Interact pour créer les données sous-jacentes que Message Connector utilisera pour fournir des offres, tels que les canaux interactifs, les stratégies, les diagrammes, les offres, etc., vous pouvez créer les liens dans votre support électronique que Message Connector acceptera.

Création des liens de Message Connector

Pour utiliser Message Connector pour fournir des images d'offre personnalisées lorsqu'un utilisateur final interagit avec vos supports électroniques (par exemple en ouvrant un message électronique), et les pages d'accueil personnalisées lorsque l'utilisateur final clique pour accéder l'offre, vous devez créer les liens à incorporer dans votre message. Cette section contient un résumé des balises HTML de ces liens.

Quel que soit le système que vous utilisez pour générer vos messages sortants vers les utilisateurs finaux, vous devez générer le balisage HTML destiné à contenir les zones appropriées (fournies dans les balises HTML sous forme d'attributs) contenant les informations que vous souhaitez transmettre au serveur d'exécution Interact. Procédez comme indiqué ci-dessous pour configurer les informations minimales requises pour un message Message Connector.

Notez que, bien que les présentes instructions fassent spécifiquement référence aux messages contenant des liens Message Connector, vous pouvez appliquer la même procédure et la même configuration pour ajouter des liens aux pages Web ou à tout autre support électronique.

1. Créez le lien `IMG` qui apparaîtra dans votre message avec, au minimum, les paramètres suivants :
   - `msgID`, indiquant l'identificateur unique de ce message.
   - `linkID`, indiquant l'identificateur unique du lien dans le message.
   - `audienceID`, l'identificateur du référentiel auquel le destinataire du message appartient.

     Notez que si l'ID référentiel est un ID composite, tous ces composants doivent être inclus dans le lien.

   Vous pouvez également inclure des paramètres facultatifs qui comprennent le référentiel, le nom du canal interactif, le nom du point d'interaction, l'adresse URL de l'emplacement de l'image, ainsi que vos propres paramètres personnalisés non spécifiquement utilisés par Message Connector.
2. (Facultatif) Créez un lien `A` qui englobe le lien `IMG` de sorte que, lorsque l'utilisateur clique sur l'image, le navigateur charge une page contenant l'offre destinée à l'utilisateur. Le lien `A` doit également contenir les trois paramètres ci-dessus (`msgID`, `linkID`, et `audienceID`), ainsi que tous les paramètres facultatifs (référentiel, nom du canal interactif, le nom du point d'interaction) et des paramètres personnalisés non spécifiquement utilisés par Message Connector. Notez que le lien `A` va probablement contenir un lien `IMG` de Message Connector, mais il peut également fonctionner en autonome sur la page selon les besoins. Si le lien ne contient pas de lien `IMG`, le lien `IMG` doit contenir le même ensemble de paramètres que le lien `A` qui le contient (y compris les paramètres facultatifs ou et personnalisés).
3. Lorsque les liens sont correctement définis, générez et envoyez les messages électroniques.

Pour obtenir des informations détaillées sur les paramètres disponibles, ainsi que des exemples de liens, voir «Paramètres de demande HTTP "IMG" et"A"»

Paramètres de demande HTTP "IMG" et"A"

Lorsque Message Connector reçoit une demande, soit parce qu'un utilisateur final a ouvert un courrier électronique contenant une balise `IMG` encodée par Message Connector, soit parce que utilisateur final a cliqué jusqu'à atteindre une balise `A`, il analyse les paramètres fournis avec la demande afin de renvoyer les données

d'offre appropriées. Cette section fournit la liste des paramètres pouvant être inclus dans l'adresse URL de demande (soit la balise IMG (chargée automatiquement lorsqu'une image balisée est affichée lors de l'ouverture du courrier électronique), soit la balise A (chargée lorsque la personne affichant le courrier électronique clique sur le message jusqu'à atteindre le site spécifié).

Paramètres

Lorsque Message Connector reçoit une demande, il analyse les paramètres fournis avec la demande. Ces paramètres contiennent tout ou une partie de ce qui suit :

Nom du paramètre	Description	Obligatoire ?	Valeur par défaut
msgId	Identificateur unique de l'instance de courrier électronique ou d'une page Web.	Oui	Aucun. Fourni par le système créant l'instance unique du message électronique ou la page Web contenant la balise.
linkId	Identificateur unique du lien dans ce courrier électronique ou la page Web.	Oui	Aucun. Fourni par le système créant l'instance unique du message électronique ou la page Web contenant la balise.
audienceLevel	Niveau de référentiel auquel le destinataire du cette communication appartient.	Non	L'audienceLevel indiqué par défaut dans l'élément audienceLevels trouvé dans le fichier MessageConnectorConfig.xml.
ic	Nom du canal interactif cible (IC)	Non	Valeur de l'élément interactiveChannel trouvé dans la section defaultParameterValues du fichier MessageConnectorConfig.xml, qui est "interactiveChannel" par défaut.
ip	Nom du point d'interaction (IP).	Non	Valeur de l'élément interactionPoint trouvé dans la section defaultParameterValues du fichier MessageConnectorConfig.xml, qui est par défaut "headBanner".
offerImageUrl	Adresse URL de l'image offre cible correspondant à l'adresse URL IMG dans le message.	Non	Aucun.
offerImageUrlAttr	Nom de l'attribut de l'offre contenant l'URL de l'image cible offre.	Non	Valeur de l'élément imageUrlAttribute trouvé dans la section defaultParameterValues du fichier MessageConnectorConfig.xml.
offerLandingPageUrl	URL de la page d'accueil correspondant à l'offre cible.	Non	Aucun.
offerLandingPageUrlAttr	Nom de l'attribut de l'offre contenant l'URL de la page d'accueil correspondant à l'offre cible.	Non	Valeur de l'élément landingPageUrlAttribute trouvé dans la section defaultParameterValues du fichier MessageConnectorConfig.xml.
contactEvent	Nom de l'événement du contact.	Non	Valeur de l'élément contactEventName trouvé dans la section defaultParameterValues du fichier MessageConnectorConfig.xml, qui est "contact" par défaut.
responseEvent	Nom de l'événement d'acceptation.	Non	Valeur de l'élément acceptEventName trouvé dans la section defaultParameterValues du fichier MessageConnectorConfig.xml, qui est par défaut "accept".
débogage	Indicateur de débogage. Définissez ce paramètre sur "true" uniquement pour l'identification des incidents et si vous le support technique IBM Unica vous l'a demandé.	Non	Valeur de l'élément debugFlag trouvé dans la section defaultParameterValues du fichier MessageConnectorConfig.xml, qui est "false" par défaut.
<id de référentiel>	ID de référentiel de cet utilisateur. Le nom de ce paramètre est défini dans le fichier de configuration.	Oui	Aucun.

Lorsque Message Connector reçoit un paramètre non reconnu (c'est-à-dire, n'apparaît pas dans la liste ci-dessus), il est traité dans l'une des deux façons possibles :

- Si un paramètre non reconnu est fourni (par exemple, "attribute", comme dans attribute="attrValue") et s'il existe un paramètre correspondant ayant le même nom plus le mot "Type" (par exemple "attributeType", comme dans attributeType="string"), Message Connector crée un paramètre correspondant Interact et le passe à l'exécution Interact.

Les valeurs du paramètre Type peuvent être les suivantes :

- string
- numeric
- datetime

  Pour un paramètre de type "datetime," Message Connector recherche également des messages pour un paramètre du même nom plus le mot "Pattern" (par exemple, "attributePattern") dont la valeur est un format de date/heure valide. Par exemple, vous pouvez fournir le paramètre `attributePattern="MM/jj/aaaa"`.

  Notez que si vous spécifiez un paramètre de type "datetime" mais ne fournissez pas de modèle de date correspondante, la valeur indiquée dans le fichier de configuration de Message Connector (qui se trouve dans `<installation_directory>/msgconnector/config/MessageConnectorConfig.xml`) sur le serveur Interact est utilisée.

- Si un paramètre non reconnu est fourni et s'il n'existe aucune valeur Type correspondante, Message Connector transmet ce paramètre à l'URL cible de redirection.

Message Connector passe tous les paramètres non reconnus au serveur d'exécution Interact sans les traiter ni les sauvegarder.

Exemple de code de Message Connector

La balise `A` contient un exemple d'ensemble de liens de Message Connector qui peut apparaître dans un message électronique :

```
<a href="http://www.example.com/MessageConnector/offerClickthru.jsp?msgId=234
   &linkId=1&userid=1&referral=xyz">
 <img src="http://www.example.com/MessageConnector/offerImage.jsp?msgId=234&linkId=1
      &userid=1&incomeCode=3&incomeType=numeric"/>
</a>
```

Dans cet exemple, la balise `IMG` charge automatiquement lorsque le message électronique est ouvert. En extrayant l'image à partir de la page spécifiée, le message transmet les paramètres de l'identificateur de message unique (msgID), l'identificateur de lien unique (linkID), et l'identificateur d'utilisateur unique (userid) avec deux paramètres supplémentaires (incomeCode et incomeType) qui doivent être transmis à l'exécution Interact.

La balise `A` fournit l'attribut HREF (Hypertext Reference) qui transforme l'image de l'offre en un lien cliquable dans le message électronique. Si le destinataire du message, lorsqu'il voit l'image, clique jusqu'à atteindre la page d'accueil, l'identificateur de message unique (`msgId`), l'identificateur de lien (`linkId`) et et l'identificateur de l'utilisateur (`userid`) sont transmis au serveur, ainsi qu'un paramètre supplémentaire (`referral`) qui est transmise à l'adresse URL cible de a redirection.

A propos de Interact Web Connector

Interact WebConnector (également appelé JavaScript Connector, ou JSConnector) fournit un service sur le serveur d'exécution Interact qui permet au code JavaScript d'appeler l'API Interact Java. Cela permet aux pages Web d'appeler Interact pour nue personnalisation d'offre en temps réel utilisant uniquement du code JavaScript intégré, sans devoir utiliser les langages de développement Web (tels que Java, PHP, JSP, etc). Par exemple, vous pouvez imbriquer un petit fragment de code JavaScript sur chaque page de votre site Web qui propose des offres

recommandées par Interact. De cette façon, chaque fois que la page se charge, des appels sont envoyés à l'API Interact pour garantir que les meilleures offres s'affichent sur la page de chargement pour le visiteur du site.

Utilisez Web Connector Interact lorsque vous souhaitez proposer des offres aux visiteurs sur une page dont vous ne pouvez pas contrôler l'affichage par programmation côté serveur (comme vous le feriez avec, par exemple, un script PHP ou un autre script basé sur un serveur), mais que vous pouvez cependant intégrer du code JavaScript au contenu de page qui va être exécuté par le navigateur Web du visiteur.

Conseil : Les fichiers Interact Web Connector sont installés automatiquement sur votre serveur d'exécution Interact, dans le répertoire <*Interact_home*>/jsconnector. Dans le répertoire <*Interact_home*>/jsconnector, vous trouverez un fichier ReadMe.txt contenant des notes et des détails importants sur les fonctions de Web Connector, ainsi que des fichiers exemple et du code source de Web Connector, à utiliser pour développer vos propres solutions. Si vous ne trouvez pas d'informations répondant à vos questions, consultez le répertoire jsconnector pour plus d'informations.

Installation de Web Connector sur le serveur d'exécution

Une instance de Web Connector est installée automatiquement avec votre serveur d'exécution IBM Unica Interact et est activée par défaut. Toutefois, il existe certains paramètres que vous devez modifier avant que pouvoir configurer et utiliser Web Connector.

Les paramètres que vous devez modifier avant de pouvoir utiliser le Web Connector installé sur le serveur d'exécution sont ajoutés à la configuration du serveur d'applications Web. Pour cette raison, vous devrez redémarrer le serveur d'applications Web après avoir effectué ces opérations.

1. Pour le serveur d'applications Web sur lequel est installé le serveur d'exécution Interact, définissez les propriétés Java suivantes :

   -DUI_JSCONNECTOR_ENABLE_INPROCESS=true

   -DUI_JSCONNECTOR_HOME=<*jsconnectorHome*>

   Remplacez <*jsconnectorHome*> par le chemin d'accès au répertoire jsconnector sur le serveur d'exécution, qui est <Interact_installation_directory>/jsconnector.

   Par exemple, sur une installation Windows , ce peut être C:\Unica\Interact\jsconnector. Sur un système UNIX , vous pouvez entrer /Unica/Interact/jsconnectorpour cette valeur.

   La manière dont vous définissez les propriétés Java dépend de votre serveur d'applications Web. Par exemple, dans WebLogic, vous modifieriez le fichier startWebLogic.sh ou startWebLogic.cmd pour mettre à jour le paramètre JAVA_OPTIONS, comme dans cet exemple :

   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/jsconnector"

   Dans WebSphere Application Server, vous définiriez cette propriété dans le panneau de la machine virtuelle Java de la console d'administration.

   Pour plus d'informations sur le paramétrage des propriétés Java, consultez la documentation de votre application Web.
2. Redémarrez votre serveur d'applications Web s'il était déjà actif, ou démarrez votre serveur d'applications Web à cette étape, pour vous assurer que les propriétés Java sont utilisées.

Lorsque le serveur d'applications Web a terminé son processus de démarrage, vous avez terminé l'installation de Web Connector sur le serveur d'exécution. L'étape suivante consiste à se connecter à la page Web de configuration à l'adresse `http://<host>:<port>/interact/jsp/WebConnector.jsp`, où *<host>* est le nom du serveur d'exécution Interact, et *<port>* est le port sur lequel Web Connector est à l'écoute, comme indiqué par le serveur d'applications Web.

Installation de Web Connector en tant qu'application Web distincte

Une instance de Web Connector est installée automatiquement avec votre serveur d'exécution IBM Unica Interact et est activée par défaut. Toutefois, vous pouvez également déployer Web Connector comme une application Web propre (par exemple, sur un serveur d'applications Web sur un système séparé) et le configurer afin qu'il communique avec le serveur d'exécution distant Interact.

Ces instructions décrivent le processus de déploiement de Web Connector en tant qu'application Web distincte avec accès à un serveur d'exécution distant Interact.

Avant de déployer Web Connector, vous devez avoir installé le serveur d'exécution IBM Unica Interact, et vous devez disposer d'un serveur d'applications Web sur un autre système disposant d'un accès au réseau (non bloqué par un pare-feu) vers le serveur d'exécution Interact.

1. Copiez le répertoire `jsconnector` contenant les fichiers du Web Connector à partir de votre serveur d'exécution Interact sur le système sur lequel le serveur d'applications Web (tel que WebSphere Application Server) est déjà configuré et en cours d'exécution. Vous trouverez le répertoire `jsconnector` dans votre répertoire d'installation Interact, par exemple `C:\Unica\Interact` ou `/Unica/Interact`.
2. Sur le système sur lequel vous allez déployer l'instance de Web Connector,configurez le fichier `jsconnector/jsconnector.xml` à l'aide d'un éditeur de texte ou d'un éditeur XML pour modifier l'attribut `interactURL`.

   Définie par défaut sur `http://localhost:7001/interact`, mais vous devez la modifier pour qu'elle corresponde à l'URL du serveur d'exécution distant Interact, par exemple `http://runtime.example.com:7011/interact`.

   Après avoir déployé Web Connector, vous pouvez utiliser une interface Web pour personnaliser les paramètres restants dans le fichier `jsconnector.xml`. Pour plus d'informations, voir «Configuration du Web Connector», à la page 236.
3. Pour le serveur d'applications Web sur lequel sera déployé Web Connector, définissez la propriété Java suivante :

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Remplacez *<jsconnectorHome>* par le chemin d'accès réel dans lequel vous avez copié le répertoire `jsconnector` sur le serveur d'applications Web.

   Par exemple, sur une installation Windows , ce peut être `C:\Unica\Interact\jsconnector`. Sur un système UNIX , vous pouvez entrer `/Unica/Interact/jsconnector`pour cette valeur.

   La manière dont vous définissez les propriétés Java dépend de votre serveur d'applications Web. Par exemple, dans WebLogic, vous modifieriez le fichier `startWebLogic.sh` ou `startWebLogic.cmd` pour mettre à jour le paramètre `JAVA_OPTIONS`, comme dans cet exemple :

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
   jsconnector"
   ```

Dans WebSphere Application Server, vous définiriez cette propriété dans le panneau de la machine virtuelle Java de la console d'administration.

Pour plus d'informations sur le paramétrage des propriétés Java, consultez la documentation de votre application Web.

4. Redémarrez votre serveur d'applications Web s'il était déjà actif, ou démarrez votre serveur d'applications Web à cette étape, pour vous assurer que la nouvelle propriété Java est utilisée.

   Attendez que le serveur d'applications Web termine son processus de démarrage avant de continuer.
5. Connectez-vous à l'interface de gestion de votre serveur d'applications Web avec les droits nécessaires pour déployer une application.
6. Suivez les instructions de votre serveur d'applications Web pour déployer et exécuter le fichier suivant : `jsConnector/jsConnector.war`

Web Connector est désormais déployé dans l'application Web. Lorsque votre serveur Interact et avez entièrement configuré et opérationnel, l'étape suivante consiste à vous connecter à la page de Configuration de Web Connector, à l'adresse `http://` *<hôte>*`:` *<port>*`/interact/jsp/WebConnector.jsp`, où *<hôte>* est le système exécutant le serveur d'applications Web sur lequel vous avez déployé Web Connector, et *<port>* est le port sur lequel Web Connector est à l'écoute, comme indiqué par le serveur d'applications Web.

Configuration du Web Connector

Les paramètre de configuration du Web Connector Interact sont enregistrés dans un fichier appelé `jsconnector.xml` qui est stocké sur le système sur lequel le connecteur Web est déployé (tel que le serveur d'exécution Interact lui-même, ou un système distinct exécutant un serveur d'application Web). Vous pouvez éditer le fichier `jsconnector.xml` directement à l'aide d'un éditeur de texte ou l'éditeur XML ; toutefois, un moyen plus facile de configurer quasiment tous les paramètres de configuration disponibles consiste à utiliser la page de configuration de Web Connector à partir de votre navigateur Web.

Avant de pouvoir utiliser l'interface Web pour configurer Web Connector, vous devez installer et déployer l'application Web qui fournit Web Connector. Sur le serveur d'exécution Interact, une instance de Web Connector est installée automatiquement lorsque vous installez et déployez Interact. Sur n'importe quel autre serveur d'applications Web, vous devez installer et déployer l'application Web de Web Connector comme indiqué à la section «Installation de Web Connector en tant qu'application Web distincte», à la page 235.

1. Lancez votre navigateur Web pris en charge et ouvrez une URL similaire à la suivante :

   `http://`*<host>*`:`*<port>*`/interact/jsp/WebConnector.jsp`

   - Remplacez *<host>* par le serveur sur lequel Web Connector s'exécute, par exemple le nom d'hôte du serveur d'exécution ou le nom du serveur sur lequel vous avez déployé une autre instance de Web Connector.
   - Remplacez *<port>* par le numéro de port sur lequel l'application Web de Web Connector est à l'écoute de connexions, et qui correspond en général au port par défaut du serveur d'applications Web.
2. Sur la page Configurations qui s'affiche, renseignez les sections suivantes :

Tableau 27. Récapitulatif des paramètres de configuration de Web Connector.

Section	Paramètres
Paramètres de base	Utilisez la page Paramètres de base pour configurer le comportement global du Web Connector pour le site sur lequel vous allez déployer les pages marquées. Ces paramètres incluent l'adresse URL de base du site, les informations que Interact doit utiliser concernant les visiteurs sur le site, ainsi que les paramètres similaires qui s'appliquent à toutes les pages que vous prévoyez de référencer avec le code du Web Connector. Reportez-vous à «Options de base de configuration de Web Connector», à la page 238 pour plus de détails.
HTML Display Types	Utilisez la page HTML Display Type pour déterminer le code HTML qui sera fourni pour chaque point d'interaction sur la page. Vous pouvez choisir dans la liste des modèles par défaut (les fichiers .flt) qui contiennent une combinaison de code de feuille de style en cascade (CSS), de code HTML, et de code Javascript à utiliser pour chaque point d'interaction. Vous pouvez utiliser les modèles comme indiqué, les personnaliser selon vos besoins, ou créer les vôtres. Les paramètres de configuration de cette page correspondent à la section `interactionPoints` du fichier de configuration `jsconnector.xml`. Reportez-vous à «Types d'affichage HTML de la configuration Web Connector», à la page 240 pour plus de détails.
Enhanced Pages	Enhanced Pages permettent de mapper des paramètres spécifiques à une page avec un modèle d'URL. Par exemple, vous pouvez configurer une page de mappage, de façon à ce que toute adresse URL contenant le texte "index.htm" affiche votre page d'accueil générale, avec des événements spécifique de chargement de page et les points d'interaction définis pour ce mappage. Les paramètres de configuration de cette page correspondent à la section `pageMapping` du fichier de configuration `jsconnector.xml`. Reportez-vous à «Enhanced Pages - Configuration de WebConnector», à la page 243 pour plus de détails.

3. Sur la page Paramètres de base, vérifiez que les paramètres appliqués à tout le site sont valides pour votre installation. (Facultatif) Indiquez le mode de débogage (déconseillé sauf en cas de dépannage d'un problème). Indiquez également l'intégration de balise de page NetInsight et les paramètres par défaut de la plupart des points d'interaction. Cliquez en suite sur le lien HTML Display Types sous Configurations.
4. Sur la page HTML Display Types, procédez comme suit pour ajouter ou modifier des modèles d'affichage qui définissent les points d'interaction sur la page Web du client.

   Par défaut, les modèles d'affichage (fichiers .flt) sont stockés dans `<jsconnector_home>/conf/html`.

   a. Sélectionnez le fichier .flt dans la liste que vous souhaitez examiner ou utiliser en tant que point de départ, ou cliquez sur Add a Type pour créer un nouveau modèle d'interaction vierge à utiliser.

      Les informations sur le contenu du modèle, le cas échéant, s'affichent en face de la liste des modèles.

   b. (Facultatif) Modifiez le nom du modèle dans la zone **File name for this display type**. Pour obtenir un nouveau modèle, mettez à jour `CHANGE_ME.flt` en lui attribuant un nom plus évocateur.

Si vous renommez le modèle ici, Web Connector crée un nouveau fichier portant ce nom lors de la prochaine sauvegarde du modèle. Les modèles sont sauvegardés lorsque vous modifiez le corps du texte, puis accédez à une autre zone.

c. Modifier ou complétez les informations du fragment HTML si nécessaire, y compris les feuilles de style (CSS), le code JavaScript et le code HTML que vous souhaitez inclure. Vous pouvez également inclure des variables qui seront remplacées par les paramètres Interact lors de l'exécution. Par exemple, `${offer.HighlightTitle}` est automatiquement remplacé par le titre de l'offre dans l'emplacement indiqué pour le point d'interaction.

   Utilisez les exemples qui apparaissent sous la zone de fragment HTML pour plus d'indications sur le format de votre feuille de style en cascade (CSS) ou de vos blocs de code JavaScript ou HTML.

5. Utilisez la page Enhanced Pages améliorées si nécessaire pour configurer les mappages de page déterminant la manière dont des modèles d'URL spécifiques sont traités sur les pages.
6. Lorsque vous avez défini les propriétés de configuration, cliquez sur **Roll Out the Changes**. Lorsque vous cliquez sur **Roll Out the Changes**, les actions suivantes sont exécutées :
   - Affiche la balise de page IBM Unica Interact de Web Connector, qui contient le code JavaScript que vous pouvez copier depuis la page Web Connector et insérer dans vos pages Web.
   - Sauvegarde le fichier de configuration du Web Connector existant sur le serveur Interact (le fichier `jsconnector.xml` sur le serveur sur lequel Web Connector est installé) et crée un nouveau fichier de configuration avec les paramètres que vous avez défini.

     Les fichiers de configuration de sauvegarde sont stockés dans `<jsconnector_home>/conf/archive/jsconnector.xml.<date>.<time>`, comme dans `jsconnector.xml.20111113.214933.750-0500` (où la chaîne de données est 20111113 et la chaîne d'heure, y compris l'indicateur de fuseau horaire, est 214933.750-0500)

Vous avez maintenant terminé de configurer Web Connector.

Pour modifier votre configuration, vous pouvez soit revenir au début de cette procédure et la réexécuter en utilisant de nouvelles valeurs, soit ouvrir le fichier de configuration (`<Interact_home>/jsconnector/conf/jsconnector.xml`) dans n'importe quel éditeur de texte ou éditeur XML et le modifier selon les besoins.

Options de base de configuration de Web Connector

Utilisez la page Paramètres de base des configurations de Web Connector pour configurer le comportement global de Web Connector pour le site sur lequel vous allez déployer les pages marquées. Ces paramètres incluent l'adresse URL de base du site, les informations que Interact doit utiliser concernant les visiteurs sur le site, ainsi que les paramètres similaires qui s'appliquent à toutes les pages que vous prévoyez de référencer avec le code du Web Connector.

Paramètres appliqués à tout le site

Les options de configuration des paramètres appliqués à tout le site sont des paramètres globaux qui affectent le comportement général de l'installation de Web Connector que vous configurez. Vous pouvez indiquer les valeurs suivantes :

Tableau 28. Paramètres appliqués à tout le site pour l'installation de Web Connector

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
Interact API URL	URL de base du serveur d'exécution Interact. **Remarque :** Ce paramètre est utilisé uniquement si Web Connector n'est pas en cours d'exécution dans le serveur d'exécution Interact (c'est-à-dire que vous avez l'avez déployé séparément).	*<interactURL>*
Web Connector URL	URL de base utilisée pour générer l'URL de clics.	*<jsConnectorURL>*
Interactive Channel name for the target website	Nom du canal interactif que vous avez défini sur le serveur Interact qui représente ce mappage de page.	*<interactiveChannel>*
Audience Level of Visitors	Référentiel Campaign du visiteur entrant ; utilisé dans l'appel d'API à l'exécution Interact.	*<audienceLevel>*
Audience ID Field Name in the Profile Table	Nom de la zone audienceId field qui sera utilisé dans l'appel d'API Interact. Les identificateurs de référentiel multi-zone ne sont actuellement pas pris en charge.	*<audienceIdField>*
Data type of the Audience ID Field	Type de données de la zone d'ID référentiel ("numérique" ou "chaîne") à utiliser dans l'appel de l'API Interact.	*<audienceIdFieldType>*
Cookie Name that represents the Session ID	Nom du cookie qui contiendra l'ID de session.	*<sessionIdCookie>*
Cookie Name that represents the Visitor ID	Nom du cookie qui contiendra l'ID du visiteur.	*<visitorIdCookie>*

Fonctions facultatives

Les options de configuration des fonctions facultatives sont des paramètres globaux facultatifs pour l'installation du Web Connector que vous configurez. Vous pouvez indiquer les valeurs suivantes :

Tableau 29. Paramètres appliqués à tout le site pour l'installation de Web Connector

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
Enable Debug Mode	Indique (réponse yes ou no) si vous souhaitez utiliser un mode de débogage spécial. Si vous le définissez cette fonction, le contenu renvoyé par Web Connector comprend un appel JavaScript défini sur 'alert' qui informe le client du mappage de page particulier qui vient de se produire. Le client doit disposer d'une entrée dans le fichier spécifié par le paramètre *<authorizedDebugClients>* afin d'obtenir l'alerte.	*<enableDebugMode>*
Authorized Debugging Clients Host File	Chemin d'accès à un fichier contenant la liste des hôtes ou adresses IP (Internet Protocol) qui qualifient le mode débogage. Un nom d'hôte ou une adresse IP de client doit figurer dans le fichier indiqué pour que les informations de débogage soient collectées.	*<authorizedDebugClients>*
Enable NetInsight Page Tag Integration	Indique (avec un yes ou no réponse) si Web Connector doit joindre la balise spécifiée IBM Unica NetInsight à la fin du contenu de la page.	<enableNetInsightTagging>
NetInsight Tag HTML Template File	Modèle HTML/Javascript permettant d'intégrer un appel à la balise NetInsight. En général, vous devez accepter la valeur par défaut, sauf si vous êtes invité à fournir un modèle différent.	<netInsightTag>

Types d'affichage HTML de la configuration Web Connector

Utilisez la page HTML Display Types pour déterminer le code HTML qui sera fourni pour chaque point d'interaction sur la page. Vous pouvez choisir dans la liste des modèles par défaut (les fichiers .flt) qui contiennent une combinaison de code de feuille de style en cascade (CSS), de code HTML, et de code JavaScript à utiliser pour chaque point d'interaction. Vous pouvez utiliser les modèles comme indiqué, les personnaliser selon vos besoins, ou créer les vôtres.

Remarque : Les paramètres de configuration de cette page correspondent à la section interactionPoints du fichier de configuration jsconnector.xml.

Le point d'interaction peut également contenir des marques de réservation (zones) dans lesquels des attributs d'offre peuvent être placés automatiquement. Par exemple, vous pouvez inclure ${offer.TREATMENT_CODE}qui sera remplacé par le code de traitement affecté à cette offre au cours de l'interaction.

Les modèles qui s'affichent sur cette page sont chargées automatiquement à partir des fichiers stockés dans le répertoire <Interact_home>/jsconnector/conf/html du serveur Web Connector. Tous les nouveaux modèles que vous créez ici sont également stockés dans ce répertoire.

Pour utiliser la page HTML Display Types pour afficher ou modifier des modèles existants, sélectionnez le fichier `.flt` dans la liste.

Pour créer un nouveau modèle sur la page HTML Display Types, cliquez sur **Add a Type**.

Quelle que soit la méthode choisie pour créer ou modifier un modèle, les informations suivantes s'affichent en regard de la liste de modèles :

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
File name for this display type	Nom affecté au modèle que vous éditez. Ce nom doit être valide pour le système d'exploitation sur lequel Web Connector est en cours d'exécution ; par exemple, vous ne pouvez pas utiliser une barre oblique (/) dans le nom si le système d'exploitation est Microsoft Windows. Si vous créez un nouveau modèle, cette zone est prédéfinie sur `CHANGE_ME.flt`. Vous devez la remplacer par une valeur significative avant de continuer.	*<htmlSnippet>*

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
HTML Snippet	Contenu spécifique que le Web Connector doit insérer dans le point d'interaction sur la page Web. Ce fragment peut contenir du code HTML, des informations de code de feuille de style en cascade (CSS) ou du code JavaScript à exécuter sur la page. Chacun de ces trois types de contenus doit être placé entre les codes BEGIN et END, comme dans les exemples suivants : • ${BEGIN_HTML} *<votre contenu HTML>* ${END_HTML} • ${BEGIN_CSS} *<vos informations sur la feuille de style spécifique à votre point d'interaction>* ${END_CSS} • ${BEGIN_JAVASCRIPT} *<votre code JavaScript spécifique à votre point d'interaction>* ${END_JAVASCRIPT} Vous pouvez également entrer un certain nombre de codes spéciaux prédéfinis qui sont remplacés automatiquement lorsque la page est chargée, notamment : • `${logAsAccept}` : Une macro qui prend deux paramètres (une URL cible, et le TreatmentCode utilisé pour identifier l'acceptation de l'offre) et les remplace par l'URL de clics. • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}` • `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` Chacun des codes d'offre listés représentent les attributs d'offre définis dans le modèle d'offre IBM Unica Campaign qui a été utilisé par le vendeur pour créer des offres renvoyées par Interact. Notez que le Web Connector utilise un moteur de modèle appelé FreeMarker qui offre de nombreuses options supplémentaires utiles pour configurer des codes sur vos modèles de page. Pour plus d'informations, voir http://freemarker.org/docs/index.html.	Aucun équivalent car le fragment HTML réside dans son propre fichier distinct de jsconnector.xml.

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
Exemples de codes spéciaux	Contient des exemples de type de codes spéciaux, notamment des codes identifiant des blocs comme HTML, CSS, ou JAVASCRIPT, et des zones déroulantes que vous pouvez insérer pour faire référence à des métadonnées d'offres spécifiques.	Aucun équivalent.

Les modifications apportées à cette page sont sauvegardées automatiquement lorsque vous accédez à une autre page de configuration de Web Connector.

Enhanced Pages - Configuration de WebConnector

Enhanced Pages permettent de mapper des paramètres spécifiques à une page avec un modèle d'URL. Par exemple, vous pouvez configurer une page de mappage, de façon à ce que toute adresse URL entrante contenant le texte "index.htm" affiche votre page d'accueil générale, avec des événements spécifique de chargement de page et les points d'interaction définis pour ce mappage.

Remarque : Les paramètres de configuration de cette page correspondent à la section `pageMapping` du fichier de configuration `jsconnector.xml`.

Pour utiliser la page Enhanced Pages pour créer un nouveau mappage de page, cliquez sur le lien **Add a Page** et renseignez les informations requises pour le mappage.

Page Info

Les options de configuration de Page info pour la page de mappage définissent le masque d'adresse URL qui sert de déclencheur pour ce mappage, ainsi que certains paramètres supplémentaires définissant la manière dont cette page de mappage est gérée par Interact.

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
URL contains	Il s'agit du masque d'URL que Web Connector doit observer dans la demande de page entrante. Par exemple, si l'URL contient la demande "hypotheque.htm", vous pouvez la faire correspondre avec votre page d'informations sur les prêts hypothécaires.	*<urlPattern>*
Friendly name for this page or set of pages	Nom significatif associé à votre propre référence et qui décrit le rôle de ce mappage de page, tels que "Page d'information sur les prêts hypothécaires".	*<friendlyName>*
Also return offers as JSON data for JavaScript use	Liste déroulante indiquant si vous souhaitez que Web Connector inclue les données d'offre brutes au format JavaScript Object Notation (http://www.json.org/) à la fin du contenu de la page.	*<enableRawDataReturn>*

Evénements à déclencher (charger) en cas de visite à cette page ou à ce jeu de pages

Ces ensembles d'options de configuration pour la page de mappage définissent le masque d'adresse URL qui sert de déclencheur à ce mappage, ainsi que certains paramètres supplémentaires définissant la manière dont cette page de mappage est gérée par Interact.

Remarque : Les paramètres de configuration de cette section correspondent à la section `<pageLoadEvents>` du fichier `jsconnector.xml`.

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
Evénements individuels	Liste des événements qui sont disponibles pour cette page ou ce jeu de pages. Les événements de cette liste sont ceux que vous avez définis dans Interact, Sélectionnez un ou plusieurs événements devant se produire lorsque la page est chargée. La séquence des appels d'API Interact est la suivante : 1. `startSession` 2. `postEvent` pour chaque événement de chargement de page individuel (si vous avez défini les événements individuels dans Interact) 3. Pour chaque point d'interaction : • `getOffers` • `postEvent(ContactEvent)`	*<event>*

Points d'interaction (emplacements d'affichage d'offre) sur cette page ou ce jeu de pages

Cet ensemble d'options de configuration pour le mappage de page vous permet de sélectionner des points d'interaction qui apparaissent sur la page Interact.

Remarque : Les paramètres de configuration de cette section correspondent à la section `<pageMapping>` | `<page>` | `<interactionPoints>` du `jsconnector.xml`.

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
Case à cocher du nom du Point d'interaction	Chaque point d'interaction qui a été défini dans le fichier de configuration s'affiche dans cette section de la page. Si vous cochez la case en regard du nom du point d'interaction, un certain nombre d'options disponibles pour ce point d'interaction s'affiche.	*<interactionPoint>*

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
HTML Element ID (Interact définit innerHTML)	Nom de l'élément HTML qui doit recevoir le contenu de ce point d'interaction. Par exemple, si vous avez indiqué `<div id="welcomebanner">` sur la page, vous devez entrer `welcomebanner` (la valeur de l'ID) dans cette zone.	*<htmlElementId>*
HTML Display Type	Liste déroulante qui vous permet de sélectionner le type d'affichage HTML (fragments HTML, ou fichiers .flt définis sur une page de configuration de Web Connector précédente) à utiliser pour ce point d'interaction.	*<htmlSnippet>*
Maximum number of offers to present (if this is a carousel or flipbook)	Nombre maximum d'offres que Web Connector doit extraire du serveur Interact pour ce point d'interaction. Cette zone est facultative et s'applique uniquement à un point d'interaction qui met à jour régulièrement les offres présentées sans recharger la page, comme dans le scénario de carrousel où plusieurs offres sont extraites afin de pouvoir être mises à disposition une par une.	*<maxNumberOfOffers>*
Event to fire when the offer is presented	Nom de l'événement de contact devant être publié pour ce point d'interaction.	*<contactEvent>*
Event to fire when the offer is accepted	Nom de l'événement d'acceptation devant être publié pour ce point d'interaction.	*<acceptEvent>*
Event to fire when the offer is rejected	Nom de l'événement de refus devant être publié pour ce point d'interaction. **Remarque :** A ce stade, cette fonction n'est pas encore utilisée.	*<rejectEvent>*

Options de configuration de Web Connector

En général, vous pouvez utiliser une interface graphique de Web Connector pour configurer vos paramètres Web Connector. Tous les paramètres que vous spécifiez sont également stockés dans un fichier appelé jsconnector.xml, qui se trouve dans votre répertoire `jsconnector/conf`. Vious trouverez ici une description de chacun des paramètres enregistrés dans le fichier de configuration jsconnector.xml.

Paramètres et leurs descriptions

Les paramètres suivants sont stockés dans le fichier `jsconnector.xml` et sont utilisés pour les interactions de Web Connector. Il existe deux solutions pour modifier ces paramètres :

- Utilisez la page Web de Configuration de Web Connector, qui est disponible automatiquement lorsque vous avez déployé et démarré l'application Web

Connector. Pour utiliser la page Web de Configuration, dans votre navigateur Web, ouvrez une URL du type suivant : `http://<host>:<port>/interact/jsp/WebConnector.jsp`.

Les modifications apportées à la page Web d'administration sont stockées dans le fichier `jsconnector.xml` sur le serveur sur lequel Web Connector est déployé.

- Editez le fichier `jsconnector.xml` directement à l'aide de n'importe quel éditeur de texte ou éditeur XML. Vous devez avoir l'habitude d'éditer les balises et les valeurs XML avant d'utiliser cette méthode.

  Remarque : Chaque fois que vous modifiez le fichier `jsconnector.xml` manuellement, vous pouvez recharger ces paramètres en ouvrant la page d'administration de Web Connector (à l'adresse `http://<host>:<port>/interact/jsp/jsconnector.jsp`) puis en cliquant sur **Reload Configuration**.

Le tableau suivant décrit les options de configuration que vous pouvez définir tels qu'elles apparaissent dans le fichier `jsconnector.xml`.

Tableau 30. Options de configuration de Web Connector

Groupe de paramètres	Paramètre	Description
`defaultPageBehavior`		
	`friendlyName`	Identificateur lisible par l'homme correspondant au modèle d'URL et devant s'afficher sur la page de configuration Web de Web Connector.
	`interactURL`	URL de base du serveur d'exécution Interact. Remarque: Vous devez définir ce paramètre uniquement si le Web Connector (jsconnector) du service est exécuté en tant qu'application Web déployée. Vous n'avez pas besoin de définir ce paramètre si WebConnector est exécuté automatiquement avec le serveur d'exécution Interact.
	`jsConnectorURL`	URL de base utilisée pour générer l'URL de clics, tel que `http://host:port/jsconnector/clickThru`
	`interactiveChannel`	Nom du canal interactif qui représente ce mappage de page.
	`sessionIdCookie`	Nom du cookie contenant l'ID de session qui est utilisé dans les appels d'API à Interact.
	`visitorIdCookie`	Nom du cookie contenant l'ID de référentiel.
	`audienceLevel`	Niveau de référentiel campagne destiné au visiteur entrant, utilisé dans l'appel d'API à l'environnement d'exécution Interact.
	`audienceIdField`	Nom de la zone `audienceId` utilisée dans l'appel d'API à l'environnement d'exécution Interact runtime. **Remarque :** Remarque: Les identificateurs de référentiel multi-zone ne sont actuellement pas pris en charge.
	`audienceIdFieldType`	Le type de données de la zone d'ID de référentiel [`numeric | string`] utilisé dans l'appel API à l'exécution Interact

Tableau 30. Options de configuration de Web Connector (suite)

Groupe de paramètres	Paramètre	Description
	audienceLevelCookie	Nom du cookie qui doit contenir le référentiel. Ce paramètre est facultatif. Si vous ne définissez pas ce paramètre, le système utilise celui défini pour audienceLevel.
	relyOnExistingSession	Utilisé dans l'appel API à l'exécution Interact. En règle générale, ce paramètre est défini sur "true".
	enableInteractAPIDebug	Utilisé dans l'appel API à l'exécution Interact pour activer la sortie de débogage dans les fichiers journal.
	pageLoadEvents	Evénement qui sera publié une fois cette page particulière chargée. Indiquez un ou plusieurs événements dans cette balise, dans un format de type <event>event1</event>.
	interactionPointValues	Tous les éléments de cette catégorie jouent le rôle de valeurs par défaut pour les valeurs manquantes dans les catégories IP spécifiques.
	interactionPointValuescontactEvent	Nom par défaut de l'événement de contact devant être publié pour ce point d'interaction particulier.
	interactionPointValuesacceptEvent	Nom par défaut de l'événement d'acceptation devant être publié pour ce point d'interaction particulier.
	interactionPointValuesrejectEvent	Nom par défaut de l'événement de refus devant être publié pour ce point d'interaction particulier. (Remarque: cette fonction n'est pas utilisée actuellement).
	interactionPointValueshtmlSnippet	Nom par défaut du modèle HTML à proposer pour ce point d'interaction.
	interactionPointValuesmaxNumberOfOffers	Nombre maximum d'offres par défaut à extraire de Interact pour ce point d'interaction.
	interactionPointValueshtmlElementId	Nom par défaut de l'élément HTML à proposer pour ce point d'interaction.
	interactionPoints	Cette catégorie contient la configuration de chaque point d'interaction. En cas de propriété absente, le système se base sur ce qui est configuré dans la catégorie interactionPointValues.
	interactionPointname	Nom du point d'interaction (IP).
	interactionPointcontactEvent	Nom de l'événement de contact devant être publié pour ce point d'interaction particulier.
	interactionPointacceptEvent	Nom de l'événement d'acceptation devant être publié pour ce point d'interaction particulier.

Tableau 30. Options de configuration de Web Connector (suite)

Groupe de paramètres	Paramètre	Description
	interactionPointrejectEvent	Nom de l'événement de refus devant être publié pour ce point d'interaction particulier. (Notez que cette fonction n'est pas encore utilisée.)
	interactionPointhtmlSnippet	Nom du modèle HTML à proposer pour ce point d'interaction.
	interactionPointmaxNumberOfOffers	Nombre maximum d'offres par défaut à extraire de Interact pour ce point d'interaction.
	interactionPointhtmlElementId	Nom de l'élément HTML devant recevoir le contenu de ce point d'interaction.
	enableDebugMode	Indicateur booléen (valeurs admises : true ou false) activant le mode de débogage spécial. Si vous le définissez ce true, le contenu renvoyé par le Web Connector inclut un appel JavaScript à 'alert' informant le client du mappage de page particulier qui vient de se produire. Le client doit disposer d'une entrée dans le fichier authorizedDebugClients pour générer l'alerte.
	authorizedDebugClients	Fichier utilisé par le mode de débogage spécial qui contient la liste des noms d'hôte ou des adresses du protocole Internet (IP) qui se qualifient pour le mode débogage.
	enableRawDataReturn	Indicateur booléen (valeurs admises: true ou false) qui détermine si le Web Connector joint l'offre des données brutes au format JSON à la fin du contenu.
	enableNetInsightTagging	Indicateur booléen (valeurs admises: true ou false) qui détermine si le Web Connector joint une balise NetInsight à la fin du contenu.
	apiSequence	Représente une implémentation de l'interface APISequence, qui dicte la séquence des appels d'API effectués par le Web Connector lorsqu'un pageTag est appelé. Par défaut, l'implémentation utilise une séquence de StartSession, pageLoadEvents, getOffers et logContact, où les deux derniers sont spécifiques à chaque point d'interaction.
	clickThruApiSequence	Représente une implémentation de l'interface APISequence, qui dicte la séquence des appels d'API effectués par le Web Connector lorsqu'un clickThru est appelé. Par défaut, l'implémentation utilise une séquence de StartSession et logAccept.
	netInsightTag	Représente le code HTML et le modèle JavaScript utilisés pour intégrer un appel à la balise NetInsight. En général, il n'est pas nécessaire de modifier cette option.

Utilisation de la page d'administration de Web Connector

Web Connector inclut une page d'administration qui fournit des outils de gestion et de test de la configuration, telle qu'elle peut être utilisée avec des modèles d'URL spécifiques. Vous pouvez également utiliser la page d'administration pour recharger une configuration que vous avez modifiée.

A propos de la page d'administration

Dans n'importe quel navigateur Web pris en charge, vous pouvez ouvrir `http://host:port/interact/jsp/jsconnector.jsp`, où *host:port* est le nom d'hôte sur lequel Web Connector s'exécute et le port sur lequel il est à l'écoute des connexions, par exemple `runtime.example.com:7001`

Vous pouvez utiliser la page d'administration de l'une des manières suivantes :

Tableau 31. Options de la page d'administration de Web Connector

Option	Objectif
Reload Configuration	Cliquez sur le lien **Reload Configuration** pour recharger les modifications de configuration enregistrés sur disque en mémoire. Cela est nécessaire lorsque vous avez apporté des modifications directement dans le fichier de configuration jsconnector.xml de Web Connector au lieu d'utiliser les pages Web de configuration.
View Config	Affichez la configuration de WebConnector en fonction du modèle d'URL que vous entrez dans la zone **View Config**. Lorsque vous entrez l'URL d'une page et cliquez sur **View Config**, Web Connector renvoie la configuration que le système utilisera en fonction de ce mappage de modèle. Si aucune correspondance n'est trouvée, la configuration par défaut est renvoyée. Ceci est utile pour tester si la configuration correcte est utilisée pour une page donnée.
Execute Page Tag	Lorsque vous complétez les zones de cette page et cliquez sur **Execute Page Tag**, Web Connector renvoie le résultat `pageTag` basé sur le modèle d'URL. Cela simule l'appel à une balise de page. La différence entre un appel au pageTag depuis cet outil et l'utilisation d'un site Web réel est que l'utilisation de cette page d'administration permet d'afficher les erreurs et les exceptions. Pour un site Web réel, les exceptions ne sont pas renvoyées et sont visibles uniquement dans le fichier journal de Web Connector.

Exemple de page Web Connector

A titre d'exemple, un fichier appelé testPage.html a été inclus avec le Web Connector Interact qui montre le nombre de fonctions de Web Connector qui serait balisées dans une page. Pour des raisons pratiques, cet exemple de page est également affiché ici.

Exemple de page HTML de Web Connector

```
<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=us-ascii" />
   <meta http-equiv="CACHE-CONTROL" content="NO-CACHE" />
<script language="javascript" type="text/javascript">
//<![CDATA[
/* ################################################################
```

```
Ceci est une page de test qui contient la balise WebConnector pageTag.
Etant donné que TestPage est intégré au nom de ce fichier,
le WebConnector détecte un modèle d'URL qui correspond avec le
masque d'URL "testpage" dans la version par défaut de
jsconnector.xml - la définition de configuration mappée avec ce
masque d'URL "testpage" s'applique ici. Cette page doit donc comporter les
ID d'élément HTML correspondants associés aux IP de ce masque
d'URL (par exemple : welcomebanner', 'crosssellcarousel' et 'textservicemessage')
######################################################################## */

/* ######################################################################
Cette section définit les cookies de sessionId et de visitorId.
Dans un site Web de production réel, cette opération est effectuée
en général par le composant de connexion. A des fins de test, elle est
effectuée ici. Le nom du cookie doit correspondre à ce qui est
configuré dans jsconnector xml.
######################################################################## */
     function setCookie(c_name,value,expiredays)
     {
       var exdate=new Date();
       exdate.setDate(exdate.getDate()+expiredays);
       document.cookie=c_name+ "=" +escape(value)+
       ((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
     }
     setCookie("SessionID","123");
     setCookie("CustomerID","1");

   /* ####################################################
   Définissez les ID d'élément HTML correspondant aux IP
   #################################################### */
     document.writeln("<div id='welcomebanner'> This should change, "
  + "otherwise something is wrong <\/div>");
     document.writeln("<div id='crosssellcarousel'> This should change, "
  + "otherwise something is wrong <\/div>");
     document.writeln("<div id='textservicemessage'> This should change, "
  + "otherwise something is wrong <\/div>");
   //]]&gt;
</script><!--
#############################################################################
Voici ce qui est collé depuis le fichier pageTag.txt dans le répertoire
conf de l'installation WebConnector... la variable unicaWebConnectorBaseURL
doit être adaptée pour se conformer à votre environnement
local WebConnector
#############################################################################
-->
<!-- BEGIN: Unica Interact Web Connector Page Tag -->
<!-- Copyright 2011, IBM Corporation All rights reserved. -->
<script language="javascript" type="text/javascript">
//<![CDATA[
   var unicaWebConnectorBaseURL = "http://localhost:7001/interact/pageTag";
   var unicaURLData = "ok=Y";
   try {
    unicaURLData += "&url=" + escape(location.href)
   } catch (err) {}
   try {
    unicaURLData += "&title=" + escape(document.title)
   } catch (err) {}
   try {
    unicaURLData += "&referrer=" + escape(document.referrer)
   } catch (err) {}
   try {
    unicaURLData += "&cookie=" + escape(document.cookie)
   } catch (err) {}
   try {
    unicaURLData += "&browser=" + escape(navigator.userAgent)
   } catch (err) {}
   try {
```

```
    unicaURLData += "&screensize=" +
    escape(screen.width + "x" + screen.height)
   } catch (err) {}
   try {
    if (affiliateSitesForUnicaTag) {
        var unica_asv = "";
        document.write("<style id=\"unica_asht1\" type=\"text/css\"> "
     + "p#unica_ashtp a {border:1px #000000 solid; height:100px "
     + "!important;width:100px "
     + "!important; display:block !important; overflow:hidden "
     + "!important;} p#unica_ashtp a:visited {height:999px !important;"
     + "width:999px !important;} <\/style>");
        var unica_ase = document.getElementById("unica_asht1");
        for (var unica_as in affiliateSitesForUnicaTag) {
         var unica_asArr = affiliateSitesForUnicaTag[unica_as];
         var unica_ashbv = false;
         for (var unica_asIndex = 0; unica_asIndex <
     unica_asArr.length && unica_ashbv == false;
     unica_asIndex++)
   {
          var unica_asURL = unica_asArr[unica_asIndex];
          document.write("<p id=\"unica_ashtp\" style=\"position:absolute; "
     + "top:0;left:-10000px;height:20px;width:20px;overflow:hidden; \
   margin:0;padding:0;visibility:visible;\"> \
   <a href=\"" + unica_asURL + "\">" + unica_as + " <\/a><\/p>");
          var unica_ae = document.getElementById("unica_ashtp").childNodes[0];
          if (unica_ae.currentStyle) {
           if (parseFloat(unica_ae.currentStyle["width"]) > 900)
       unica_ashbv = true
          } else if (window.getComputedStyle) {
           if (parseFloat(document.defaultView.getComputedStyle
       (unica_ae, null).getPropertyValue("width")) > 900)
    unica_ashbv = true
          }
          unica_ae.parentNode.parentNode.removeChild(unica_ae.parentNode)
         }
         if (unica_ashbv == true) {
          unica_asv += (unica_asv == "" ? "" : ";") + unica_as
         }
        }
        unica_ase.parentNode.removeChild(unica_ase);
        unicaURLData += "&affiliates=" + escape(unica_asv)
    }
   } catch (err) {}
   document.write("<script language='javascript' "
     + " type='text/javascript' src='" + unicaWebConnectorBaseURL + "?"
  + unicaURLData + "'><\/script>");
  //]]&gt;
</script>
<style type="text/css">
/*<![CDATA[*/
   .unicainteractoffer {display:none !important;}
   /*]]&gt;*/
</style>
   <title>Sample Interact Web Connector Page</title>
   </head>
   <body>
   <!-- END: Unica Interact Web Connector Page Tag -->
   <!--
################################################################################
end of pageTag paste
################################################################################
-->
   </body>
</html>
```

Annexe E. Interact et Intelligent Offer - Recommandations concernant le produit intégré

IBM Unica Interact peut s'intégrer à IBM Coremetrics Intelligent Offer et fournir des recommandations de produit déterminées par Interact. Mes deux produits peuvent fournir des recommandations de produit mais avec des méthodes différentes. Intelligent Offer utilise le comportement Web d'un visiteur (filtre collaboratif) pour créer des corrélations entre les visiteurs et les offres recommandées. Interact se base sur le comportement antérieur du client, ainsi que sur ses attributs, son historique et ses offres les moins vues. Il permet d'apprendre quelles offres correspondent le mieux au profil de comportement du client en fonction des données démographiques et d'autres informations le concernant. Les taux d'acceptation des offres contribuent à créer un modèle prédictif grâce à l'auto-apprentissage. En utilisant les meilleures qualités de ces produits, Interact a recours à un profil personnel pour définir les offres qui passeront un ID de catégorie à Intelligent Offer et pour récupérer les produits recommandés en fonction de la popularité (la "sagesse des collectivités") afin de les proposer au visiteur dans le cadre des offres sélectionnées. Cela permet de fournir de meilleures recommandations aux clients, avec à la clé un plus grand nombre de clics et de meilleurs pour résultats que si chaque produit agissait seul.

Les sections suivantes expliquent comme cette intégration fonctionne, et montrent comment utiliser l'exemple d'application fourni pour créer votre propre intégration d'offre personnalisée.

Présentation de l'intégration Interact avec Intelligent Offer

Cette section explique comment IBM Unica Interact peut s'intégrer à IBM Coremetrics Intelligent Offer pour fournir des recommandations de produits régies par Interact, notamment une description du processus, et les mécanismes par lesquels l'intégration a lieu.

IBM Unica Interact s'intègre à IBM Coremetrics Intelligent Offer via une interface de programme d'application (API) REST (Representational state transfer), disponible dans l'installation Intelligent Offer. En effectuant les appels REST API avec l'ID de catégorie approprié, Interact peut extraire les produits recommandés et les inclure dans les informations de l'offre affichées sur la page personnalisée visualisée par le visiteur.

Lorsqu'un visiteur affiche l'URL de la page Web (par exemple la page JSP d'exemple fournie avec votre installation Interact), la page appelle Interact pour appeler une offre. Si l'on suppose que l'offre a été configurée dans Interact avec les paramètres corrects, le processus suivant se déroule (dans le scénario le plus simple) :

1. La logique de la page identifie l'ID client du visiteur.
2. Un appel d'API à Interact est effectué, et transmet les informations requises pour générer une offre pour ce c client.
3. L'offre renvoyée fournit au moins trois attributs à la page Web : l'URL de l'image de l'offre, l'URL de la page d'accueil à laquelle les clics amènent le client, et l'ID de catégorie à utiliser pour déterminer les produits à recommander.

© Copyright IBM Corp. 2001, 2012

4. L'ID de catégorie est alors utilisé pour appeler Intelligent Offer pour extraire les produits recommandés. Cet ensemble de produits est au format JSON (JavaScript Object Notation) classé par les produits se vendant le mieux dans cette catégorie.
5. L'offre et les produits s'affichent alors dans le navigateur du visiteur.

Cette intégration est utile pour combiner une recommandation d'offre et des recommandations de produit. Par exemple, sur une page Web, vous pouvez avoir deux points d'interaction : un pour une offre et un pour les recommandations correspondant à cette offre. Pour ce faire, la page Web lance un appel à Interact pour réaliser une segmentation en temps réel afin de déterminer la meilleure offre (par exemple une remise de 10% sur tout le petit électroménager). Lorsque la page reçoit l'offre de Interact, cette offre contient alors l'ID de catégorie (dans cet exemple, le petit électroménager). La page passe alors l'ID de catégorie des petits appareils d'électroménager) à Intelligent Offer à l'aide d'un appel API et reçoit en réponse les meilleures recommandations de produits correspondant à cette catégorie en fonction de la popularité.

Un exemple plus simple est le cas où une page Web lance un appel à Interact mais ne trouve qu'une seule catégorie (par exemple des couverts haut de gamme) correspondant au profil du client. Elle passe dans ce cas l'ID de catégorie reçu à Intelligent Offer, et reçoit des recommandations sur les types de couverts recommandés.

Prérequis d'intégration

Avant de pouvoir utiliser l'intégration Intelligent Offer - Interact, vous devez vérifier que vous respectez les prérequis décrits dans cette section.

Vérifiez que les prérequis suivants sont respectés :

- Vous savez utiliser l'API Interact, documentée dans le *Guide d'administration* et dans l'aide en ligne.
- Vous savez utiliser l'API REST Intelligent Offer décrite dans la documentation de Intelligent Offer pour les développeurs.
- Vous avez des connaissances de base en HTML, JavaScript, CSS et JSON (JavaScript Object Notation).

  JSON est important car l'API REST Intelligent Offer renvoie les informations produit demandées sous forme de données au format JSON.
- Vous connaissez le codage côté serveur des pages Web, car l'application de démonstration fournie avec Interact utilise JSP (même si JSP n'est pas obligatoire).
- Vous avez un compte Intelligent Offer valide et la liste des ID de catégorie que Interact doit utiliser pour extraire les recommandations de produits (les produits les mieux vendus ou les plus populaires dans la catégorie que vous indiquez).
- Vous avez le lien de l'API REST Intelligent Offer (une adresse URL de votre environnement Intelligent Offer).

  Consultez le modèle d'application fourni avec votre installation Interact à titre d'exemple, ou consultez l'exemple de code dans «Utilisation de l'exemple de projet d'intégration», à la page 256 pour plus d'informations.

Configuration d'une offre pour l'intégration Intelligent Offer

Pour que votre page Web puisse appeler IBM Coremetrics Intelligent Offer afin d'extraire un produit recommandé, vous devez d'abord configurer l'offre IBM Unica Interact avec les informations nécessaires à transmettre à Intelligent Offer.

Pour configurer une offre à lier à Intelligent Offer, vérifiez que les conditions suivantes sont respectées :

- Vérifiez que votre serveur d'exécution Interact est configuré et fonctionne correctement.
- Vérifiez que le serveur d'exécution peut établir une connexion avec le serveur Intelligent Offer, et assurez-vous notamment que votre pare-feu n'empêche pas l'établissement d'une connexion Web sortante standard (port 80).

Pour configurer une offre pour l'intégration à Intelligent Offer, procédez comme suit.

1. Créez ou éditez une offre pour Interact.

   Pour savoir comment ou modifier des offres, voir le *Guide d'utilisation de IBM Unica Interact*, et la documentation de IBM Unica Campaign.
2. Outre les autres paramètres de l'offre, vérifiez que l'offre comprend les attributs d'offre suivants :
   - L'adresse URL de lien à l'image de l'offre.
   - L'adresse URL de lien à la page d'accueil de l'offre.
   - Un ID de catégorie Intelligent Offer associé à cette offre.

     Vous pouvez récupérer manuellement l'ID de catégorie depuis votre configuration Intelligent Offer. Interact ne peut pas récupérer directement les valeurs d'ID de catégorie.

   Dans l'application Web de démonstration fournie avec votre installation Interact, ces attributs d'offre sont appelés `ImageURL`, `ClickThruURL` et `CategoryID`. Vous pouvez utiliser des noms significatifs pour vous, à condition que votre application Web établisse la correspondance avec les valeurs attendues par l'offre.

   Par exemple, vous pouvez définir une offre appelée "10PercentOff" contenant ces attributs, où l'ID de catégorie (extrait de votre configuration Intelligent Offer) est `PROD1161127`, l'adresse URL des clics de l'offre est `http://www.example.com/success`, et l'adresse URL de l'image à afficher pour l'offre est `http://localhost:7001/sampleIO/img/10PercentOffer.jpg` (il s'agit dans ce cas d'une adresse URL locale sur le serveur d'exécution Interact).
3. Définissez les règles de traitement d'un canal interactif afin qu'il inclue cette offre, et déployez le canal interactif comme d'habitude.

L'offre est maintenant définie avec les informations requises pour l'intégration Intelligent Offer. Les tâches restantes qui permettront à Intelligent Offer de fournir des recommandations de produits à Interact consistent à configurer vos pages Web afin qu'elles lancent les appels d'API appropriés.

Lorsque vous configurez votre application Web afin qu'elle serve la page intégrée aux visiteurs, vérifiez que les fichiers suivants sont inclus dans le répertoire `WEB-INF/lib` :

- *`Interact_Home`*`/lib/interact_client.jar`, qui est obligatoire pour gérer les appels de votre page Web à l'API Interact.

- *Interact_Home*/lib/JSON4J_Apache.jar, qui est obligatoire pour gérer les données renvoyées par l'appel à l'API REST Intelligent Offer, qui renvoie des données au format JSON.

Voir «Utilisation de l'exemple de projet d'intégration» pour savoir comment servir les offres à vos clients.

Utilisation de l'exemple de projet d'intégration

Chaque installation de l'exécution Interact est fournie avec un exemple de projet qui explique le processus d'intégration de Intelligent Offer - Interact. L'exemple de projet fournit une démonstration complète et de bout en bout. Il explique comment créer une page Web qui appelle une offre contenant un ID de catégorie, qui est ensuite transmis à Intelligent Offer en vue de récupérer une liste de produits recommandés à présenter dans les points d'interaction de la page.

Présentation

Vous pouvez utiliser l'exemple de projet tel quel, si vous souhaitez tester le processus d'intégration, ou l'utiliser comme point de départ pour développer vos propres pages personnalisées. L'exemple de projet se trouve dans le fichier suivant :

Interact_home/samples/IntelligentOfferIntegration/MySampleStore.jsp

Ce fichier contient un exemple fonctionnel complet du processus d'intégration. Il inclut également des commentaires détaillés expliquant ce que vous devez configurer dans Interact, ce que vous devez personnaliser dans le fichier .jsp, et comment déployer correctement la page à exécuter avec votre installation.

MySampleStore.jsp

Pour plus de commodité, le fichier MySampleStore.jsp est représenté ici. Cet exemple peut être mis à jour dans les éditions suivantes de Interact. Utilisez par conséquent le fichier fourni avec votre installation comme point de départ pour les exemples dont vous avez besoin.

```
<!--
# ********************************************************************************
#  Licensed Materials - Property of IBM
#  Unica Interact
#  (c) Copyright IBM Corporation 2001, 2011.
#  US Government Users Restricted Rights - Use, duplication or disclosure
#  restricted by GSA ADP Schedule Contract with IBM Corp.
# ********************************************************************************

-->


<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page import="java.net.URL,
    java.net.URLConnection,
    java.io.InputStreamReader,
    java.io.BufferedReader,
    com.unicacorp.interact.api.*,
    com.unicacorp.interact.api.jsoverhttp.*,
    org.apache.commons.json.JSONObject,
    org.apache.commons.json.JSONArray" %>

<%

 /*******************************************************************************
 * Cet exemple de programme jsp explique l'intégration d'Interact et d'IntelligentOffer.
 *
 * Lorsque vous accédez à l'URL de ce jsp via un navigateur, la logique appelle Interact
```

```
* pour recherche une Offre.  En fonction de l'ID de catégorie associé à l'offre, la logique
* appelle IntelligentOffer pour rechercher des produits recommandés. L'offre et les produits
* s'affichent.
* Pour basculer l'ID client pour voir différentes offres, vous pouvez simplement
* ajouter cid=<id> à l'URL de ce JSP.
*
* Prérequis pour comprendre cette démo :
* 1) Connaissance d'Interact et de son API java
* 2) Connaissance d'IntelligentOffer et de son API REST
* 3) Connaissances Web de base (html, css, javascript) pour le marquage de page
* 4) Technologie utilisée pour générer une page Web (pour cette démo, nous utilisons JSP exécuté côté serveur)
*
*
* Marche à suivre pour faire fonctionner cette démo :
* 1) Configurez un environnement d'exécution Interact pouvant proposer des offres
* ayant les attributs suivants :
*  ImageURL : URL de lien à l'image de l'offre
*  ClickThruURL : URL de lien à la page d'accueil de l'offre
*  CategoryID : Catégorie IntelligentOffer associée à l'offre
* REMARQUE : d'autres noms peuvent être utilisés pour les attributs, à condition que les références à ces
* attributs dans ce jsp soient modifiées en conséquence.
* 2) Obtenez une URL d'API REST valide pour l'environnement Intelligent Offer
* 3) Intégrez ce JSP dans une application Web Java
* 4) Vérifiez que interact_client.jar se trouve dans le répertoire WEB-INF/lib (communication avec Interact)
* 5) Vérifiez que JSON4J_Apache.jar (provenant de l'installation d'interact) se trouve dans le
 *    répertoire WEB-INF/lib (communication avec IO)
* 6) Définissez les propriétés de l'environnement dans les deux sections suivantes
*****************************************************************************/

/*****************************************************************************
*   *****************MODIFIEZ CES PARAMETRES EN FONCTION DE VOTRE ENVIRONNEMENT*******************
* Définissez ici les propriétés de votre environnement Interact...
*****************************************************************************/

final String sessionId="123";
final String interactiveChannel = "SampleIO";
final String audienceLevel = "Customer";
final String audienceColumnName="CustomerID";
final String ip="ip1";
int customerId=1;
final String interactURL="http://localhost:7011/interact/servlet/InteractJSService";
final boolean debug=true;
final boolean relyOnExistingSession=true;

/*****************************************************************************
  *****************MODIFIEZ CES PARAMETRES EN FONCTION DE VOTRE ENVIRONNEMENT*********************
* Définissez ici les propriétés de votre environnement Intelligent Offers...
*****************************************************************************/

final String ioURL="http://recs.coremetrics.com/iorequest/restapi";
final String zoneID="ProdRZ1";
final String cID="90007517";



/*****************************************************************************
  *****************************************************************************/

StringBuilder interactErrorMsg = new StringBuilder();
StringBuilder intelligentOfferErrorMsg = new StringBuilder();

// get the customerID if passed in as a parameter
String cid = request.getParameter("cid");
if(cid != null)
{
 customerId = Integer.parseInt(cid);
}

// call Interact to get offer
Offer offer=getInteractOffer(interactURL,sessionId,interactiveChannel,audienceLevel,
       audienceColumnName,ip,customerId,debug,relyOnExistingSession,interactErrorMsg);

// get specific attributes from the offer (img url, clickthru url, & category id)
String offerImgURL=null;
String offerClickThru=null;
String categoryId="";

if(offer != null)
{
```

```
  for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
     {
         if(offerAttribute.getName().equalsIgnoreCase("ImageURL"))
         {
          offerImgURL=offerAttribute.getValueAsString();
         }
         else if(offerAttribute.getName().equalsIgnoreCase("ClickThruURL"))
         {
          offerClickThru=offerAttribute.getValueAsString();
         }
         else if(offerAttribute.getName().equalsIgnoreCase("CategoryID"))
         {
          categoryId=offerAttribute.getValueAsString();
         }
     }
 }

 // call IO to get products
 JSONObject products=getProductsFromIntelligentOffer(ioURL, cID, zoneID, categoryId,
        intelligentOfferErrorMsg);

%>

<html>
 <head>
  <title>Ma boutique préférée</title>

  <script language="javascript" type="text/javascript">
   var unicacarousel=(function(){var g=false;var h;var j=0;var k=0;var l=0;var m=40;
          var n=new Array(0,2,6,20,40,60,80,88,94,97,99,100);var o=function(a){var b=a.parentNode;
          h=b.getElementsByTagName("UL")[0];var c=h.getElementsByTagName("LI");j=c[0].offsetWidth;
          k=c.length;l=Math.round((b.offsetWidth/j));unicacarousel.recenter()};var p=function(a)
          {var b=parseFloat(h.style.left);if(isNaN(b))b=0;for(var i=0;i<n.length;i++)
          {setTimeout("unicacarousel.updateposition("+(b+(a*(n[i]/100)))+");",((i*m)+50))}
          setTimeout("unicacarousel.recenter();",((i*m)+50))};return{gotonext:function(a,b)
          {if(!g){o(a);g=true;p((-1*b*j))}},gotoprev:function(a,b){if(!g){o(a);g=true;p((b*j))}},
          updateposition:function(a){h.style.left=a+"px"},recenter:function(){var a=parseFloat(h.style.left);
          if(isNaN(a))a=0;var b=j*Math.round(((l-k)/2));var c=Math.abs(Math.round((b-a)/j));
          if(a<b){var d=h.getElementsByTagName("LI");var e=new Array();
          for(var i=0;i<c;i++){e[e.length]=d[i]}for(var i=0;i<e.length;i++)
          {h.insertBefore(e[i],null)}unicacarousel.updateposition(b)}else
          if(a>b){var d=h.getElementsByTagName("LI");var e=new Array();
          for(var i=0;i<c;i++){e[e.length]=d[d.length-c+i]}var f=d[0];
          for(var i=0;i<e.length;i++){h.insertBefore(e[i],f)}unicacarousel.updateposition(b)}g=false}})();
  </script>

  <style type="text/css">
 .unicaofferblock_container {width:250px; position:relative; display:block;
                             text-decoration:none; color:#000000; cursor: pointer;}
 .unicaofferblock_container .unicateaserimage {margin:0px 0.5em 0.25em 0px; float:left;}
 .unicaofferblock_container .unicabackgroundimage {position:absolute; top:0px; left:0px;}
 .unicaofferblock_container .unicabackgroundimagecontent {width:360px; height:108px;
                     padding:58px 4px 4px 20px; position:relative; top:0px;}
 .unicaofferblock_container h4 {margin:0px; padding:0px; font-size:14px;}

 .unicacarousel {width:588px; position:relative; top:0px;}
 .unicacarousel_sizer {width:522px; height:349px; margin:0px 33px; padding:0;
                       overflow:hidden; position:relative;}
 .unicacarousel_rotater {height:348px; width:1000px; margin:0 !important;
                         padding:0; list-style:none; position:absolute; top:0px;
                         left:0px;}
 .unicacarousel li {width:167px; height:349px; float:left; padding:0 4px;
                    margin:0px !important; list-style:none !important;
                    text-indent:0px !important;}
 .unicacarousel_gotoprev, .unicacarousel_gotonext {width:18px; height:61px;
                     top:43px; background:url(../img/carouselarrows.png) no-repeat;
                    position:absolute; z-index:2; text-align:center; cursor:pointer;
                    display:block; overflow:hidden; text-indent:-9999px;
                    font-size:0px; margin:0px !important;}
 .unicacarousel_gotoprev {background-position:0px 0; left:0;}
 .unicacarousel_gotonext {background-position:-18px 0; right:0;}

  </style>


 </head>

 <body>
```

```
  <b>Bienvenue dans ma boutique</b> M./Mme <%=customerId %>
  <br><br>
<% if(offer != null) { %>
 <!-- Interact Offer HTML -->

 <div onclick="location.href='<%=offerClickThru %>'" class="unicaofferblock_container">
  <div class="unicabackgroundimage">
   <a href="<%=offerClickThru %>"><img src="<%=offerImgURL %>" height="170"
         width="695" border="0"></a>
  </div>
 </div>


<% } else { %>
 No offer available.. <br> <br>
 <%=interactErrorMsg.toString() %>
<% } %>

<% if(products != null) { %>
 <!-- IntelligentOffer Products HTML -->
 <br><br><br> <br><br><br> <br><br><br> <br><br><br> <br>
 <div class="unicacarousel">
 <div class="unicacarousel_sizer">
  <ul class="unicacarousel_rotater">


 <% JSONArray recs = products.getJSONObject("io").getJSONArray("recs");
  if(recs != null)
  {
   for(int x=0;x< recs.length();x++)
   {
    JSONObject rec = recs.getJSONObject(x);
    if(rec.getString("Page produit") != null &&
              rec.getString("Page produit").trim().length()>0) {
    %>

    <li>
     <a href="<%=rec.getString("Page produit") %>" title="<%=rec.getString("Nom produit") %>">
             <img src="<%=rec.getString("Image") %>"  width="166" height="148" border="0" />
     <%=rec.getString("Nom produit") %>
     </a>
    </li>

    <% }
   }
  }
 %>
 </ul>
 </div>
 <p class="unicacarousel_gotoprev" onclick="unicacarousel.gotoprev(this,1);"></p>
 <p class="unicacarousel_gotonext" onclick="unicacarousel.gotonext(this,1);"></p>
</div>
<% } else { %>
 <div>
 <br><br> <br><br><br> <br><br><br> <br><br><br> <br>
 No products available...<br> <br>
 <%=intelligentOfferErrorMsg.toString() %>
 </div>
<% } %>

 </body>
</html>



<%!
 /*****************************************************************************
 * Les fonctions suivantes sont des fonctions libre-service qui font des extractions dans Interact et
 *   Intelligent Offer
 *****************************************************************************/

 /*****************************************************************************
 * Appeler IntelligentOffer pour extraire les produits recommandés
 *****************************************************************************/
 private JSONObject getProductsFromIntelligentOffer(String ioURL, String cID,
     String zoneID, String categoryID, StringBuilder intelligentOfferErrorMsg)
 {

  try
```

```
 {

  ioURL += "?cm_cid="+cID+"&cm_zoneid="+zoneID+"&cm_targetid="+categoryID;
  System.out.println("CoreMetrics URL:"+ioURL);
  URL url = new java.net.URL(ioURL);

   URLConnection conn = url.openConnection();

   InputStreamReader inReader = new InputStreamReader(conn.getInputStream());
   BufferedReader in = new BufferedReader(inReader);

   StringBuilder response = new StringBuilder();

   while(in.ready())
   {
    response.append(in.readLine());
   }

   in.close();

   intelligentOfferErrorMsg.append(response.toString());

   System.out.println("CoreMetrics:"+response.toString());

   if(response.length()==0)
    return null;

   return new JSONObject(response.toString());
 }
 catch(Exception e)
 {
  intelligentOfferErrorMsg.append(e.getMessage());
  e.printStackTrace();
 }

 return null;

}

/*****************************************************************************
* Appeler Interact pour extraire l'offre
*****************************************************************************/
private Offer getInteractOffer(String interactURL,String sessionId,String interactiveChannel,
    String audienceLevel,
       String audienceColumnName,String ip, int customerId,boolean debug,
                    boolean relyOnExistingSession, StringBuilder interactErrorMsg)
{
 try
 {
  InteractAPI api = InteractAPI.getInstance(interactURL);
  NameValuePairImpl custId = new NameValuePairImpl();
     custId.setName(audienceColumnName);
      custId.setValueAsNumeric(Double.valueOf(customerId));
    custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
      NameValuePairImpl[] audienceId = { custId };

      // call startSession
     Response response = api.startSession(sessionId, relyOnExistingSession,
         debug, interactiveChannel, audienceId, audienceLevel, null);

     if(response.getStatusCode() == Response.STATUS_ERROR)
     {
      printDetailMessageOfWarningOrError("startSession",response, interactErrorMsg);
  }

     // call getOffers
     response = api.getOffers(sessionId, ip, 1);
     if(response == null  || response.getStatusCode() == Response.STATUS_ERROR)
     {
      printDetailMessageOfWarningOrError("getOffers",response, interactErrorMsg);
  }

           OfferList offerList=response.getOfferList();

           if(offerList != null && offerList.getRecommendedOffers() != null)
           {
            return offerList.getRecommendedOffers()[0];
           }
 }
```

```
  catch(Exception e)
  {
   interactErrorMsg.append(e.getMessage());
   e.printStackTrace();
  }
  return null;
 }

    private void printDetailMessageOfWarningOrError(String command, Response response,
        StringBuilder interactErrorMsg)
    {
     StringBuilder sb = new StringBuilder();
        sb.append("Calling "+command).append("<br>");
        AdvisoryMessage[] messages = response.getAdvisoryMessages();

        for(AdvisoryMessage msg : messages)
        {
         sb.append(msg.getMessage()).append(":");
         sb.append(msg.getDetailMessage());
         sb.append("<br>");
        }
        interactErrorMsg.append(sb.toString());
    }
%>
```

Coordonnées du support technique d'IBM Unica

Si vous rencontrez un problème que vous ne parvenez pas à résoudre à l'aide de la documentation, le responsable désigné dans votre société peut contacter le support technique d'IBM Unica. Utilisez les informations de cette section pour garantir la résolution efficace de votre problème.

Si vous ne faites pas partie des personnes autorisées dans votre société à appeler le support technique, contactez votre administrateur IBM Unica qui vous donnera toutes les informations nécessaires.

Informations à rassembler

Avant de contacter le support technique d'IBM Unica, rassemblez les informations suivantes :

- une brève description de la nature du problème,
- les messages d'erreur détaillés qui apparaissent lorsque l'erreur se produit,
- la liste détaillée des étapes permettant de reproduire l'erreur,
- les fichiers journaux, les fichiers de la session, les fichiers de configuration et les fichiers de données appropriés,
- les informations sur l'environnement de votre système et de votre produit, que vous pouvez obtenir en procédant comme indiqué dans la section Informations sur le système.

Informations système

Lorsque vous appelez le support technique d'IBM Unica, vous pouvez être invité à fournir des informations sur votre environnement.

Si vous pouvez vous connecter à votre application, la plupart de ces informations sont disponibles dans la page À propos, qui affiche des informations sur l'application IBM Unica installée.

Vous pouvez accéder à la page A propos de en sélectionnant **Aide > A propos de**. Si la page A propos de est inaccessible, il est possible d'obtenir le numéro de version de chaque application IBM Unica en consultant le fichier `version.txt` situé dans le répertoire d'installation de chaque application.

Coordonnées du support technique d'IBM Unica

Pour savoir comment contacter le support technique d'IBM Unica, consultez le site Web du support technique des produits IBM Unica : (http://www.unica.com/about/product-technical-support.htm).

© Copyright IBM Corp. 2001, 2012

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Consultez votre interlocuteur IBM local pour obtenir des informations sur les produits et les services actuellement disponibles dans votre pays. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7 Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales. LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

© Copyright IBM Corp. 2001, 2012

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
170 Tracer Lane
Waltham, MA 02451
U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Tous les tarifs indiqués sont les prix de vente actuels suggérés par IBM et sont susceptibles d'être modifiés sans préavis. Les tarifs appliqués peuvent varier selon les revendeurs.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les exemples de programmes sont fourni "en l'état", sans garantie d'aucune sorte. IBM ne sera en aucun cas responsable des dommages liés à l'utilisation de ces exemples de programmes.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp. aux Etats-Unis et/ou dans certains autres pays. D'autres noms de services et de produits peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques IBM est disponible sur la page Web «Copyright and trademark information» à l'adresse www.ibm.com/legal/copytrade.shtml.