

**Unica Interact V12.1.1  
Guide de l'administrateur**



# Contents

<b>Chapter 1. Administration d'Unica Interact.....</b>	<b>1</b>
Unica Interact : concepts clés.....	1
Niveaux d'audience.....	1
Environnement de phase de conception.....	2
Modèles d'événements et d'événements.....	3
Canaux interactifs.....	9
Diagrammes temps réel.....	9
Points d'interaction.....	10
Offres.....	10
Profils.....	11
Environnement d'exécution.....	12
Sessions d'exécution.....	12
Points de contact.....	12
Stratégie et règles de traitement.....	13
FlexOffers.....	13
Passerelles.....	15
Architecture Unica Interact.....	24
Considérations réseau Unica Interact.....	26
Ports de serveur et sécurité des réseaux Unica Interact.....	27
Connexion à Unica.....	30
<b>Chapter 2. Gestion de la sécurité.....</b>	<b>32</b>
Authentifier les pages Unica Interact JSP.....	32
<b>Chapter 3. Configuration des utilisateurs.....</b>	<b>33</b>

Configuration de l'utilisateur de l'environnement d'exécution.....	33
Configuration des utilisateurs de l'environnement de conception.....	35
Exemple de permissions dans un environnement de conception.....	38
<b>Chapter 4. Gestion des sources de données Unica Interact.....</b>	<b>42</b>
Sources de données Unica Interact.....	42
Bases de données et applications.....	43
Tables système Unica Campaign.....	45
Tables d'exécution.....	46
Tables d'exécution de test.....	48
Substitution des types de données par défaut utilisés pour les tables créées dynamiquement.....	49
Substitution des types de données par défaut.....	50
Types de données par défaut pour les tables créées dynamiquement.....	51
Base de données de profil.....	51
Tables d'apprentissage.....	54
Historique des contacts pour le suivi de réponse intersession.....	55
Exécution des scripts de base de données pour activer les fonctions d'Unica Interact.....	55
About Cross Session Contact Tracking.....	57
A propos du suivi de l'historique des contacts et des réponses.....	60
Types de contact et de réponse.....	61
Autres types de contacts.....	62
Autres types de réponse.....	63
Mappage des tables de transfert de l'environnement d'exécution avec les tables d'historique Unica Campaign.....	67

Configuration de la surveillance JMX pour le module de l'historique des contacts et des réponses.....	77
A propos du suivi de réponse intersession.....	78
Activer la détection et la suppression des doublons.....	79
Cross-session response process.....	80
Configuration de la source de données du suivi de réponse intersession.....	82
Configuration des tables de l'historique des contacts et des réponses pour le suivi de réponse intersession.....	83
Activation du suivi de réponse intersession.....	88
Mise en correspondance d'offre de réponse intersession.....	89
Utilisation d'un utilitaire de chargement de base de données avec l'environnement d'exécution.....	93
Activation d'un utilitaire de chargement de base de données avec l'environnement d'exécution.....	95
Processus ETL de modèle d'événement.....	95
Exécution du processus ETL autonome.....	96
Arrêt du processus ETL autonome.....	98
<b>Chapter 5. Présentation des offres.....</b>	<b>100</b>
Éligibilité d'une offre.....	100
Génération d'une liste d'offres candidates.....	100
Calcul du score marketing.....	102
Influencer l'apprentissage.....	103
Supprimer des offres.....	104
Activation de la suppression des offres.....	104
Table de suppression des offres.....	105
Ignorer la suppression d'offres.....	106

Offre globale et affectations individuelles.....	106
Définition des codes de cible par défaut.....	107
Définition d'offres non utilisées dans une règle de traitement.....	107
A propos de la table des offres globales.....	108
Affectation d'offres globales.....	109
Table des offres globales.....	109
A propos de la table de substitution de score.....	114
Configuration des substitutions de score.....	114
Table de substitution de score.....	115
Présentation de l'auto-apprentissage Unica Interact.....	120
Module d'apprentissage Unica Interact.....	121
Activation du module d'apprentissage.....	124
Attributs d'apprentissage.....	124
Définition d'un attribut d'apprentissage.....	127
Définition d'attributs d'apprentissage dynamique.....	127
Unica Interact AutoBinning.....	129
Configuration de l'environnement d'exécution pour qu'il reconnaisse les modules d'apprentissage externes.....	130
<b>Chapter 6. Compréhension de l'API Unica Interact.....</b>	<b>132</b>
Flux de données de l'API Unica Interact.....	133
Exemple simple de planification d'interaction.....	137
Conception de l'intégration de l'API Unica Interact.....	144
Points à prendre en compte.....	145
Authentification API.....	147
<b>Chapter 7. Gestion de l'API Unica Interact.....</b>	<b>148</b>

Paramètres régionaux et API Unica Interact.....	148
A propos de la surveillance JMX.....	149
Configuration d'Unica Interact pour une utilisation de la surveillance JMX avec le protocole RMI.....	149
Configuration d'Unica Interact pour une utilisation de la surveillance JMX avec le protocole JMXMP.....	150
Configuration d'Unica Interact afin d'utiliser les scripts jconsole pour la surveillance JMX.....	151
Attributs JMX.....	152
Opérations JMX.....	175
Surveillance des threads.....	176
<b>Chapter 8. Classes et méthodes de l'API Java, SOAP et REST d'Unica Interact.....</b>	<b>178</b>
Unica Interact API Classes.....	178
Méthodes pour indiquer les paramètres d'authentification si l'authentification d'API est activée avant les appels d'API.....	179
Prérequis de la sérialisation Java™ via HTTP.....	179
Prérequis SOAP.....	180
Configuration requise pour REST.....	181
API JavaDoc.....	182
Exemples d'API.....	182
Gestion des données de session.....	183
A propos de la classe InteractAPI.....	184
Terminer la session.....	184
executeBatch.....	186
getInstance.....	190
Obtenir des offres.....	191

Obtenir des offres pour des points d'interaction multiples.....	193
Obtenir le profil.....	197
Obtenir la version.....	199
postEvent.....	201
Définir l'audience.....	205
Définir le débogage.....	207
Démarrer la session.....	209
Paramètres réservés.....	217
A propos de la classe AdvisoryMessage.....	232
getDetailMessage.....	233
getMessage.....	233
getMessageCode.....	234
getStatusLevel.....	235
A propos de la classe AdvisoryMessageCode.....	235
Codes des messages de recommandation.....	236
A propos de la classe BatchResponse.....	246
getBatchStatusCode.....	246
getResponses.....	248
A propos de l'interface de commande.....	249
setAudienceID.....	250
setAudienceLevel.....	251
Définir le débogage.....	252
setEvent.....	254
setEventParameters.....	254
setGetOfferRequests.....	256

setInteractiveChannel.....	258
setInteractionPoint.....	259
setMethodIdentifier.....	260
setNumberRequested.....	261
setRelyOnExistingSession.....	262
A propos de l'interface NameValuePair.....	262
getName.....	263
getValueAsDate.....	263
getValueAsNumeric.....	264
getValueAsString.....	265
getValueDataType.....	265
setName.....	266
setValueAsDate.....	267
setValueAsNumeric.....	268
setValueAsString.....	268
setValueDataType.....	269
setScope (portée).....	270
getScope().....	271
A propos de la classe Offer.....	271
getAdditionalAttributes.....	271
getDescription.....	273
getOfferCode.....	273
getOfferName.....	274
getScore.....	275
getTreatmentCode.....	275



A propos de la classe OfferList.....	276
getDefaultString.....	277
getRecommendedOffers.....	277
A propos de la classe Response.....	278
getAdvisoryMessages.....	279
getApiVersion.....	280
getOfferList.....	280
getAllOfferLists.....	281
getProfileRecord.....	282
getSessionID.....	283
getStatusCode.....	283
<b>Chapter 9. Classes et méthodes de l'API JavaScript d'Unica Interact.....</b>	<b>286</b>
Conditions requises pour JavaScript.....	286
Gestion des données de session.....	286
Utilisation du paramètre de rappel.....	288
A propos de la classe InteractAPI.....	288
Démarrer la session.....	289
Obtenir des offres.....	297
Obtenir des offres pour des points d'interaction multiples.....	298
Définir l'audience.....	301
Obtenir le profil.....	303
Terminer la session.....	304
Définir le débogage.....	305
Obtenir la version.....	306
executeBatch.....	306

Exemple d'API JavaScript.....	308
Exemple d'objet de réponse JavaScript onSuccess.....	323
<b>Chapter 10. A propos de l'API ExternalCallout.....</b>	<b>326</b>
Interface IAffiniumExternalCallout.....	326
Ajout d'un service Web à utiliser avec la macro EXTERNALCALLOUT.....	327
getNumberOfArguments.....	328
getValue.....	328
initialiser.....	329
shutdown.....	330
Exemple d'API ExternalCallout.....	331
Interface IInteractProfileDataService.....	333
Ajout d'une source de données à utiliser avec Profile Data Services.....	334
Interface IParameterizableCallout.....	335
initialiser.....	336
shutdown.....	336
Interface ITriggeredMessageAction.....	337
getName.....	337
setName.....	337
Interface IChannelSelector.....	338
selectChannels.....	338
Interface IDispatcher.....	339
dispatch.....	340
Interface IGateway.....	341
deliver.....	342
validate.....	342

<b>Chapter 11. Utilitaires Unica Interact.....</b>	<b>344</b>
Utilitaire Run Deployment (runDeployment.sh/.bat).....	344
Nettoyage de l'utilitaire du jeton arrivé à expiration.....	351
<b>Chapter 12. A propos de l'API d'apprentissage.....</b>	<b>353</b>
Configuration de l'environnement d'exécution pour qu'il reconnaisse les modules d'apprentissage externes.....	355
Interface ILearning.....	356
initialiser.....	356
logEvent.....	357
optimizeRecommendList.....	358
reinitialize.....	360
shutdown.....	360
Interface IAudienceID.....	361
getAudienceLevel.....	361
getComponentNames.....	362
getComponentValue.....	362
IClientArgs.....	362
getValue.....	363
IInteractSession.....	363
getAudienceId.....	363
getSessionData.....	363
Interface IInteractSessionData.....	364
getDataType.....	364
getParameterNames.....	364
getValue.....	365

setValue.....	365
ILearningAttribute.....	366
getName.....	366
ILearningConfig.....	366
ILearningContext.....	367
getLearningContext.....	367
getResponseCode.....	367
IOffer.....	368
getCreateDate.....	368
getEffectiveDateFlag.....	368
getExpirationDateFlag.....	368
getOfferAttributes.....	369
getOfferCode.....	369
getOfferDescription.....	369
getOfferID.....	370
getOfferName.....	370
getUpdateDate.....	370
IOfferAttributes.....	371
getParameterNames.....	371
getValue.....	371
Interface IOfferCode.....	371
getPartCount.....	371
getParts.....	372
LearningException.....	372
IScoreOverride.....	372

getOfferCode.....	373
getParameterNames.....	373
getValue.....	374
ISelectionMethod.....	374
Interface ITreatment.....	374
getCellCode.....	375
getCellId.....	375
getCellName.....	375
getLearningScore.....	376
getMarketerScore.....	376
getOffer.....	377
getOverrideValues.....	377
getPredicate.....	377
getPredicateScore.....	378
getScore.....	378
getTreatmentCode.....	378
setActualValueUsed.....	379
Exemple d'API d'apprentissage.....	379
<b>Chapter 13. Unica Interact WSDL.....</b>	<b>388</b>
<b>Chapter 14. Propriétés de configuration de l'environnement d'exécution d'Unica</b>	
<b>Interact.....</b>	<b>411</b>
Interact   général.....	411
Interact   général   API.....	412
Interact   général   centralizedLogger.....	413
Interact   general   learningTablesDataSource.....	414

Interact   general   prodUserDataSource.....	417
Interact   general   API   requestThreadPool.....	419
Interact   general   systemTablesDataSource.....	421
Interact   general   testRunDataSource.....	428
Interact   general   contactAndResponseHistoryDataSource.....	431
Interact   général   idsByType.....	433
Interact   diagramme.....	434
Interact   diagramme   ExternalCallouts   [NomAppelExterne].....	436
Interact   diagramme   ExternalCallouts   [NomAppelExterne]   Données de paramètre   [NomParamètre].....	438
Interact   surveillance.....	439
Interact   surveillance   activitySubscribers.....	440
Interact   profil.....	442
Interact   profil   Niveaux d'audience   [Nom du niveau d'audience].....	445
Interact   profil   Niveaux d'audience   [Nom du niveau d'audience]   Offres par SQL brut.....	450
Interact   profil   Niveaux d'audience   [NomNiveauAudience   Services de données de profil   [Source de données].....	454
Affinium interact profile Audience Levels [Audience Levels]requestLogTable.....	456
Interact   offerserving.....	456
Interact   offerserving   Configuration de l'apprentissage intégré.....	461
Interact   offerserving   Configuration de l'apprentissage intégré   Données de paramètre   [Nom du paramètre].....	464
Interact   offerserving   Configuration de l'apprentissage externe.....	465
Interact   offerserving   Configuration de l'apprentissage externe   Données de paramètre   [Nom du paramètre].....	467

Interact   offerserving   Contraintes.....	467
Interact   services.....	469
Affinium interact services contactHist treatmentStoreReference.....	469
daysBackForXSessContact.....	469
Interact   services   contactHist.....	470
Interact   services   contactHist   cache.....	471
Interact   services   contactHist   contactStatusCodes.....	472
Interact   services   contactHist   fileCache.....	472
Interact   services   defaultedStats.....	473
Interact   services   defaultedStats   cache.....	474
Interact   services   statsOpsElig.....	474
Interact   services   tatsOpsElig   cache.....	475
Interact   services   eventActivity.....	476
Interact   services   eventActivity   cache.....	476
Interact   services   eventPattern.....	477
Interact   services   eventPattern   userEventCache.....	478
Interact   services   eventPattern   advancedPatterns.....	479
Interact   services   customLogger.....	483
Interact   services   customLogger   cache.....	484
Interact   services   responseHist.....	484
Interact   services   responseHist   cache.....	487
Interact   services   response Hist   responseTypeCodes.....	488
Interact   services   responseHist   fileCache.....	489
Interact   services   crossSessionResponse.....	490
Interact   services   crossSessionResponse   cache.....	492

Interact   services   crossSessionResponse   OverridePerAudience   [Niveau d'audience]   TrackingCodes   byTreatmentCode.....	493
Interact   services   crossSessionResponse   OverridePerAudience   [Niveau d'audience]   TrackingCodes   byOfferCode.....	495
Interact   services   crossSessionResponse   OverridePerAudience   [Niveau d'audience]   TrackingCodes   byAlternateCode.....	496
Interact   services   threadManagement   contactAndResponseHist.....	498
Interact   services   threadManagement   allOtherServices.....	499
Interact   services   threadManagement   flushCacheToDB.....	501
Interact   services   threadManagement   eventHandling.....	503
Interact   services   configurationMonitor.....	504
Interact   services   CampaignSegments.....	505
Interact   cacheManagement.....	507
Interact   cacheManagement   Gestionnaires de cache.....	508
Interact   caches.....	511
Interact   triggeredMessage.....	523
Interact   triggeredMessage   offerSelection.....	525
Interact   triggeredMessage   répartiteurs.....	526
Interact   triggeredMessage   passerelles   <nomPasserelle>.....	541
Interact   triggeredMessage   canaux.....	543
Interact   activityOrchestrator.....	547
Interact   activityOrchestrator   récepteurs.....	547
Interact   activityOrchestrator   passerelles.....	557
Interact   ETL   patternStateETL.....	559
Interact   ETL   patternStateETL   <NomETLEtatDeModèle>   RuntimeDS.....	561
Interact   ETL   patternStateETL   <NomETLEtatDeModèle>   TargetDS.....	564



Interact   ETL   patternStateETL   <NomETLEtatDeModèle>   Rapport.....	566
<b>Chapter 15. Unica Interact Simulateur.....</b>	<b>569</b>
Interact   simulateur.....	569
Interact   simulateur   scenarioDataSource.....	570
<b>Chapter 16. Propriétés de configuration de l'environnement de conception d'Unica</b>	
<b>Interact.....</b>	<b>575</b>
Campaign   partitions   partition[n]   reports.....	575
Campaign   partitions   partition[n]   UnicaInsightsReports.....	578
Campaign   partitions   partition[n]   Interact   contactAndResponseHistTracking.....	579
Campaign   partitions   partition[n]   Interact   contactAndResponseHistTracking   runtimeDataSources   [Source de données de l'environnement d'exécution].....	586
Campaign   partitions   partition[n]   Interact   contactAndResponseHistTracking   contactTypeMappings.....	587
Campaign   partitions   partition[n]   Interact   contactAndResponseHistTracking   responseTypeMappings.....	588
Campaign   partitions   partition[n]   Interact   rapport.....	589
Campaign   partitions   partition[n]   Interact   apprentissage.....	590
Campaign   partitions   partition[n]   Interact   apprentissage   learningAttributes   [attribut d'apprentissage].....	594
Campaign   partitions   partition[n]   Interact   déploiement.....	595
Campaign   partitions   partition[n]   Interact   serverGroups   [Groupe de serveurs]....	595
Campaign   partitions   partition[n]   Interact   serverGroups   [Groupe de serveurs]   instanceURLs   [URL de l'instance].....	596
Campaign   partitions   partition[n]   Interact   diagramme.....	597
Campaign   partitions   partition[n]   Interact   whiteList   [NiveauD'audience]   DefaultOffers.....	598

Campaign   partitions   partition[n]   Interact   whiteList   [NiveauD'audience]   offersBySQL.....	599
Campaign   partitions   partition[n]   Interact   whiteList   [AudienceLevel]   ScoreOverride.....	599
Campaign   partitions   partition[n]   server   internal.....	600
Campaign   surveillance.....	606
Campaign   partitions   partition[n]   Interact   outboundChannels.....	610
Campaign   partitions   partition[n]   Interact   outboundChannels   Données de paramètre.....	610
Campagne   partitions   partition[n]   Interact   Simulateur.....	611
offerArbitrition.....	611
<b>Chapter 17. Personnalisation d'offre en temps réel côté client.....</b>	<b>614</b>
A propos de Unica Interact Message Connector.....	614
Installation de Message Connector.....	616
Création des liens de Message Connector.....	631
A propos de Unica Interact Web Connector.....	636
Installation de Web Connector sur le serveur d'exécution.....	637
Installation de Web Connector en tant qu'application Web distincte.....	638
Configuration du Web Connector.....	640
Utilisation de la page d'administration de Web Connector.....	663
Exemple de page Web Connector.....	665
<b>Chapter 18. Intégration d'Unica Interact et de Digital Recommendations.....</b>	<b>670</b>
Présentation de l'intégration Unica Interact avec Digital Recommendations.....	670
Prérequis d'intégration.....	672
Configuration d'une offre pour l'intégration Digital Recommendations.....	673
Utilisation de l'exemple de projet d'intégration.....	674

<b>Chapter 19. Intégration d'Unica Interact et de Digital Data Exchange.....</b>	<b>690</b>
Prérequis.....	690
Intégration d' Unica Interact à votre site Web à l'aide de IBM Digital Data Exchange.....	691
Balises Unica Interact dansDigital Data Exchange.....	692
End Session.....	693
Get Offers.....	693
Load Library.....	694
Afficher l'événement.....	695
Set Audience.....	695
Démarrage de session.....	696
Exemple de paramétrage des balises.....	698
Vérifiez la configuration de votre intégration.....	704
<b>Chapter 20. Intégration d'Unica Interact et d'Unica Journey.....</b>	<b>705</b>
Présentation.....	705
Cartographie des champs Interact-Journey.....	706
Configurations d'exécution d'Interact.....	709
Deployment.....	710
<b>Chapter 21. Intégration Unica Interact et Unica Deliver.....</b>	<b>711</b>
Présentation.....	711
Cartographie Interact-Deliver.....	712
Configurations d'exécution d'Interact.....	713
Deployment.....	713
<b>Chapter 22. Configurer les passerelles.....</b>	<b>714</b>
Utilisation de Unica Interact Inbound Gateway for IBM Universal Behavior Exchange.....	715

Utilisation de Unica Interact Outbound Gateway for IBM Universal Behavior Exchange.....	724
Utilisation de Unica Interact Outbound Gateway for IBM Mobile Push Notification.....	727
Utilisation d'Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud.....	730
Ajout d'un répartiteur pour l'intégration de passerelle.....	730
Configuration du paramètre OMO-conf_outbound_common_httpConnectionConfig.....	731
Configuration du paramètre OMO-conf_outbound_silverpop_silverpopConfig.....	731
Configuration du paramètre OMO-conf_outbound_silverpop_silverpopContentMapping.....	732
Configuration du paramètre deliveryTimeoutMillis.....	732
Ajout d'un gestionnaire de canal pour Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud.....	733
Ajout d'un canal sortant pour Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud.....	733
Configuration du mailing transactionnel avec Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud.....	734
<b>Index.....</b>	

# Chapitre 1. Administration d'Unica Interact

Lorsque vous administrez Unica Interact, vous configurez et gérez les utilisateurs et les rôles, les sources de données et les fonctionnalités facultatives des produits. Vous surveillez et gérez également les environnements de conception et d'exécution. Des interfaces de programme d'application (API) spécifiques aux produits sont à votre disposition.

L'administration de Unica 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 fonctionnalités facultatives de proposition d'offre dans Unica Interact
- Contrôle et gestion des performances de l'environnement d'exécution

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

La deuxième partie du Guide d'administration décrit les API disponibles avec Unica Interact :

- Interface de programme d'application Unica Interact
- API ExternalCallout
- API Learning

## Unica Interact : concepts clés

Unica Interact est un moteur interactif qui cible les offres de marketing personnalisées pour différentes audiences.

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

## Niveaux d'audience

Un niveau d'audience correspond à une collection d'identifiants pouvant être ciblés par une campagne. Vous pouvez définir les niveaux d'audience pour cibler l'ensemble d'audiences correct pour votre campagne.

Par exemple, un ensemble de campagnes peut utiliser les niveaux d'audience "Foyer", "Prospect", "Client" et "Compte". Chacun de ces niveaux représente une vue particulière des données marketing disponibles pour une campagne.

Les niveaux d'audience 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 exemples plus complexes de hiérarchies d'audience, notamment dans les environnements business-to-business pour lesquels des niveaux d'audience peuvent exister pour les entreprises, les sociétés, les divisions, les groupes, les individus, les comptes, etc.

Ces niveaux d'audience peuvent avoir des relations différentes les unes par rapport aux autres, par exemple un à un, plusieurs à un, plusieurs à plusieurs. En définissant des niveaux d'audience, vous permettez à ces concepts d'être représentés dans Unica Campaign et aux utilisateurs de gérer les relations entre les différentes audiences 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.

## Environnement de phase de conception

Utilisez l'environnement de phase de conception pour configurer plusieurs composants Unica Interact et déployez-les dans l'environnement d'exécution.

L'environnement de phase de conception est l'environnement dans lequel vous effectuez la plus grande partie de la configuration Unica Interact. Dans l'environnement de phase

de conception, vous définissez des canaux interactifs, des diagrammes temps réel, des stratégies et des règles de traitement, des événements et des modèles d'événement, des points d'interaction, des segments dynamiques et des FlexOffers. Après avoir configuré ces composants, vous les déployez dans l'environnement d'exécution.

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

## Modèles d'événements et d'événements

### Événement

Un événement représente une activité utilisateur survenue qui peut déclencher une action dans l'environnement d'exécution. Des exemples d'événement peuvent être la visite d'un site Web, l'ouverture d'un compte courant, l'appel du service client, etc.

Les événements sont d'abord créés dans les canaux interactifs via l'interface utilisateur Interact Design Time, puis publiés dans l'environnement d'exécution Interact en appelant l'API d'exécution postEvent.

### Modèles d'événements

Un modèle d'événement consiste en une série d'événements qui se produisent d'une manière particulière. Les spécialistes du marketing peuvent utiliser des modèles d'événements pour suivre et enregistrer les modèles d'activités des clients en temps réel et agir en conséquence. Un modèle commence par l'état de modèle "condition-non-satisfaite". En publiant des événements sur Interact à des étapes sélectionnées des activités des clients, l'état du modèle est vérifié et mis à jour. Lorsque tous les événements définis pour le modèle se produisent de la manière définie, l'état du modèle est changé en "condition remplie" et les actions configurées sont déclenchées. Les modèles d'événements peuvent être utilisés dans les segmentations de clientèle et offrent des logiques d'arbitrage.

Interact prend en charge les neuf types de modèles d'événements suivants.

- Tout correspondre
- Compteur
- Compteur pondéré

- Correspondance totale (plage de temps limité)
- Compteur (plage de temps limité)
- Compteur pondéré (plage de temps limité)
- Correspondance totale (plage de temps glissante)
- Compteur (plage de temps glissante)
- Compteur pondéré (plage de temps glissante)

Match all : il s'agit d'un modèle qui se déclenche (étant défini sur l'état « condition remplie ») lorsque tous les événements de composition se produisent. Par exemple, "Event A" et "Event B" et "Event C" doivent tous se produire, alors la condition du modèle est remplie. La séquence d'occurrence des événements n'a pas d'importance.

Compteur: il s'agit d'un modèle qui se déclenche si chaque événement composant se produit plus d'un nombre de fois prédéfini. Par exemple, "l'événement A" se produit  $\geq 5$  fois et "l'événement B" se produit  $\geq 5$  fois, alors la condition du modèle est remplie. La séquence d'occurrence des événements n'a pas d'importance.

Compteur pondéré : il s'agit d'un modèle dans lequel chaque événement composant est pondéré et le modèle se déclenche lorsqu'une somme cumulée est atteinte à un nombre prédéfini de fois. Par exemple, si un modèle se compose de "l'événement A" avec le score 2 et "l'événement B" avec le score 5, et un score total est de 10, alors les conditions du modèle sont remplies lorsque l'une des situations suivantes se produit.

- "L'événement A" se produit 5 fois car  $5 \times 2 = 10$
- "L'événement B" se produit 2 fois car  $2 \times 5 = 10$
- "L'événement A" se produit 3 fois et "L'événement B" se produit 1 fois car  $3 \times 2 + 1 \times 5 = 11$ .

Pour les modèles de type Tout correspondre, Compteur et Compteur pondéré, il n'y a pas de contraintes de temps pour eux. Tant que les événements publiés correspondent à des dates de début et de fin définies, ils sont évalués pour le modèle. Si la date de début n'est pas définie, le modèle commence à être effectif immédiatement une fois déployé. Si la date de fin n'est pas définie, le modèle est efficace pour toujours. Les spécialistes du marketing peuvent utiliser la fonction de réinitialisation du modèle pour réinitialiser l'état du modèle



pour ces trois types de modèles. En revanche, les modèles limités dans le temps et les modèles de temps roulant sont des modèles limités dans le temps.

**Modèle de temps de roulement:** un modèle de temps de roulement peut être un modèle "Tout correspondre", "Compteur" ou "Compteur pondéré", mais tous les événements de composition doivent se produire dans une fenêtre de temps. À tout moment lorsqu'un événement de composition est publié dans Interact Runtime, Interact vérifie les occurrences de tous les événements de composition du modèle dans la fenêtre temporelle à partir du point temporel actuel. Si les occurrences d'événement ne répondent pas à la définition du modèle, l'état du modèle reste « condition non remplie ». Sinon, si tous les événements se produisent dans la fenêtre de temps, l'état du modèle est défini sur « condition remplie » (peut déclencher des actions si configuré). Après cela, l'état du motif est continuellement réévalué de la même manière que ci-dessus et est répété sur une base roulante

**Modèle limité dans le temps :** un modèle limité dans le temps peut être un modèle "Tout correspondre", "Compteur" ou "Compteur pondéré", mais tous les événements de composition doivent se produire dans une fenêtre temporelle. À tout moment, lorsqu'un événement de composition est publié dans Interact Runtime, Interact vérifie les occurrences de tous les événements de composition du modèle dans la fenêtre temporelle à partir du point temporel actuel. Si les occurrences d'événement ne répondent pas à la définition du modèle, l'état du modèle reste « condition non remplie ». Sinon, si tous les événements se produisent dans la fenêtre de temps, l'état du modèle est défini sur "condition remplie" (peut déclencher des actions si configuré). Désormais, Interact vérifie un autre paramètre appelé "Prolonger l'état vrai pendant une période supplémentaire" et conserve le modèle à l'état "condition remplie" pendant la période supplémentaire (aucune évaluation de modèle pendant cette période). Lorsque le temps supplémentaire s'écoule, l'état du motif est réinitialisé à « condition non remplie » et l'évaluation démarre un autre cycle. En d'autres termes, le modèle lié au temps permet au modèle de s'arrêter pendant un certain temps après la condition remplie. Le paramètre "Prolonger l'état vrai pendant une période supplémentaire" s'applique uniquement au modèle Time Bound.

Par exemple, P1 est un modèle Time Bound et P2 est un modèle Rolling Time. Les deux modèles consistent en "l'événement A" et "l'événement B", et ils doivent se produire dans les 7 jours. En temps d'exécution, "l'événement A" s'est produit le lundi et "l'événement

B" s'est produit le samedi. Lorsque "l'événement B" s'est produit, l'état du modèle est modifié en "condition remplie" pour P1 et P2 car deux événements se sont produits en 7 jours. Maintenant, pour P1, si le paramètre "Prolonger l'état vrai pendant une période supplémentaire" est de 4 jours, alors P1 reste dans l'état "condition remplie" jusqu'à mercredi, puis toutes les occurrences de deux événements sont effacées et le modèle commence à partir du scratch mercredi. Au contraire, l'état de P2 est évalué en continu après le samedi. Si "l'événement B" se produit mardi, l'état de P1 deviendra "condition non remplie" car "l'événement A" ne s'est pas produit du dernier mercredi à ce mardi.

Événement de qualification et événement de suspension: un modèle d'événement est composé d'une série d'événements. Les événements qui font passer l'état du modèle à "condition remplie" sont appelés événements de qualification. Alors que les événements qui mettent le modèle en pause pour l'évaluation sont appelés événements de suspension. Par exemple, un modèle a deux événements, "open\_bank\_account", "ATM\_activity" et "offer\_credit\_card", tous doivent se produire dans 2 mois. Si un client a déjà fait une demande et obtenu la carte de crédit de la banque dans un délai d'un mois à compter de l'ouverture du compte, les spécialistes du marketing ne voudront plus déranger le client en lui proposant une carte. Par conséquent, les spécialistes du marketing peuvent définir un événement de suspension "got\_card" dans le modèle qui mettra le modèle en pause pour évaluation. Les spécialistes du marketing peuvent également utiliser le paramètre "Durée effective" pour définir si le modèle est suspendu pour toujours ou juste pour une période de temps.

Macro d'événement: Outre les événements définis par les clients, Interact prend également en charge six macros d'événement qui peuvent participer à la définition de modèles, en tant qu'événements de qualification et événements de suspension. Voici les six macros.

- offerAccepted
- offerContacted
- offerRejected
- offerAcceptedInCategory
- offerContactedInCategory
- offerRejectedInCategory

offerAccepted, offerContacted ou offerRejected pour une offre peut être servie en tant qu'événement dans un modèle. offerAcceptedInCategory, offerContactedInCategory, ou offerRejectedInCategory peuvent avoir toutes les offres qui ont une valeur d'attribut similaire en tant qu'événement dans un modèle.

Modèle d'inactivité : un modèle peut non seulement être évalué pour la « condition remplie », mais également pour la « condition non remplie ». Les spécialistes du marketing peuvent utiliser cette fonctionnalité pour suivre les inactivités des clients. Par exemple, un modèle a deux événements, "add\_item\_to\_cart" et "checkout", tous doivent se produire en sept jours. Les spécialistes du marketing peuvent ajouter un point de contrôle le 3ème jour, si le client n'a pas encore vérifié l'article, c'est-à-dire que l'état du modèle est "condition non remplie", alors une action de "send\_reminder\_email" sera exécutée pour le client.

## Catégorie d'événement

Les événements ou les modèles d'événements peuvent être organisés en catégories pour votre commodité dans l'environnement de conception. Les catégories d'événements n'ont pas d'objectif fonctionnel dans l'environnement d'exécution.

## Actions

Une action peut être déclenchée lorsqu'un événement se produit ou lorsque les conditions du modèle d'événement sont remplies ou non. Ils sont configurés dans Interact Design Time lorsque vous définissez des événements ou des modèles d'événements.

Interact prend en charge huit types d'actions.

- Déclencher la resegmentation: l'environnement d'exécution exécute tout ou partie des diagrammes interactifs pour le niveau d'audience actuel qui est à nouveau associé au canal interactif, en utilisant les données actuelles de la session du visiteur. Ceci est utile pour placer le visiteur dans de nouveaux segments après que de nouvelles données significatives ont été modifiées dans l'objet de session d'exécution, telles que de nouvelles données provenant de requêtes de l'API d'Unica Interact (telles que la modification de l'audience) ou des actions client (telles que l'ajout de nouveaux éléments à une liste de souhaits ou un panier). Il convient de noter qu'une

resegmentation excessive au cours d'une seule visite peut affecter les performances du point de contact d'une manière visible par le client.

- Consigner le contact de l'offre: l'environnement d'exécution marque les offres recommandées pour que le service de base de données enregistre les offres dans l'historique des contacts. Pour les intégrations Web, conservez le contact de l'offre dans l'appel dans lequel vous demandez des offres afin de limiter le nombre de demandes entre le point de contact et le serveur d'exécution. Si le point de contact ne spécifie pas le code de traitement de l'offre qu'Unica Interact a présentée au visiteur, l'environnement d'exécution enregistre la dernière liste d'offres recommandées
- Consigner l'acceptation de l'offre: l'environnement d'exécution marque l'offre sélectionnée pour que le service de base de données enregistre dans l'historique des réponses.
- Consigner le rejet de l'offre: l'environnement d'exécution marque l'offre sélectionnée pour que le service de base de données enregistre dans l'historique des réponses
- Déclencher l'expression utilisateur : une action d'expression est une action dans laquelle vous pouvez définir la valeur d'une variable de session à l'aide d'attributs de profil, d'attributs en temps réel, ainsi que de macros Unica Interact, y compris des fonctions, des variables et des opérateurs, notamment EXTERNALCALLOUT. Vous pouvez affecter la valeur de retour de l'expression à n'importe quel attribut de profil
- Déclencher des événements: vous pouvez utiliser l'action Déclencher des événements pour déclencher un ou plusieurs autres événements lorsqu'un événement source se produit. Cela permet aux marketeurs d'enchaîner les événements.
- Supprimer les offres. La suppression des offres peut être déclenchée à partir d'événements et de modèles d'événements. Les règles de suppression peuvent être définies sur la base d'offres spécifiques ou d'un groupe d'offres ayant les mêmes valeurs d'attributs. La différence entre l'action de suppression d'offre et les règles de suppression existantes est que la première peut être déclenchée sans rapport avec les règles de traitement.
- Qualifiez les segments. L'utilisateur peut spécifier quel segment est activé à la suite d'un événement ou d'un modèle d'événement.

En plus d'invoquer des actions immédiatement lorsqu'un événement se produit ou qu'une condition de modèle est remplie, les actions peuvent également être appelées avec un délai,

soit retardé après une période de temps, soit à la date et à l'heure prévues. Cela donne aux spécialistes du marketing le contrôle sur l'exécution des actions aux moments préférés. Le délai d'action ne s'applique pas à la "suppression d'offre" et aux "segments éligibles"

## Canaux interactifs

Utilisez les canaux interactifs dans Unica Interact pour coordonner tous les objets, les données et les ressources serveur nécessaires au marketing interactif.

Un canal interactif est une représentation dans Unica Interact 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 temps réel

Utilisez les diagrammes interactifs pour diviser vos clients en segments et affecter un profil à un segment.

Un diagramme temps réel est associé à diagramme de traitement par lots Unica Campaign mais présente de légères différences. Les diagrammes temps réel exécutent la même fonction principale que les diagrammes par lots : ils divisent vos clients en groupes appelés segments. Cependant, dans le cas des diagrammes temps réel, les groupes sont des segments dynamiques. Unica Interact utilise ces diagrammes temps réel pour affecter un profil à un segment lorsqu'un comportemental ou système indique qu'une resegmentation du visiteur est nécessaire.

Les diagrammes temps réel contiennent un sous-ensemble des processus de diagrammes par lots ainsi que plusieurs processus qui leur sont propres. L'option "Mettre à jour" n'est pas disponible dans les diagrammes temps réel.



**Remarque :** Les diagrammes temps réel peuvent être créés uniquement dans une session Unica Campaign.



**Remarque :** Pour tester l'exécution d'un diagramme temps réel, il est recommandé d'utiliser le groupe de serveurs plutôt que le groupe de serveurs de production.



**Remarque :** Vous ne pouvez pas utiliser de formats DT\_DELIM\_XXX avec des diagrammes de session temps réel.

## Points d'interaction

Un point d'interaction est un emplacement de votre point de contact où vous souhaitez présenter une offre.

Les points d'interaction contiennent un contenu 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 livré de différentes façons.

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

Une offre peut être réutilisée :

- Dans différentes campagnes
- A différents points dans le temps

- Pour différents groupes de personnes (cibles)
- En tant que "versions" différentes en variant ses zones paramétrées

Vous affectez des offres aux points d'interaction dans les points tactiles présentés aux visiteurs.

Interact prend en charge les états de gestion centralisée des offres "DRAFT", "PUBLISHED" et "RETIRED". Seules les offres "PUBLISHED" peuvent être utilisées et déployées dans Interact. Si une offre «PUBLIÉE» dans Interact est reformulée ou retirée, le statut correspondant « (redraft)/(retired) » s'affiche avec l'offre.



**Note:** Chaque offre utilisée dans Interact doit avoir un code d'offre unique. La casse n'a pas d'importance. De plus, chaque attribut d'offre doit avoir un nom unique. La casse n'a pas d'importance.

## 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.

Les données 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 niveau d'audience 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 Unica 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. L'environnement d'exécution peut être constitué d'un ou plusieurs serveurs d'exécution connectés à un point de contact.

L'environnement d'exécution utilise les informations déployées à partir de l'environnement de conception en combinaison avec l'API Unica 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 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 Unica Interact, vous pouvez intégrer Unica 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).



## Stratégie et règles de traitement

Un canal interactif peut posséder plusieurs stratégies marketing. Une stratégie, point clé de l'application Interact, est un ensemble de règles de traitement. Les règles de traitement sont également appelées règles intelligentes dans Interact après la version 12.0. Les règles de traitement permettent d'affecter une offre à un segment dynamique. Ces affectations sont soumises à des contraintes supplémentaires imposées par le champ personnalisé que vous associez à l'offre dans la règle de traitement.

Par exemple, vous affectez 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".

Chaque règle de traitement possède également un score marketing. Si plusieurs segments ont été affectés à un client et que plusieurs offres lui sont donc applicables, les scores marketing permettent de définir l'offre qu'Interact doit suggérer. Le score d'offre peut disposer d'un score fixe, appelé Score du spécialiste du marketing, ou d'un score dynamique, défini sous la forme d'une expression (appelé également Prédicat dans Interact) de profil ou d'attributs d'offre. Interact Runtime se sert de cette expression pour calculer le score de l'offre en fonction des valeurs des attributs lors de l'exécution. L'éligibilité de l'offre permet de déterminer si une offre est éligible ou non, même si elle est activée. Une offre est éligible uniquement si elle se trouve dans sa période d'effet (entre sa date d'effet et sa date d'expiration) et/ou sur une expression est évaluée comme vraie lors de l'exécution. Lorsqu'Interact présente une offre aux utilisateurs finaux, au lieu de récupérer les valeurs des attributs de l'offre à partir de l'offre, Interact est capable de passer outre les valeurs des attributs de l'offre, même avec une expression calculée depuis les données de profil lors de l'exécution d'Interact. Les utilisateurs peuvent définir les attributs d'offre paramétrés pour l'offre dans une règle de traitement. Les offres que l'environnement d'exécution suggère peuvent être influencées par un module d'apprentissage, une liste de suppression d'offres et des affectations d'offres individuelles ou globales.

## FlexOffers

Un canal interactif peut être configuré pour contenir plusieurs mappages FlexOffer.

FlexOffers offre un moyen plus simple d'affecter des offres directement aux clients ciblés

correspondants. Le mappage FlexOffers peut être créé à partir d'une table déjà créée ou en important un fichier CSV contenant les données de mappage requises ou en créant une nouvelle table de règles.

Chaque mappage peut avoir plusieurs règles et filtres. Chaque règle peut être utilisée pour affecter des offres en fonction de différents attributs personnalisés. Ces affectations peuvent faire l'objet de contraintes supplémentaires à l'aide des zones et des cibles associées à l'offre dans la règle. En outre, une règle peut être définie de manière à ce que n'importe quel nombre d'attributs personnalisés lui soit associé.

Chaque règle possède également un score marketing. Si un client peut bénéficier de plusieurs offres, le score marketing permet de déterminer l'offre suggérée par l'application Interact. Le score marketing peut avoir une valeur statique ou une valeur dynamique définie en tant qu'expression d'attributs d'offre. Cette expression est ensuite utilisée pour calculer le score marketing par l'environnement d'exécution Interact.

L'éligibilité de l'offre permet de déterminer si une offre est éligible ou non, même si elle est activée. Une offre est éligible uniquement si elle se trouve dans sa période d'effet (entre sa date d'effet et sa date d'expiration) et/ou si une expression est évaluée comme vraie lors de l'exécution. Lorsqu'Interact présente une offre aux utilisateurs finaux, au lieu de récupérer les valeurs des attributs de l'offre à partir de l'offre, Interact est capable de passer outre les valeurs des attributs de l'offre, même avec une expression calculée depuis les données de profil lors de l'exécution d'Interact. Les utilisateurs peuvent définir les attributs d'offre paramétrés pour l'offre dans la règle FlexOffers. Les offres que l'environnement d'exécution suggère peuvent être influencées par un module d'apprentissage, une liste de suppression d'offres et des affectations d'offres individuelles ou globales.

Des filtres peuvent être appliqués aux règles pour obtenir les offres requises pour les clients ciblés. Chaque filtre possède des conditions relatives aux attributs de règle. Ces conditions lors de l'exécution déterminent l'ensemble d'offres présentées au client. Un certain nombre de filtres peuvent être appliqués simultanément aux règles pour obtenir les offres requises.

Par exemple, vous pouvez disposer de plusieurs règles ayant des offres différentes, liées à des attributs tels que l'emplacement et les dépenses totales d'un client. Vous pouvez créer des filtres comportant des conditions relatives à ces attributs. En fonction de ces conditions, les offres sont affichées pour le client lors de l'exécution.

Les règles et les filtres sont définis sous l'onglet FlexOffers d'un canal interactif.

Sous l'onglet 'FlexOffers', vous pouvez créer le mappage, copier vers le groupe de serveurs requis, créer des règles directement ou en importer à partir d'un fichier, ajouter de nouvelles règles et critères, modifier ou supprimer des règles et des critères uniques ou multiples, dupliquer des règles et créer des filtres. Pour plus de détails, voir le guide d'utilisation d'Interact.

Le mappage FlexOffer ainsi que ses règles et filtres fournissent une solution pour personnaliser les offres en fonction d'un certain nombre d'attributs personnalisés et récupérer ces offres en appliquant différentes conditions à ces attributs.

Lors de l'extraction d'offres depuis l'environnement d'exécution Interact, les filtres sont appliqués conformément à la logique suivante :

UACIEnableOfferMappingFilter : Ce paramètre est défini avec le nom du filtre lors de l'exécution lors de startSession ou de getOffers. Le filtre spécifique sera appliqué pour obtenir les offres de la table de mappage FlexOffer.

UACIDisableOfferMappingFilter : Ce paramètre est défini avec le nom du filtre lors de l'exécution lors de startSession ou de getOffers. Le filtre spécifique ne sera pas appliqué lors de l'obtention des offres de la table.

En outre, tous les filtres marqués comme filtres par défaut, s'ils ne sont pas désactivés, seront appliqués à la table pour obtenir des offres

## Passerelles

Interact prend en charge les passerelles entrantes et sortantes. Cependant, toutes les configurations ont été effectuées via des fichiers de propriétés, difficiles à gérer et sujets aux erreurs.

Un canal interactif peut être configuré pour y définir plusieurs mappages de passerelle.

Vous pouvez créer des mappages de passerelle pour les éléments suivants.

- Journey sortant
- Deliver sortant

- Sortant générique
- Entrant générique

Pour plus de détails sur le trajet aller, consultez le [Intégration d'Unica Interact et d'Unica Journey \(à la page 705\)](#) section.

Pour plus de détails sur la livraison sortante, consultez le [Intégration Unica Interact et Unica Deliver \(à la page 711\)](#) section.

Sortant générique : Il peut être utilisé pour définir les mappages pour les passerelles, qui sont configurées pour être utilisées pour la communication sortante.

- Nombre de messages : Il définit le nombre de messages qui seront envoyés via la passerelle
- Priorité : définit la priorité de la passerelle, qui est une valeur numérique. La combinaison du nom et de la priorité identifie de manière unique la passerelle dans le canal interactif englobant. La passerelle ayant une valeur de priorité minimale sera déployée sur l'environnement d'exécution Interact.
- Propriétés du canal – Ceci est utilisé pour définir les propriétés au format clé-valeur qui sont nécessaires pour la passerelle.
- Mappage – Ceci est utilisé pour définir le mappage entre le champ de point de terminaison et le champ Interact.

Entrant générique : cela peut être utilisé pour définir les mappages pour les passerelles qui sont configurées pour être utilisées pour la communication entrante.

- Priorité : définit la priorité de la passerelle, qui est une valeur numérique. La combinaison du nom et de la priorité identifie de manière unique la passerelle dans le canal interactif englobant. La passerelle ayant une valeur de priorité minimale sera déployée sur l'environnement d'exécution Interact.

- Propriétés du canal - Ceci est utilisé pour définir les propriétés au format clé-valeur qui sont nécessaires pour la passerelle. Ces propriétés sont transmises en tant que paramètres à l'API `startSession`.
- Mappage - Ceci est utilisé pour définir le mappage des paramètres entre l'événement Interact et l'événement de point de terminaison. Ces propriétés sont transmises en tant que paramètres à l'API `postEvent`.

Voir le [Configurer les passerelles \(à la page 714\)](#) section pour plus de détails sur la configuration des mappages pour la passerelle Email, MobilePush et UBX.

## Implémentation d'une passerelle générique entrante ou générique sortante

Interact fournit une implémentation prête à l'emploi pour les passerelles entrantes et sortantes génériques. Les utilisateurs peuvent configurer les propriétés et le mappage du canal à l'aide de la passerelle générique sortante ou entrante et utiliser ces implémentations d'Interact en créant des passerelles à l'aide du modèle `InteractGateway` dans la configuration d'Interact runtime `triggerMessage/activityOrchestration`.

Les deux implémentations prennent en charge Kafka comme canal de communication entre Interact et les systèmes tiers.

## Implémentation d'une passerelle entrante générique

Pour traiter le message entrant, Interact requiert des informations sous la forme d'une configuration afin de mapper les zones de message entrant aux propriétés Interact correspondantes. La configuration de l'interface utilisateur doit définir toutes les propriétés obligatoires dont Interact a besoin pour appeler les API Interact : `startSession`, `postEvent` et `endSession`. Vous pouvez effectuer la configuration en accédant à « Canal interactif » -> « Passerelles » -> « Passerelle entrante générique ».

Voici un exemple de message entrant via le canal Kafka. Le récepteur d'activité Kafka impose les propriétés « **passerelle** » et « **message** » pour identifier le canal à traiter. « **message** » contient les informations envoyées à la passerelle entrante pour traitement.

```
{
  "gateway": "GenericIn",
  "message":
  {
    "ICName": "SB_InteractiveChannel",
    "audienceID": [
      {
        "n": "CustomerID",
        "v": "1",
        "t": "numeric"
      }
    ],
    "events": [
      {
        "event": "EP_contact",
        "parameters": [
          {
            "n": "UACIOfferTrackingCode",
            "v": "5.2.ffffffffffe4699811.4fad551",
            "t": "string"
          }
        ]
      }
    ],
    "parameters": [
      {
        "n": "country",
        "v": "INDIA",
        "t": "string"
      },
      {
        "n": "UACILogSeparationFileName",
```

```

        "v": "log123",
        "t": "string"
    }
],
"CH_debug": "true",
"CH_sessionID": "session1"
}
}

```

## Configuration de passerelle entrante

La configuration doit fournir un mappage des propriétés de message entrant avec les paramètres d'API `startSession`, `postEvent` et `endSession`.

### Propriétés `startSession`

Interact requiert les informations suivantes pour l'API `startSession`.

- SessionId
- Nom du canal interactif
- niveau d'audience
- ID audience
- Paramètres
- Rely on existing Session
- Déboguer

Tenez compte des remarques suivantes :

- Le nom du canal interactif doit être fourni par le nom de propriété réservé « `ICName` ». Cela est obligatoire pour tous les messages entrants. Par exemple :

`ICName` : `SB_InteractiveChannel`

- Le niveau d'audience, l'ID d'audience et « `Rely on existing Session` » sont mappés à partir de l'onglet « Général » de la configuration de la passerelle entrante générique.

- Pour les ID d'audience, les utilisateurs peuvent configurer n'importe quelle propriété de nœud final à mapper et à traiter à partir du message entrant. La valeur de l'ID d'audience doit suivre le format prédéfini ci-dessous.

```
"EndPointField_audienceID": [  
  {  
    "n": "CustomerID",  
    "v": "1",  
    "t": "numeric"  
  }  
]
```

- Le mappage de l'ID de session et des indicateurs de débogage peut être configuré à partir de l'onglet « Propriétés du canal » de la passerelle entrante générique. Toutes les propriétés définies dans l'onglet Propriétés du canal sont également transmises en tant que paramètres startSession. Les utilisateurs peuvent donc accéder à cet onglet pour définir les paramètres de démarrage de session. En outre, les utilisateurs peuvent transmettre les paramètres startSession sous le nom de propriété réservée « parameters », comme indiqué ci-dessous.

```
"parameters": [  
  {  
    "n": "country",  
    "v": "INDIA",  
    "t": "string"  
  },  
  {  
    "n": "UACILogSeparationFileName",  
    "v": "log123",  
    "t": "string"  
  }  
]
```



## Propriétés postEvent

Le nom d'événement et les paramètres d'événement peuvent être mappés à partir de l'onglet « Mappage » dans la configuration entrante générique.

Tous les mappages de paramètres d'événement sont transmis en tant que paramètre d'événement pour l'appel de l'API postEvent. En outre, s'ils le souhaitent, les utilisateurs peuvent utiliser le nom de propriété réservée « parameters » sous « events » pour définir les paramètres d'événement comme indiqué ci-dessous.

```
"events": [  
  {  
    "event": "EP_contact",  
    "parameters": [  
      {  
        "n": "UACIOfferTrackingCode",  
        "v": "5.2.ffffffffffe4699811.4fad551",  
        "t": "string"  
      }  
    ]  
  }  
]
```

## Propriétés endSession

La propriété de fin de session est mappée à partir de l'onglet « Général ». Les utilisateurs peuvent également remplacer cette configuration en fournissant une valeur à la propriété réservée « endSession » dans le message entrant.

## Noms des zones réservées de la passerelle entrante

Voici les noms de zone réservés que le système tente d'examiner dans le message entrant, soit lorsqu'il n'est pas en mesure de les trouver via la configuration de passerelle, soit lorsque les utilisateurs souhaitent les remplacer.

Voici les zones réservées que le système recherche dans le cas où il ne trouve pas le mappage dans la configuration de passerelle.

- ICName
- paramètres
- debug
- sessionId

Voici les zones pour lesquelles les utilisateurs peuvent remplacer la valeur de configuration en les fournissant dans le message entrant.

- endSession
- relyOnExistingSession

## Configuration d'exécution

Créez une passerelle à l'aide du modèle `Affinium|interact|activityOrchestrator|gateways|` (`InteractGateway`) Le nom de la passerelle doit correspondre au nom de passerelle donné pour la passerelle entrante générique créée à partir de la phase de conception d'Interact. Pour recevoir le message entrant, vous devez créer un type de récepteur « Kafka » à partir de « `Affinium|interact|activityOrchestrator|receivers` » et ajouter les paramètres requis pour la connexion à Kafka en tant que consommateur.

## Implémentation d'une passerelle sortante générique

La configuration de passerelle sortante permet d'identifier les informations à envoyer dans le cadre du message sortant. Cette implémentation prête à l'emploi pour la passerelle sortante générique est propre aux passerelles Kafka.

Voici un exemple de message sortant produit par l'implémentation de passerelle sortante :

```
{
  "Gateway": "GenericOut",
  "Channel": "testChannel",
  "ic": "SB_InteractiveChannel",
```

```
"ProcessTime": 1609841939584,
"audienceLevel": "Customer",
"audienceID": [
  {
    "t": "numeric",
    "v": 1,
    "n": "CUSTOMERID"
  }
],
"OfferName": "Offer1",
"TreatmentCode": "0.2.6e0cce60.ffffffff49103c0",
"OfferCode": ["000000001"],
"EP_expiration": "00/02/2012",
"EP_Fulfillment Cost": "10",
"EP_NAME": "Raphael Villareal",
"EP_defaultField2": "12345",
"EP_Field1": "InteractField1"
}
```

## Zones par défaut

Voici les zones qui font partie de chaque message sortant par défaut.

- Passerelle
- Canal
- Canal interactif
- Horodatage de traitement
- niveau d'audience
- ID audience
- OfferName
- OfferCode
- TreatmentCode

## Attributs et profils d'offre

Les propriétés d'offre et de profil devant être envoyées dans le cadre du message sortant peuvent être configurées via l'onglet « Mappage ». Les valeurs sont validées selon des critères de taille et de format de date, si ces critères sont configurés. De même, les zones configurées comme obligatoires sont validées avant que les messages ne soient envoyés.

Les zones de nœud final avec des valeurs par défaut peuvent être configurées à partir de l'onglet « Propriétés du canal ». Toutes les zones configurées dans cet onglet sont envoyées dans le cadre de chaque message sortant.

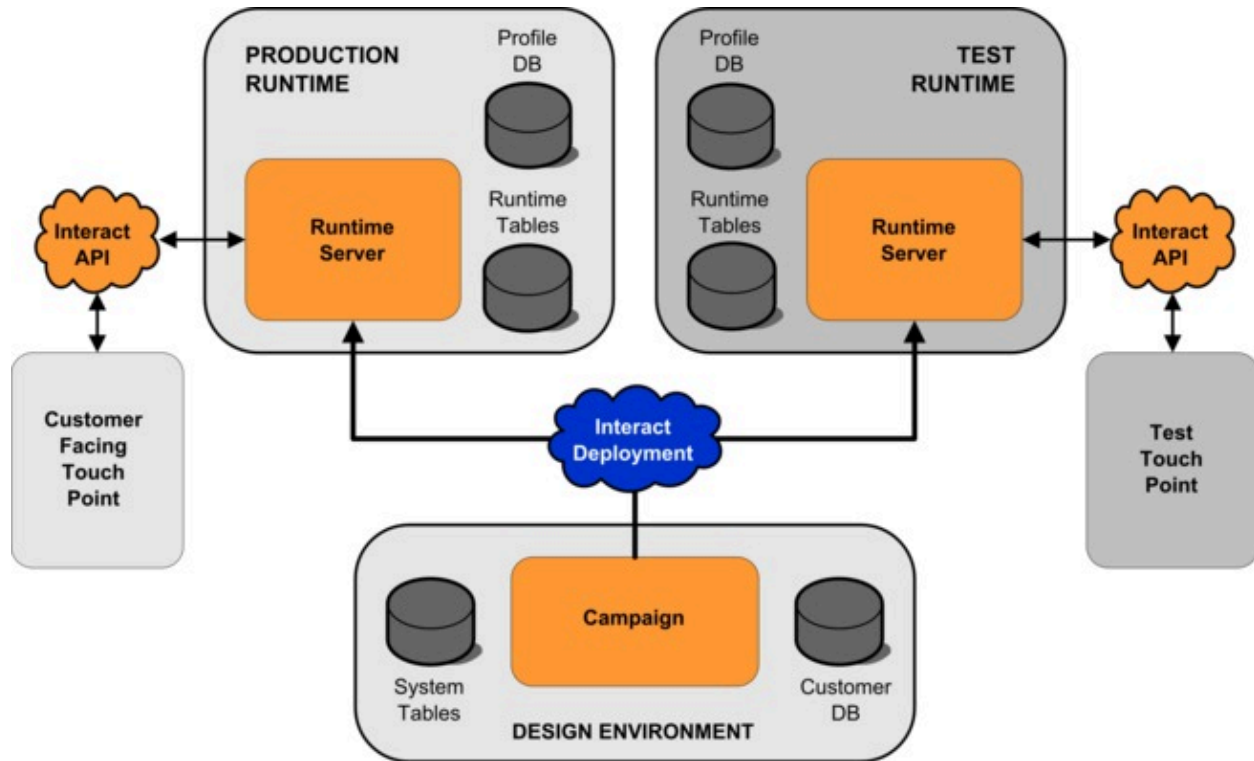
## Configuration d'exécution

Les utilisateurs doivent configurer une passerelle en accédant à « Affinium|interact|triggeredMessage|gateways|InteractGateway ». Le nom de cette passerelle doit correspondre à celui de la passerelle sortante générique créée à partir de l'interface utilisateur. Le modèle nécessite que les utilisateurs fournissent les détails de connexion Kafka pour l'envoi du message.

## Architecture Unica Interact

L'environnement Unica 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 Unica Interact. L'environnement de conception est installé avec l'application Web Unica Campaign et fait référence aux tables système Unica 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 Unica 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 Unica Interact et les utilisez pour demander des informations au serveur d'exécution.

## Considérations réseau Unica Interact

Une installation de production de Unica Interact concerne au moins deux machines.

Dans un environnement de production traitant de gros volumes, avec plusieurs serveurs d'exécution Unica 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 mise en oeuvre de l'API Unica 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 Unica Interact. Vous pouvez configurer la gestion de session des serveurs d'exécution Unica Interact pour le type de cache local.

- Si votre mise en oeuvre de l'API Unica 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 Unica 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 Unica Interact pour un `cacheType` distribué. Si vous utilisez un cache distribué, tous les serveurs d'exécution Unica Interact doivent pouvoir communiquer via la multidiffusion. Vous devrez peut-être régler votre réseau de sorte que la communication entre les serveurs Unica 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.

- La mise en cache distribuée entre plusieurs groupes de serveurs n'est pas prise en charge.
- Placez les serveurs Unica Interact de votre environnement d'exécution, Unica 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 Unica 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 Unica Campaign) ainsi que d'accéder à la base de données d'exécution (tables d'exécution Unica 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 installation de test ou de transfert, vous pouvez installer les phases de conception et d'exécution Unica Interact sur la même machine. Ce scénario n'est pas recommandé dans un environnement de production.

## Ports de serveur et sécurité des réseaux Unica Interact

Configurez Unica Interact pour sécuriser vos ports de serveur.

## Ports d'exécution Unica Interact

Certains de ces ports peuvent être fermés, ou ne sont pas requis par toutes les installations Unica Interact, selon votre configuration.

### Port de serveur d'application Unica Interact pour HTTP

Port par défaut sur lequel les requêtes Unica Interact sont gérées.

### Port de serveur d'application Unica Interact pour HTTPS

Port SSL par défaut sur lequel les requêtes Unica Interact sont gérées.

### Port Unica Interact `systemTablesDataSource`

Voir la configuration JDBC de la source de données dans Unica Platform.

### Port Unica Interact `learningTablesDataSource`

Voir la configuration JDBC de la source de données dans Unica Platform.

### Port Unica Interact `contactAndResponseHistoryDataSource`

Voir la configuration JDBC de la source de données dans Unica Platform.

### Port Unica Interact `prodUserDataSource`

Voir la configuration JDBC de la source de données dans Unica Platform.

### Port Unica Interact `testRunDataSource`

Voir la configuration JDBC de la source de données dans Unica Platform.

### Port de communication d'extraction, transformation et chargement

Configurez ce port dans **Unica Interact | ETL | patternStateETL | communicationPort** dans les propriétés de configuration.

### Port de multidiffusion `EHCache`

Configurez ce port dans **Unica Interact | cacheManagement | Cache | Gestionnaires | EHCache | Données de paramètre | multicastPort** dans les propriétés de configuration lorsque le mode de mise en cache est distribué.

### Port de surveillance `JMX Unica Interact`



Configurez ce port dans **Unica Interact | surveillance | port** dans les propriétés de configuration ou exécutez

```
-Dinteract.jmx.monitoring.port=portNumber.
```

### Port Unica Interact WebConnector

Ce port est généralement le même que le port de serveur Unica Interact, mais il est modifiable dans `jsconnector.xml`.

Pour les ports des produits intégrés Unica Interact, voir la documentation de ces produits.

La surveillance JMX n'est pas requise pour la fonctionnalité Unica Interact classique. Elle est cependant utilisée pour le diagnostic et la surveillance.

L'accès au port JMX peut être désactivé dans la configuration Unica Interact ou limité à une adresse IP spécifique via la configuration de pare-feu. Cette opération est recommandée en raison de la vulnérabilité de JMX récemment prouvée dans la bibliothèque commune Apache tierce partie.

La fonctionnalité JMX Remoting d'Apache Geronimo 3.x (avant la version 3.0.1), telle qu'utilisée dans IBM WebSphere Application Server (WAS) Community Edition 3.0.0.3 et d'autres produits, n'implémente pas correctement le chargeur de classe RMI, ce qui permet à des agresseurs éloignés d'exécuter un code arbitraire à l'aide du connecteur JMX pour envoyer un objet sérialisé artisanal.

## Ports de conception Unica Interact

Certains de ces ports peuvent être fermés, ou ne sont pas requis par toutes les installations Unica Interact, selon votre configuration.

### Port de serveur d'application Unica Campaign pour HTTP

Port par défaut sur lequel les requêtes Unica Interact sont gérées.

### Port de serveur d'application Unica Campaign pour HTTPS

Port SSL par défaut sur lequel les requêtes Unica Interact sont gérées.

### Port d'écoute Unica Campaign

**Port utilisé en interne par Unica Campaign pour accepter les connexions du client Web.**

## Autres ports de conception Unica Campaign

Pour plus d'informations sur ces ports, voir la documentation Campaign.

## Port du connecteur JMX Unica Campaign

Configurez ce port dans **Unica Campaign | surveillance | port** dans les propriétés de configuration pour la surveillance de l'historique des réponses de contact uniquement.

## Port du serveur de surveillance opérationnel Unica Campaign

Configurez ce port dans **Unica Campaign | surveillance | serverURL** dans les propriétés de configuration.

# Connexion à Unica

Utilisez la procédure ci-dessous pour vous connecter à Unica.

Vous devez disposer des éléments suivants :

- Une connexion Intranet (réseau) permettant d'accéder à votre serveur Unica.
- Un navigateur pris en charge installé sur votre ordinateur.
- Un nom d'utilisateur et un mot de passe permettant de se connecter à Unica.
- Adresse URL permettant d'accéder à Unica sur votre réseau.

L'adresse URL est la suivante :

`http://host.domain.com:port/unica`

où

`host` est la machine où Unica Platform est installé.

`domain.com` est le domaine dans lequel réside la machine hôte.

`port` est le numéro de port sur lequel le serveur d'application Unica Platform écoute.



**Remarque :** La procédure suivante suppose que vous êtes connecté avec un compte disposant d'un accès Admin à Unica Platform.

Accédez à l'URL d'Unica à l'aide de votre navigateur.

- Si Unica est configuré pour s'intégrer à Windows™ Active Directory ou une plateforme de contrôle d'accès Web et que vous êtes connecté à ce système, la page du tableau de bord par défaut s'affiche. Votre connexion est terminée.
- Si l'écran de connexion s'affiche, connectez-vous à l'aide des droits d'administrateur par défaut. Dans un environnement à partition unique, utilisez l'identifiant `asm_admin` et le mot de passe `password`. Dans un environnement à plusieurs partitions, utilisez l'identifiant `platform_admin` et le mot de passe `password`.

Vous êtes invité à changer de mot de passe. Vous pouvez réutiliser le mot de passe existant, mais pour des raisons de sécurité, il est conseillé d'en choisir un nouveau.

- Si Unica est configurée pour utiliser une connexion SSL, il vous sera éventuellement demandé lors de votre première connexion d'accepter un certificat de sécurité numérique. Cliquez sur **Oui** pour accepter le certificat.

Si votre connexion aboutit, Unica affiche la page du tableau de bord par défaut.

Avec les autorisations par défaut affectées aux comptes administrateur Unica Platform, vous pouvez administrer les comptes utilisateur et la sécurité en utilisant les options répertoriées dans le menu **Paramètres**. Pour effectuer les tâches d'administration de niveau supérieur d'Unica, vous devez vous connecter avec l'identifiant **platform\_admin**.

# Chapitre 2. Gestion de la sécurité

Une authentification est requise pour accéder à l'interface utilisateur de l'exécution Unica Interact. Seuls les identifiants de connexion définis dans Unica Platform et disposant du rôle d'administration Unica Interact peuvent accéder aux pages.

## Authentifier les pages Unica Interact JSP

Les utilisateurs Platform ou LDAP qui ont un rôle `InteractAdminRole` ou `InteractUserRole` peuvent se connecter à l'interface d'exécution d'Interact. Le nom d'utilisateur et le mot de passe doivent être créés à l'aide de la configuration Unica Platform ou LDAP. Si vous fermez le navigateur ou l'onglet, vous serez déconnecté. Le nom d'utilisateur est désactivé si vous essayez de vous connecter trois fois avec les mauvais identifiants. Si le nom d'utilisateur est désactivé, vous devez l'activer avec l'administrateur Unica Platform. Cette authentification n'est valide que pour les pages JSP.

# Chapitre 3. Configuration des utilisateurs

Dans Unica 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 Unica Platform qui est configuré pour fonctionner avec les serveurs d'exécution.
- **Les utilisateurs de la phase de conception** sont les utilisateurs de Unica Campaign. Configurez la sécurité pour les différents membres de votre équipe de conception comme pour Unica Campaign.

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

Une fois que vous avez installé Unica Interact, vous devez configurer au moins un utilisateur Unica Interact ; l'utilisateur de l'environnement d'exécution. Les utilisateurs de l'environnement d'exécution sont créés dans Unica Platform.

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

Un utilisateur LDAP et tout utilisateur de plateforme peuvent déployer un canal interactif. Le rôle InteractAdminRole n'est pas requis pour déployer le canal interactif.

Lorsque vous créez des utilisateurs d'environnement d'exécution :

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

- 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'environnement d'exécution dans vos propriétés de configuration `SOUS Interact > general > systemTablesDataSource`.
- Si vous activez la sécurité pour la surveillance JMX avec le protocole JMXMP, vous aurez peut-être besoin d'un utilisateur distinct pour la sécurité de la surveillance JMX.

Pour les étapes de création des utilisateurs d'environnement d'exécution, reportez-vous à la documentation Unica Platform.

Interact Runtime prend en charge les rôles et les droits d'accès pour contrôler l'accès utilisateur aux objets et aux fonctionnalités contenus dans Interact Runtime. Ces rôles et droits d'accès peuvent être configurés dans Platform.

Voici les droits d'exécution Interact applicables à la Stratégie globale par défaut et aux nouvelles stratégies.

<b>Catégorie</b>	<b>Droits</b>	<b>Description</b>
Interact	Afficher le statut d'Interact Runtime	Permet de vérifier le statut de l'initialisation, d'afficher la validation de la configuration et la page A propos de.
Interact	Exécuter les API d'Interact Runtime	Exécute les API d'Interact Runtime.
Interact	Afficher les liens Interact Admin	Affiche d'autres pages Admin, comme Balayage JMX, Balayage de configuration du gestionnaire, Contrainte d'offre et Etats de modèle d'événement.

## Configuration des utilisateurs de l'environnement de conception

Les utilisateurs de l'environnement de conception sont les utilisateurs Unica Campaign. Vous configurez les utilisateurs de l'environnement de conception de la même manière que vous configurez les droits d'utilisation dans Unica Campaign.

Certains utilisateurs de l'environnement de conception ont également besoin de certains droits Unica Campaign tels que les macros personnalisées.

Lorsque vous créez des utilisateurs d'environnement de conception, tenez compte des points suivants :

- Si des utilisateurs Unica Campaign sont autorisés à éditer les diagramme temps réel, octroyez-leur l'accès à la source de données des tables d'exécution de test.
- Si vous avez installé et configuré Unica Interact, les options supplémentaires suivantes sont disponibles pour la Stratégie globale et les nouvelles stratégies.

Catégorie	Droits
Campagnes	<ul style="list-style-type: none"> <li>• Afficher les stratégies d'interaction des campagnes : possibilité d'afficher, mais pas de changer, les onglets de stratégie d'interaction dans une campagne.</li> <li>• Editer des stratégies d'interaction de campagnes : possibilité de changer les onglets de stratégie d'interaction, et notamment les règles de traitement.</li> <li>• 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 incluse dans un déploiement de canal interactif.</li> <li>• Ajouter des stratégies d'interaction de campagnes : possibilité de créer des onglets de stratégie d'interaction dans une campagne.</li> </ul>

Catégorie	Droits
	<ul style="list-style-type: none"> <li>• Initiate® 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.</li> </ul>
Canaux interactifs	<ul style="list-style-type: none"> <li>• Déploiement de canaux interactifs : possibilité de déployer un canal interactif dans les environnements d'exécution Unica Interact.</li> <li>• Editer les canaux interactifs : possibilité de changer l'onglet Récapitulatif des canaux interactifs.</li> <li>• Supprimer des canaux interactifs : possibilité de supprimer des canaux interactifs. La suppression des canaux interactifs est limitée si le canal interactif a été déployé.</li> <li>• Afficher des canaux interactifs : possibilité d'afficher, mais pas de modifier les canaux interactifs.</li> <li>• Ajouter des canaux interactifs : possibilité de créer des canaux interactifs.</li> <li>• Afficher des rapports sur les canaux interactifs : possibilité d'afficher l'onglet analyse du canal interactif.</li> <li>• Ajouter des objets enfants aux canaux interactifs : possibilité d'ajouter des points d'interaction, des zones, des événements et des catégories.</li> <li>• Ajouter des points/zones d'interaction : possibilité d'ajouter/d'éditer/de supprimer des points et des zones d'interaction disponibles sous le canal interactif "Points d'interaction".</li> <li>• Ajouter des événements/modèles : possibilité d'ajouter/d'éditer/de supprimer des événements/modèles sous l'onglet "Evénements" du canal interactif.</li> <li>• Ajouter des contraintes d'offre : possibilité d'ajouter/d'éditer/de supprimer des contraintes d'offre disponibles sous l'onglet "Contraintes" du canal interactif.</li> </ul>



Catégorie	Droits
	<ul style="list-style-type: none"> <li>• Ajouter un modèle d'auto-apprentissage : possibilité d'ajouter/d'éditer/de supprimer un modèle d'auto-apprentissage sous l'onglet "Auto-apprentissage" du canal interactif.</li> <li>• Ajouter des messages déclenchés : possibilité d'ajouter/d'éditer/de supprimer des messages déclenchés disponibles sous l'onglet "Messages déclenchés" du canal interactif.</li> <li>• Ajouter des scénarios de simulation : possibilité d'ajouter/d'éditer/de supprimer des scénarios de simulation présents sous l'onglet "Simulator" du canal interactif.</li> <li>• Afficher le mappage d'offres : possibilité d'afficher des mappages d'offres existants.</li> <li>• Editer le mappage d'offres : possibilité d'ajouter/d'éditer/de supprimer des mappages d'offres.</li> <li>• Lancer le déploiement du mappage d'offres : possibilité de marquer le mappage d'offres pour le déploiement.</li> </ul>
Sessions	<ul style="list-style-type: none"> <li>• Afficher des diagrammes interactifs : possibilité d'afficher un diagramme interactif dans une session.</li> <li>• Ajouter des diagrammes interactifs : possibilité de créer des diagrammes interactifs dans une session.</li> <li>• Editer des diagrammes temps réel : possibilité de changer des diagrammes temps réel.</li> <li>• Supprimer des diagrammes interactifs : possibilité de supprimer des diagrammes interactifs. La suppression des diagrammes temps réel est limitée si le canal interactif auquel le diagramme temps réel est affecté a été déployé.</li> <li>• Copier des diagrammes temps réel : possibilité de copier des diagrammes temps réel.</li> </ul>

Catégorie	Droits
	<ul style="list-style-type: none"> <li>• Tester des diagrammes temps réel : possibilité de lancer l'exécution de test pour un diagramme temps réel.</li> <li>• 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.</li> <li>• Déployer des diagrammes interactifs : possibilité de marquer des diagrammes interactifs pour le déploiement ou l'annulation du déploiement.</li> </ul>
Apprentissage global	<ul style="list-style-type: none"> <li>• Afficher des définitions de casier : possibilité d'afficher des définitions de casier disponibles sous Apprentissage global.</li> <li>• Ajouter des définitions de casier : possibilité d'ajouter/d'éditer/de supprimer des définitions de casier disponibles sous Apprentissage global.</li> </ul>
Définitions globales	<ul style="list-style-type: none"> <li>• Afficher des attributs en temps réel : possibilité d'afficher des attributs en temps réel sous Définition globale.</li> <li>• Ajouter des attributs en temps réel : possibilité d'ajouter/d'éditer/de supprimer des attributs en temps réel sous Définition globale.</li> </ul>

## Exemple de permissions dans un environnement de conception

Cet exemple répertorie les droits octroyés à deux rôles différents ; un pour les utilisateurs qui créent des diagrammes temps réel et l'autre pour les utilisateurs qui définissent des stratégies d'interaction.

### Rôle du diagramme temps réel

Cette table illustre les droits octroyés au rôle du diagramme temps réel :

Catégorie	Autorisation
Macro personnalisée	<p>Le rôle utilisateur possède les droits suivants :</p> <ul style="list-style-type: none"> <li>• Ajouter des macros personnalisées</li> <li>• Modifier des macros personnalisées</li> <li>• Utiliser des macros personnalisées</li> </ul>
Zone dérivée	<p>Le rôle utilisateur possède les droits suivants :</p> <ul style="list-style-type: none"> <li>• Ajouter des zones dérivées</li> <li>• Editer des zones dérivées</li> <li>• Utiliser des zones dérivées</li> </ul>
Modèle de diagramme	<p>Le rôle utilisateur possède les droits suivants :</p> <ul style="list-style-type: none"> <li>• Coller des modèles</li> </ul>
Modèle de segment	<p>Le rôle utilisateur possède les droits suivants :</p> <ul style="list-style-type: none"> <li>• Ajouter des segments</li> <li>• Editer des segments</li> </ul>
Session	<p>Le rôle utilisateur possède les droits suivants :</p> <ul style="list-style-type: none"> <li>• Afficher le récapitulatif des sessions</li> <li>• Afficher des diagrammes temps réel</li> <li>• Ajouter des diagrammes temps réel</li> </ul>

Catégorie	Autorisation
	<ul style="list-style-type: none"> <li>• Modifier des diagrammes temps réel</li> <li>• Copier des diagrammes temps réel</li> <li>• Tester des diagrammes temps réel</li> <li>• Déployer des diagrammes temps réel</li> </ul>

## Rôle de stratégie d'interaction

Cette table illustre les droits octroyés au rôle de la stratégie d'interaction :

Catégorie	Autorisation
Campaign	<p>Le rôle utilisateur possède les droits suivants :</p> <ul style="list-style-type: none"> <li>• Afficher récapitulatif de la campagne</li> <li>• Mettre à jour les populations ciblées pour une campagne</li> <li>• Afficher les stratégies d'interaction des campagnes</li> <li>• Modifier des stratégies d'interaction de campagnes</li> <li>• Ajouter des stratégies d'interaction de campagnes</li> <li>• Initiate® des déploiements de stratégie d'interaction de campagne</li> </ul>
Offre	<p>Le rôle utilisateur possède les droits suivants :</p> <ul style="list-style-type: none"> <li>• Afficher le résumé des offres</li> </ul>

<b>Catégorie</b>	<b>Autorisation</b>
Modèle de segment	Le rôle utilisateur possède les droits suivants : <ul style="list-style-type: none"><li>• Afficher le récapitulatif des segments</li></ul>
Session	Le rôle utilisateur possède les droits suivants : <ul style="list-style-type: none"><li>• Réviser des diagrammes temps réel</li></ul>

# Chapitre 4. Gestion des sources de données Unica Interact

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

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

## Sources de données Unica Interact

Unica Interact nécessite plusieurs ensembles de données pour fonctionner. Les ensembles de données sont stockés et extraits de sources de données et les sources de données que vous configurez dépendent des fonctions Unica Interact que vous activez.

- **Unica Campaign tables système.** En plus de toutes les données de Unica Campaign, les tables système Unica Campaign contiennent des données des composants Unica 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 Unica Campaign utilisent la même base de données physique et le même schéma.
- **Tables d'exécution** (`systemTablesDataSource`). Cette source de données contient 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 des profils** (`prodUserDataSource`). Cette source de données contient les données client, au delà des informations rassemblées en temps réel, qui sont requises par les diagrammes interactifs pour placer correctement les visiteurs dans des segments dynamiques. 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 niveau d'audience utilisé par le canal interactif.

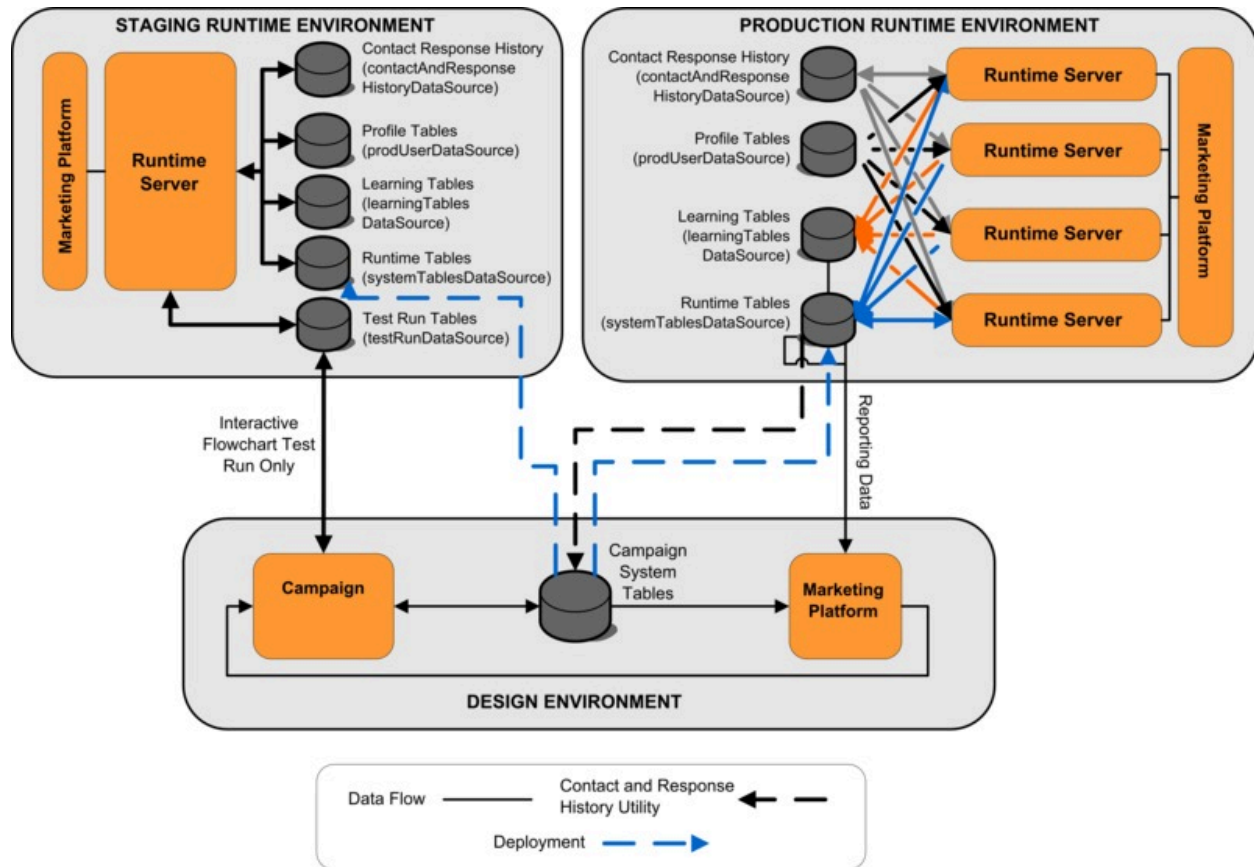
Les tables de profil peuvent également contenir les tables utilisées pour enrichir la proposition d'offres, y compris les tables de suppression d'offre, la substitution de score, et affectation d'offre globale et individuelle.

- **Tables d'exécution de test** (`testRunDataSource`). Cette source de données contient 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`). Cette source de données contient toutes les données recueillies par l'utilitaire d'auto-apprentissage. 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`). Cette source de données contient les tables d'historique de contacts Unica Campaign ou une copie de ces tables. Si vous n'utilisez pas la fonctionnalité de réponse inter-sessions, vous n'avez pas besoin de configurer ces tables d'historique des contacts.

## Bases de données et applications

Les sources de données que vous créez pour utilisation avec Unica Interact peuvent également être utilisées pour échanger ou partager des données avec d'autres applications Unica.

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



- Unica Campaign et le groupe de serveurs 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 temps réel uniquement.
- Lorsque vous utilisez un serveur d'exécution pour tester un déploiement, y compris les API Unica 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 Unica Campaign.
- La fonction de production de rapports interroge les données à partir des tables d'apprentissage, des tables d'exécution et des tables système Unica Campaign pour afficher des rapports dans Unica Campaign.



Vous devez configurer les environnements d'exécution de test à utiliser un autre ensemble de tables que vos environnements 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 Unica Campaign (Unica 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 Unica Campaign

Lorsque vous installez l'environnement de conception Unica Interact, vous pouvez également créer de nouvelles tables spécifiques à Unica Interact dans les tables système Unica Campaign. Les tables que vous créez dépendent des fonctions Unica Interact que vous activez.

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 Unica 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 Unica 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 les offres affectées, vous devrez peut-être remplir la table `UACI_ICBatchOffers` dans les tables système Unica Campaign si vous utilisez des offres non contenues dans les règles de traitement du canal interactif.

## Tables d'exécution

Si plusieurs niveaux d'audience sont définis, vous devez créer des tables de transfert pour les données de l'historique des contacts et des réponses pour chaque niveau d'audience.

Les scripts SQL créent les tables suivantes pour le niveau d'audience par défaut :

- UACI\_CHStaging
- UACI\_CHOfferAttrib
- UACI\_RHStaging

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

Si vos tables d'historique des contacts et de réponse Unica 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 Unica 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 profils, les données temporelles, l'apprentissage, ou l'API Unica Interact.

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

## Runtime Tables (systemTablesDataSource)

UACI_CHStagingAud1
ContactID
TreatmentCode
CampaignID
OfferID
CallID
Aud1_ID
ContactDate
ExpirationDateTime
EffectiveDateTime
ContactType
UserDefinedFields
Mark

UACI_CHStagingAud2
ContactID
TreatmentCode
CampaignID
OfferID
CallID
Aud2_ID
ContactDate
ExpirationDateTime
EffectiveDateTime
ContactType
UserDefinedFields
Mark

UACI_CHOffer AttributesAud1
ContactID
AttributeID
StringValue
NumberValue
DateTimeValue

UACI_CHOffer AttributesAud2
ContactID
AttributeID
StringValue
NumberValue
DateTimeValue

UACI_RHStagingAud1
SeqNum
TreatmentCode
Aud1_ID
ResponseDate
ResponseType
UserDefinedFields
Mark

UACI_RHStagingAud2
SeqNum
TreatmentCode
Aud2_ID
ResponseDate
ResponseType
UserDefinedFields
Mark

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

## Tables d'exécution de test

Les tables d'exécution de test sont utilisées pour tester des diagrammes temps réel uniquement. Les exécutions de test des diagrammes temps réel doivent tester votre logique de segmentation. Il vous suffit de configurer une seule base de données d'exécution de test pour votre installation Unica 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 Unica Campaign.

L'utilisateur de la base de données associée aux tables d'exécution de test doit 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 temps réel. Par exemple, si votre logique de tri des diagrammes temps réel 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 niveau d'audience approprié, et une colonne représentant les données en temps réel que vous prévoyez d'utiliser. Comme les exécutions de test n'ont pas accès aux données en temps réel, vous devez fournir des données d'échantillon 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 d'échantillon. 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 temps réel 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 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 temps réel dans l'environnement d'exécution, vous devez créer un environnement d'exécution de test, y compris une table de profils 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 temps réel, des tables de profil de test pour le groupe de serveurs de test, qui ne doivent pas être le groupe de serveurs de production, mais 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 d'instantané 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 table `UACI_Column_types` n'existe pas dans `testRunDataSource`, ou si `prodUserDataSource` est d'un autre type de base de données, Interact vérifie alors le `prodUserDataSource` de la table.

## Substitution des types de données par défaut

Cette procédure permet de remplacer les types de données par défaut des tables créées dynamiquement.

Vous devez redémarrer le serveur d'exécution chaque fois que vous modifiez la table `UACI_Column_types`. Prévoyez d'apporter vos modifications de sorte que le redémarrage du serveur affecte le moins possible les opérations.

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

Nom de table : `UACI_Column_types`

Noms de colonne :

- `UACI_Float`
- `UACI_Number`
- `UACI_String`

Utilisez le type de données approprié pris en charge par votre base de données pour définir chaque colonne.

2. Redémarrez le serveur d'exécution pour permettre à Interact de reconnaître la nouvelle table.

## Types de données par défaut pour les tables créées dynamiquement

Pour chaque base de données prise en charge, utilisée par le système d'exécution Unica Interact, il existe des types de données codés en dur, utilisées par défaut 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®	<ul style="list-style-type: none"> <li>• float</li> <li>• bigint</li> <li>• timestamp</li> <li>• varchar</li> </ul>
Oracle	<ul style="list-style-type: none"> <li>• float</li> <li>• number(19)</li> <li>• timestamp</li> <li>• varchar2</li> </ul>
SQL Server	<ul style="list-style-type: none"> <li>• float</li> <li>• bigint</li> <li>• datetime</li> <li>• nvarchar</li> </ul>

## 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 temps réel et l'API Unica Interact. Unica 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 temps réel 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 profils dans les zones d'ID de niveau d'audience. 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 audience des tables de profil doivent correspondre aux ID audience définis dans Unica Campaign.

- Si vous définissez la propriété de configuration `enableScoreOverrideLookup` sur `true`, vous devez inclure une table de substitution de score pour au moins un niveau d'audience. 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 échantillon, `UACI_ScoreOverride` en exécutant le script SQL `aci_usrtab` sur votre base de données de profil. Vous devez également indexer cette table sur la colonne ID audience.

Si vous définissez la 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_usrtab` 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 niveau d'audience. Vous pouvez définir les noms de table de suppression d'offre avec la propriété `offerSuppressionTable`.



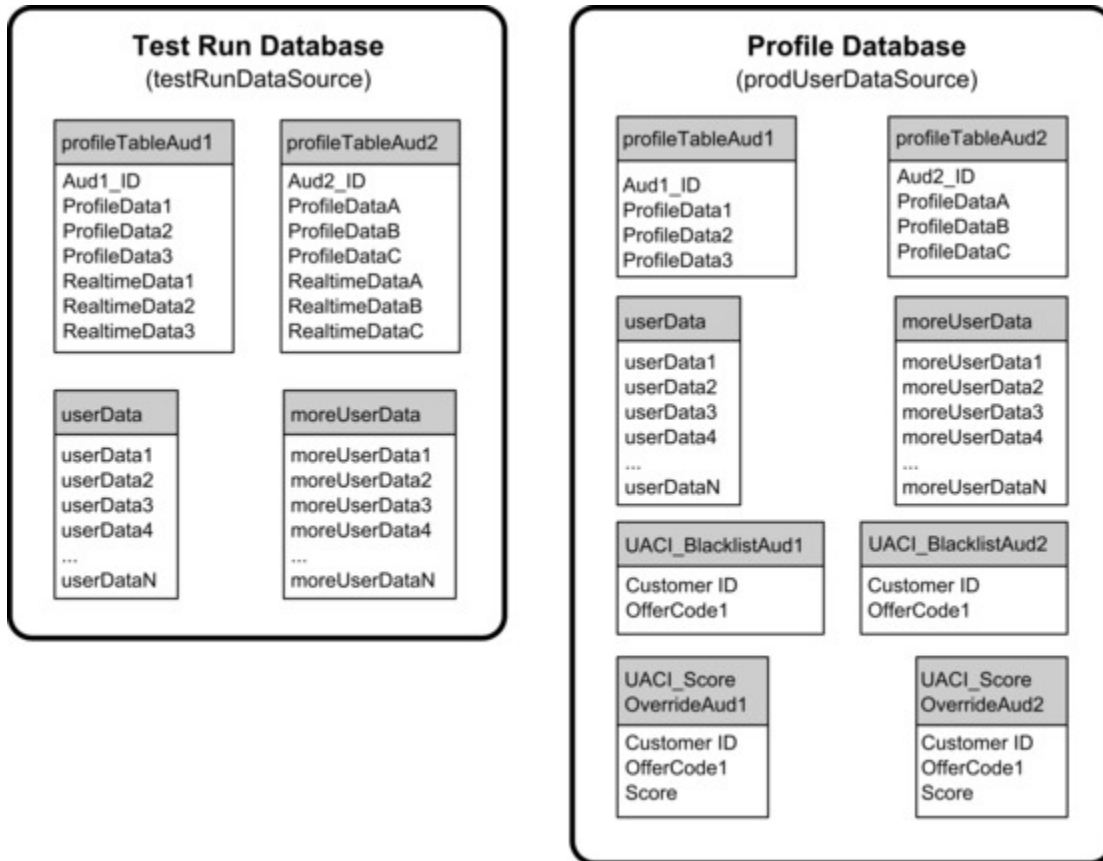
La suppression de la table d'offre peut contenir une ligne pour chaque offre supprimée pour un membre de l'audience, même si une entrée n'est pas requise pour tous les membres. Vous pouvez créer un échantillon de table de suppression d'offre `UACI_BlackList` en exécutant le script SQL `aci_usrtab` sur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

Si vous définissez la 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 des niveaux d'audience 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 > Niveaux d'audience > AudienceLevel**. Le script SQL `aci_usrtab` se trouve dans le répertoire `ddl` du répertoire d'installation de l'environnement d'exécution.

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



## Tables d'apprentissage

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

Si vous utilisez des attributs d'apprentissage dynamique, vous devez remplir 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_OfferTxAll`. 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'auto-apprentissage Unica Interact utilise ces données afin de calculer les scores définitifs des offres.

## Historique des contacts pour le suivi de réponse intersession

Si vous activez la fonctionnalité de réponse inter-sessions, l'environnement d'exécution nécessite un accès en lecture seule aux tables de l'historique des contacts Unica Campaign. Vous pouvez configurer l'environnement d'exécution pour afficher les tables système Unica Campaign, ou vous pouvez créer une copie des tables d'historique des contacts de Unica 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 fonctionnalité de suivi de réponse intersession.

## Exécution des scripts de base de données pour activer les fonctions d'Unica Interact

Pour utiliser les fonctions facultatives qui sont disponibles dans Unica Interact, exécutez des scripts de base de données sur la base de données pour créer des tables ou mettre à jour les tables existantes.

Votre installation Interact, à la fois l'environnement de phase de conception et l'environnement d'exécution, inclut des scripts de fonction ddl. Les scripts ddl ajoutent des colonnes requises à vos tables.

Pour activer l'une des fonctions facultatives, exécutez le script approprié pour la base de données ou la table indiquée.

`dbType` correspond au type de base de données, comme `sqlsvr` pour Microsoft™ SQL Server, `ora` pour Oracle, `db2` pour DB2®, ainsi que MariaDB pour base de données MariaDB.

Utilisez le tableau suivant pour exécuter des scripts de base de données sur la base de données pour créer des tables ou mettre à jour les tables existantes :

**Tableau 2. Scripts de base de données**

Ce tableau à quatre colonnes contient des informations sur les noms de fonction dans la première colonne, des scripts de fonction dans la deuxième colonne, des détails sur l'exécution des scripts de base de données dans la troisième colonne, et les modifications des tables dans la quatrième colonne.

Nom de la fonction	Script de fonction	A exécuter sur	Change
Offres globales, suppression d'offres et substitution de score	aci_usrtab_dbType.sql dans <i>Interact_Home\ddl\acifeatures\</i> (répertoire d'installation dans l'environnement d'exécution)	Votre base de données de profils (user-ProdDataSource)	Crée les tables UACI_DefaultOffers, UACI_BlackList et UACI_ScoreOverride.
Score	aci_scoringfeature_dbType.sql dans <i>Interact_Home\ddl\acifeatures\</i> (répertoire d'installation dans l'environnement d'exécution)	Tables de substitution de score dans votre base de données de profils (user-ProdDataSource)	Ajoute les colonnes LikelihoodScore et AdjExploreScore.
Apprentissage	aci_lrnfeature_dbType.sql dans <i>Interact_</i>	Base de données Unica Campaign qui	Ajoute les colonnes RTSelectionMethod,

## Tableau 2. Scripts de base de données

Ce tableau à quatre colonnes contient des informations sur les noms de fonction dans la première colonne, des scripts de fonction dans la deuxième colonne, des détails sur l'exécution des scripts de base de données dans la troisième colonne, et les modifications des tables dans la quatrième colonne.

(suite)

Nom de la fonction	Script de fonction	A exécuter sur	Change
	<code>Home\interact-DT\ddl\acifeatures\</code> (répertoire d'installation de l'environnement de phase de conception)	contient vos tables d'historique des contacts	RTLearningMode et RTLearningModelID à la table UA_DtlContactHist. Ajoute également les colonnes RTLearningMode et RTLearningModelID à la table UA_ResponseHistory. Ce script est également requis par les fonctions de production de rapports mises à disposition par le package de rapports d'.

## About Cross Session Contact Tracking

Un événement de contact est publié sur Interact après la présentation d'une offre à un utilisateur final. Pour le moment, il est supposé que l'utilisateur final agit dès la présentation de l'offre. Dès lors, l'événement de contact doit être publié dans la même session Interact que la requête d'API `getOffers` qui renvoie l'offre d'origine. Cependant, il existe de nombreux

cas au cours desquels des événements de contact se produisent dans une session différente.

Etant donné que la publication d'un événement de contact nécessite les données de la session et des attributs d'offre personnalisés, les données requises doivent être conservées dans le système après `getOffers`. En raison de la limitation du stockage, les événements de contact intersession peuvent être uniquement gérés au cours d'une période limitée après `getOffers`.

### Comment fonctionne un contact intersession ?

- Lorsqu'un événement de contact est publié sans code de traitement ou code d'offre, tous les traitements renvoyés dans la précédente invocation `getOffers` sont journalisés et sont toujours stockés dans l'objet de la session en cours. Interact recherche le traitement correspondant parmi les traitements disponibles renvoyés dans la précédente invocation `getOffers`.
- Si aucun traitement n'est trouvé dans l'objet de la session et si le suivi du contact intersession est activé, Interact recherche et charge le traitement correspondant dans le magasin de données configuré.
- Si aucun type de contact n'est spécifié ou si le type de contact spécifié est défini en tant qu'événement de contact `true`, cet événement est envoyé au moteur d'apprentissage et les règles de suppression d'offre sont mises à jour, le cas échéant.
- Une erreur est retournée lorsqu'aucun traitement n'est trouvé.

### Modification du schéma et de la base de données

La table suivante est ajoutée à la base de données d'exécution. Elle contient les traitements personnalisés fournis aux clients par le biais d'appels d'API `getOffers` ou d'appels sortants. Cette table est requise pour chaque niveau d'audience.

**Tableau 3. Traitement UACI.**

Colonne	DataType	Description
SeqNum	nombre	ID interne de ce traitement.

**Tableau 3. Traitement UACI. (suite)**

<b>Colonne</b>	<b>Data Type</b>	<b>Description</b>
TreatmentCode	varchar	Le code de traitement de ce traitement personnalisé.
19	nombre	L'ID de l'offre dont provient ce traitement.
OfferCode	varchar	Le code de l'offre dont provient ce traitement. Si le code de l'offre contient plusieurs parties, elles sont toutes concaténées en une chaîne unique séparée par des virgules. Si la longueur totale de ces codes d'offre est trop longue, il peut être nécessaire d'augmenter la taille de cette colonne.
PresentDate	horodatage	L'horodatage du moment où ce traitement personnalisé a été fourni.
IPName	varchar	Le nom du point d'interaction créé pour ce traitement. S'il n'existe aucun point d'interaction associé, comme quelques cas de messages déclenchés, la valeur de ce champ a la valeur Null.
CustomerID	nombre	L'ID d'audience, lorsque le niveau d'audience est

**Tableau 3. Traitement UACI. (suite)**

Colonne	Data Type	Description
		"Client". Si le niveau d'audience n'est pas "Client", cette colonne doit être remplacée par les zones d'ID d'audience correspondantes.
Contacted	nombre	Indique si un événement de contact est publié pour ce traitement. Si ce traitement a la valeur contacted, il renvoie une valeur '1' et s'il n'a pas de valeur contacted, il renvoie une valeur différente.
Détails	varchar	Spécifie les informations associées à ce traitement personnalisé, comme les paramètres de session et les attributs de profil requis pour les tables de transfert CH/RH correspondantes et les attributs d'offre personnalisés.

## 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 Unica 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 Unica Campaign.

Le module de l'historique des contacts et des réponses fonctionne uniquement si vous définissez les propriétés `Campaign > partitions > partition1 > InteractInteractInstalled` et `contactAndResponseHistTracking > isEnabled` de la page Configuration de l'environnement de conception sur `yes`.

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

## Types de contact et de réponse

Vous pouvez enregistrer un type de contact et deux types de réponses avec Unica Interact. Vous pouvez également enregistrer davantage de types de réponse personnalisés à l'aide de la méthode `postEvent`.

### Propriétés de la table `contactAndResponseHistTracking`

Cette table répertorie les propriétés qui se trouvent dans la catégorie `contactAndResponseHistTracking` :

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

## Propriétés de la table UA\_UsrResponseType

Vérifiez que la colonne `CountsAsResponse` de la table `UA_UsrResponseType` dans les tables système Unica Campaign est correctement configurée. Tous ces types de réponse doivent exister dans la table `UA_UsrResponseType`.

Pour que l'entrée soit valide dans la table `UA_UsrResponseType`, vous devez définir une valeur pour toutes les colonnes de la table, y compris `CountsAsResponse`. Les valeurs valides pour `CountsAsResponse` sont :

- 0 - pas de réponse
- 1 - une réponse
- 2 - un rejet
- 

Ces réponses sont utilisées pour la génération de rapports.

## Autres types de contacts

Dans Interact, vous pouvez utiliser la méthode `postEvent` dans l'API Interact afin de déclencher un événement de contact. Vous pouvez également augmenter le système afin de permettre à l'appel `postEvent` d'enregistrer des types de contacts personnalisés supplémentaires. Tous ces types de contacts doivent exister dans la table `UA_ContactStatus` des tables système Campaign. L'utilisation de paramètres d'événements spécifiques à la méthode `postEvent` vous permet d'enregistrer des types de contacts supplémentaires et de définir s'il s'agit d'un vrai contact ou non. Pour journaliser des types de contacts supplémentaires, vous devez ajouter les paramètres d'événement suivants :

`UACIContactStatusCode` - une chaîne représentant un code de type de contact. La valeur doit être une entrée valide de la table `UA_ContactStatus`. Pour que l'entrée dans `UA_ContactStatus` soit valide, vous devez définir toutes les colonnes de la table, y compris `CountsAsContact`. Les valeurs admises pour `CountsAsContact` sont 0 et 1. 0 représente un contact qui n'a pas abouti et 1 indique un contact qui a abouti.

## Autres types de réponse

Dans Unica Interact, vous pouvez utiliser la méthode `postEvent` dans l'API Unica 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 Unica Campaign. L'utilisation de paramètres d'événement spécifiques à la méthode `postEvent` vous permet d'enregistrer des types de réponse supplémentaires et définir si une acceptation doit être incluse dans l'apprentissage. Il est également suggéré de ne pas publier des réponses multiples (Accepter/Refuser) pour un contact unique, cela pouvant fausser les résultats d'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** -

Le paramètre "UACILogToLearning" est obsolète dans la version 11.0. A la place, les valeurs réelles définies dans les tables "UA\_ContactStatus" et "UA\_UsrResponseType" de la base de données de Campaign associées aux valeurs définies dans les paramètres "Affinium|interact|services|contactHist|contactStatusCodes" et "Affinium|interact|services|responseHist|responseTypeCodes" seront prises en compte par le système Interact.

Si vous sélectionnez "UACILogToLearning= 1" dans un appel `postEvent`, l'action par défaut associée au type de réponse/statut du contact sera ignorée et cet événement sera toujours traité comme véritable réponse/contact.

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)] : Explorer (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. Appelez-le `JournalRéponsePersonnalisée`, par exemple.

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 Unica Campaign

Les tables de transfert de l'historique des contacts Unica Interact sont mappées aux tables d'historique Unica Campaign. Vous devez disposer de l'une des tables de transfert de l'environnement d'exécution pour chaque niveau d'audience.

### Mappage des tables de transfert de l'historique des contacts

#### UACI\_CHStaging

Cette table illustre le mappage des tables de transfert de l'environnement d'exécution `UACI_CHStaging` à la table d'historique des contacts Unica Campaign. Les noms de table affichés sont les échantillons de tables créés pour le référentiel par défaut dans les tables d'exécution et les tables système Unica Campaign.

**Tableau 4. Historique des contacts**

<b>UACI_CHStaging</b> <b>Nom de colonne dans la table de transfert de l'historique des contacts Unica Interact</b>	<b>Table de l'historique des contacts Unica Campaign</b>	<b>Nom de colonne de la table</b>
ContactID	S/O	S/O
TreatmentCode	UA_Treatment	TreatmentCode
CampaignID	UA_Treatment	CampaignID
19	UA_Treatment	19
CellID	UA_Treatment	CellID
CustomerID	UA_DtlContactHist	CustomerID
ContactDate	UA_DtlContactHist	ContactDateTime

**Tableau 4. Historique des contacts (suite)**

<b>UACI_CHStaging</b> <b>Nom de colonne dans la table de transfert de l'historique des contacts Unica Interact</b>	<b>Table de l'historique des contacts Unica Campaign</b>	<b>Nom de colonne de la table</b>
ExpirationDateTime	UA_Treatment	ExpirationDateTime
EffectiveDateTime	UA_Treatment	EffectiveDateTime
ContactType	UA_DtlContactHist	ContactStatusID
ContactStatusCode	UA_DtlContactHist	ContactStatusId
UserDefinedFields	UA_DtlContactHist	UserDefinedFields

ContactID est une clé qui permet de joindre la table UACI\_CHOfferAttrib avec la table UACI\_CHStaging. La colonne userDefinedFields peut contenir les données de votre choix.

## **Mappage des tables de transfert de l'historique des contacts UACI\_CHOfferAttrib**

Cette table illustre le mappage des tables de transfert de l'environnement d'exécution UACI\_CHOfferAttrib à la table d'historique des contacts Unica Campaign. Les noms de table affichés sont les échantillons de tables créés pour le référentiel par défaut dans les tables d'exécution et les tables système Unica Campaign.

**Tableau 5. Attributs d'offre**

<b>UACI_CHOfferAttrib</b> <b>Nom de colonne dans la table de transfert de l'historique des contacts Unica Interact</b>	<b>Table de l'historique des contacts Unica Campaign</b>	<b>Nom de colonne de la table</b>
ContactID	S/O	S/O
AttributeID	UA_OfferHistAttrib	AttributeID



**Tableau 5. Attributs d'offre (suite)**

<b>UACI_CHOfferAttrib</b> <b>Nom de colonne dans la table de transfert de l'historique des contacts Unica Interact</b>	<b>Table de l'historique des contacts Unica Campaign</b>	<b>Nom de colonne de la table</b>
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

## **Mappage des tables de transfert de l'historique des réponses**

### **UACI\_RHStaging**

Cette table illustre le mappage des tables de transfert de l'environnement d'exécution `UACI_RHStaging` à la table d'historique des réponses Unica Campaign. Les noms de table affichés sont les échantillons de tables créés pour le référentiel par défaut dans les tables d'exécution et les tables système Unica Campaign.

**Tableau 6. Historique des réponses**

<b>UACI_RHStaging</b> <b>Nom de colonne dans la table de transfert de l'historique des réponses Unica Interact</b>	<b>Table d'historique des réponses Unica Campaign</b>	<b>Nom de colonne de la table</b>
SeqNum	S/O	S/O
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime

**Tableau 6. Historique des réponses (suite)**

<b>UACI_RHStaging</b> <b>Nom de colonne dans la table de transfert de l'historique des réponses Unica Interact</b>	<b>Table d'historique des réponses Unica Campaign</b>	<b>Nom de colonne de la table</b>
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 Unica Campaign. La colonne `userDefinedFields` peut contenir les données de votre choix.

### Colonnes supplémentaires dans les tables de transfert

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 copie ces données dans la colonne `linkFrom` de la table `UA_DtlContactHist`.

### Colonnes supplémentaires dans les tables d'historique des contacts et des réponses Unica Campaign

Si vous avez des colonnes supplémentaires dans vos tables d'historique des contacts et des réponses Unica Campaign, ajoutez des colonnes correspondantes aux tables de transfert avant d'exécuter le module de l'historique des contacts et des réponses.

Vous remplissez 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, vous créez des paires nom/valeur `NumberItemsInWishList` et `NumberItemsInShoppingCart` et les ajoutez à votre table `UACI_RHStaging`. Lorsqu'un événement Journaliser l'acceptation de l'offre ou Journaliser le refus de l'offre se produit, 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.

## Utilisation de tables afin d'inclure un score pour une offre

Vous pouvez utiliser les 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 Unica Campaign. Unica 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 Unica Campaign. Unica Interact complète automatiquement la colonne `Score` en y insérant le score utilisé pour l'offre.

## Création de tables d'historique dans Unica Campaign et de tables de transfert dans Unica Interact

Si vous utilisez un niveau d'audience autre que `Customer` (Client), vous devez créer des tables d'historique dans Unica Campaign et des tables de transfert dans Unica Interact.

Ainsi, l'exemple de script ci-dessous est utilisé dans la base de données IBM DB2® de la phase de conception afin de créer des tables d'historique dans Unica Campaign pour un niveau d'audience de type `Account` (Compte).

```
DROP TABLE ACCT_UA_ResponseHistory;
DROP TABLE ACCT_UA_DtlContactHist;
DROP TABLE ACCT_UA_ContactHistory;
CREATE TABLE ACCT_UA_ResponseHistory (
```

```

AccountID          varchar(30) NOT NULL,
TreatmentInstID    bigint NOT NULL,
ResponsePackID     bigint NOT NULL,
ResponseDateTime   timestamp NOT NULL,
WithinDateRangeFlg int,
OrigContactedFlg  int,
BestAttrib         int,
FractionalAttrib   float,
DirectResponse     int,
CustomAttrib       float,
ResponseTypeID     bigint,
DateID             bigint,
TimeID             bigint,
UserDefinedFields  char(18),
CONSTRAINT ACCT_cRespHistory_PK
    PRIMARY KEY (AccountID, TreatmentInstID,
                ResponsePackID )
);

CREATE TABLE ACCT_UA_ContactHistory (
    AccountID          varchar(30) NOT NULL,
    CellID             bigint NOT NULL,
    PackageID          bigint NOT NULL,
    ContactDateTime    timestamp,
    UpdateDateTime     timestamp,
    ContactStatusID    bigint,
    DateID             bigint,
    TimeID             bigint,
    UserDefinedFields  char(18),
    CONSTRAINT ACCT_cContactHist_PK
        PRIMARY KEY (AccountID, CellID, PackageID )
);

CREATE INDEX ACCT_cContactHist_IX1 ON ACCT_UA_ContactHistory

```

```

(
    CellID
);
CREATE INDEX ACCT_cContactHist_IX2 ON ACCT_UA_ContactHistory
(
    PackageID
    ,
    CellID
);
CREATE TABLE ACCT_UA_DtlContactHist (
    AccountID          varchar(30) NOT NULL,
    TreatmentInstID    bigint NOT NULL,
    ContactStatusID    bigint,
    ContactDateTime     timestamp,
    UpdateDateTime     timestamp,
    UserDefinedFields  char(18),
    DateID             bigint NOT NULL,
    TimeID             bigint NOT NULL
);
CREATE INDEX ACCT_cDtlContHist_IX1 ON ACCT_UA_DtlContactHist
(
    AccountID
    ,
    TreatmentInstID
);
ALTER TABLE ACCT_UA_ResponseHistory
    ADD CONSTRAINT ACCT_cRespHistory_FK2
        FOREIGN KEY (TimeID)
            REFERENCES UA_Time (TimeID);
ALTER TABLE ACCT_UA_ResponseHistory
    ADD CONSTRAINT ACCT_cRespHistory_FK4
        FOREIGN KEY (DateID)
            REFERENCES UA_Calendar (DateID);
ALTER TABLE ACCT_UA_ResponseHistory

```

```

ADD CONSTRAINT ACCT_cRespHistory_FK3
    FOREIGN KEY (ResponseTypeID)
        REFERENCES UA_UsrResponseType (
            ResponseTypeID);

ALTER TABLE ACCT_UA_ResponseHistory
    ADD CONSTRAINT ACCT_cRespHistory_FK1
        FOREIGN KEY (TreatmentInstID)
            REFERENCES UA_Treatment (
                TreatmentInstID);

ALTER TABLE ACCT_UA_ContactHistory
    ADD CONSTRAINT ACCT_cContactHist_FK2
        FOREIGN KEY (DateID)
            REFERENCES UA_Calendar (DateID);

ALTER TABLE ACCT_UA_ContactHistory
    ADD CONSTRAINT ACCT_cContactHist_FK3
        FOREIGN KEY (TimeID)
            REFERENCES UA_Time (TimeID);

ALTER TABLE ACCT_UA_ContactHistory
    ADD CONSTRAINT ACCT_cContactHist_FK1
        FOREIGN KEY (ContactStatusID)
            REFERENCES UA_ContactStatus (
                ContactStatusID);

ALTER TABLE ACCT_UA_DtlContactHist
    ADD CONSTRAINT ACCT_cDtlContactH_FK3
        FOREIGN KEY (TimeID)
            REFERENCES UA_Time (TimeID);

ALTER TABLE ACCT_UA_DtlContactHist
    ADD CONSTRAINT ACCT_cDtlContactH_FK2
        FOREIGN KEY (DateID)
            REFERENCES UA_Calendar (DateID);

ALTER TABLE ACCT_UA_DtlContactHist
    ADD CONSTRAINT ACCT_cDtlContactH_FK1

```

```

        FOREIGN KEY (ContactStatusID)
                REFERENCES UA_ContactStatus (
        ContactStatusID);
alter table ACCT_UA_DtlContactHist add RTSelectionMethod int;
alter table ACCT_UA_ResponseHistory add RTSelectionMethod int;

```

L'exemple de script ci-dessous est utilisé dans la base de données IBM DB2® de la phase d'exécution afin de créer des tables de transfert de l'historique dans Unica Interact pour un niveau d'audience de type `Compte`.

```

DROP TABLE ACCT_UACI_RHStaging;
DROP TABLE ACCT_UACI_CHOfferAttrib;
DROP TABLE ACCT_UACI_CHStaging;
DROP TABLE ACCT_UACI_UserEventActivities;
DROP TABLE ACCT_UACI_EventPatternState;
CREATE TABLE ACCT_UACI_RHStaging (
        SeqNum                bigint NOT NULL,
        TreatmentCode         varchar(512),
        AccountID             varchar(30),
        ResponseDate          timestamp,
        ResponseType          int,
        ResponseTypeCode      varchar(64),
        Mark                   bigint NOT NULL
                                DEFAULT 0,
        UserDefinedFields     char(18),
        RTSelectionMethod     int,
        CONSTRAINT iRHStaging_PK1
                PRIMARY KEY (SeqNum)
);
CREATE TABLE ACCT_UACI_CHOfferAttrib (
        ContactID             bigint NOT NULL,
        AttributeID           bigint NOT NULL,
        StringValue           varchar(512),

```

```

        NumberValue          float,
        DateTimeValue        timestamp,
        CONSTRAINT ACCT_iCHOfferAttrib_PK
                PRIMARY KEY (ContactID, AttributeID)
);

CREATE TABLE ACCT_UACI_CHStaging (
        ContactID            bigint NOT NULL,
        TreatmentCode        varchar(512),
        CampaignID           bigint,
        OfferID              bigint,
        CellID               bigint,
        AccountID            varchar(30),
        ContactDate          timestamp,
        ExpirationDateTime   timestamp,
        EffectiveDateTime    timestamp,
        ContactType          int,
        UserDefinedFields    char(18),
        Mark                 bigint NOT NULL DEFAULT 0,
        RTSelectionMethod    bigint,
        CONSTRAINT ACCT_iCHStaging_PK
                PRIMARY KEY (ContactID)
);

CREATE TABLE ACCT_UACI_UserEventActivity
(
        SeqNum               bigint NOT NULL GENERATED ALWAYS AS IDENTITY,
        ICID                 bigint NOT NULL,
        ICName               varchar(64) NOT NULL,
        CategoryID          bigint NOT NULL,
        CategoryName        varchar(64) NOT NULL,
        EventID             bigint NOT NULL,
        EventName           varchar(64) NOT NULL,
        TimeID              bigint,

```



```

        DateID                bigint,
        Occurrences           bigint NOT NULL,
        AccountID varchar(30) not null,
        CONSTRAINT iUserEventActivity_PK
                PRIMARY KEY (SeqNum)
);
create table ACCT_UACI_EventPatternState
(
    UpdateTime bigint not null,
    State varchar(1000) for bit data,
    AccountID varchar(30) not null,
        CONSTRAINT iCustomerPatternState_PK
                PRIMARY KEY (AccountID,UpdateTime)
);
ALTER TABLE ACCT_UACI_CHOfferAttrib
        ADD CONSTRAINT ACCT_iCHOfferAttrib_FK1
                FOREIGN KEY (ContactID)
                        REFERENCES ACCT_UACI_CHStaging (ContactID);

```

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

Cette procédure permet de configurer la surveillance JMX pour le module de l'historique des contacts et des réponses. Les protocoles JMXMP et RMI sont pris en charge. La configuration de la surveillance JMX n'active pas la sécurité pour le module de l'historique des contacts et des réponses. Utilisez Unica Platform pour que l'environnement de conception configure la surveillance JMX.

Pour utiliser votre outil de surveillance JMX pour le module de l'historique des contacts et des réponses, l'adresse par défaut utilisée pour :

- Le protocole JMXMP est `service:jmx:jmxmp://CampaignServer:port/campaign`.
- Le protocole RMI est `service:jmx:rmi:///jndi/rmi://CampaignServer:port/campaign`.

Lorsque vous affichez les données dans votre outil de surveillance JMX, les attributs des résultats sont organisés d'abord par la partition puis par niveau d'audience.

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

Propriété de configuration	Paramètre
<code>monitorEnabledForInteract</code>	<b>True</b>
<code>port</code>	Numéro de port du service JMX
<code>protocol</code>	Protocole à utiliser : <ul style="list-style-type: none"> <li>• <b>JMXMP</b></li> <li>• <b>RMI</b></li> </ul> La sécurité n'est pas activée pour le module d'historique des réponses et des contacts, même si vous sélectionnez le protocole JMXMP.

## A propos du suivi de réponse intersession

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 intersession 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 Unica 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 Unica Campaign. La réussite du traitement de la réponse intersession dépend des enregistrements d'historique des contacts d'origine qui ont été migrés dans la base de données de Unica Campaign par le processus ETL de l'historique des contacts.

## Activer la détection et la suppression des doublons

Pour activer la vérification des doublons des réponses et des réponses entre sessions, ajoutez le paramètre JVM suivant au runtime Interact.

```
-Dcom.unicacorp.interact.rhDupeCheckLimit=<max records>
```

Ici, `<max records>` est le nombre maximum d'enregistrements uniques à conserver pour vérifier les doublons. Cette vérification est désactivée, si sa valeur est 0, qui est la valeur par défaut.

En outre, le paramètre JVM suivant peut également être ajouté à l'environnement d'exécution d'Interact. Si sa valeur est `true`, les réponses dupliquées et les réponses intersessions sont supprimées. Par défaut, c'est désactivé.

```
-Dcom.unicacorp.interact.rhSuppressDupe=<true|false>
```

Une fois la vérification des doublons activée, un message d'avertissement est consigné dans `interact.log` avec les informations des doublons. Cette vérification se produit aux deux étapes suivantes dans le code.

- Après qu'Interact a traité l'événement de réponse, mais avant qu'il ne soit ajouté au cache mémoire.
- Lorsqu'Interact est sur le point de conserver l'événement de réponse dans la table intermédiaire.

De plus, une nouvelle propriété `CacheInfo` est ajoutée au bean JMX `com.unicacorp.interact:type=Services,group=Response History Memory Cache Statistics`. Lorsque la vérification des doublons est activée, elle renvoie des informations sur le cache mémoire au format suivant.

```
{<cache ID>=<earliest response timestamp--><latest response timestamp>:
<number of records in this cache> - {<audience ID>, <treatment code>,
<response timestamp>=<number of occurrences>}}
```

Par exemple :

```
{2043682026=20201113102136->20201113102136: 10 - {Customer ([1.0]),
7.a.ffffffff9aae0b53.4d37d0d3, 2020-11-13 10:21:36.377=10}}
```

Seules les entrées en double sont incluses dans le message.

Cela revient à activer la vérification des contacts en double, ajoutez le paramètre JVM suivant à l'exécution Interact. Les paramètres JVM suivants sont requis.

- `-Dcom.unicacorp.interact.chDupeCheckLimit=<max records>`
- `-Dcom.unicacorp.interact.chSuppressDupe=<true|false>`

Le bean JMX concerné est `com.unicacorp.interact:type=Services,group=Contact History Memory Cache Statistics`

## Cross-session response process

Le processus de réponse intersession démarre par l'initialisation d'Interact. Il traite les enregistrements dans une table de transfert de réponse intersession. Le processus continue d'interroger la table afin de traiter tous les nouveaux enregistrements ou les nouvelles tentatives d'enregistrement après un intervalle de temps configurable.

`Affinium|interact|services|crossSessionResponse|xsessionProcessIntervalInSecs.`

Les enregistrements réussis sont purgés de la table. Les enregistrements échoués sont marqués afin d'effectuer une nouvelle tentative à une heure définie. `Affinium|interact|`

`services|crossSessionResponse|purgeOrphanResponseThresholdInMinutes.` Les

enregistrements ne sont pas traités avec succès après plusieurs tentatives. `Affinium|`

`interact|services|crossSessionResponse|purgeOrphanResponseThresholdInMinutes.`

Quelques minutes après le temps de réponse de l'enregistrement, ces enregistrements sont indiqués comme étant "En échec".

Pour prendre en charge le fonctionnement de plusieurs instances d'exécution sur les mêmes données, chaque processus de réponse intersession met à jour un nombre d'enregistrements `xsessionResponseBatchSize` en "En cours" avec une valeur unique pour une colonne marquée connue de ce seul processus. Le processus essaie de faire correspondre ces enregistrements avec les données d'historique de contact disponibles à l'aide des SQL définis par le système, pour mettre en correspondance `byTreatmentCode` et `byOfferCode`. En ce qui concerne les enregistrements pour lesquels une correspondance est trouvée, le système déclenche une réponse de journal et la colonne marquée est alors mise à jour en "Réussite".



**Remarque :** La mise à jour de la colonne marquée à l'aide d'une valeur unique spécifique au processus n'est applicable que lorsque l'utilisateur n'a configuré aucune propriété "Effacer SQL". Lorsque l'option "Effacer SQL" sert à définir la requête correspondante, le processus de réponse intersession prend le verrouillage de base de données exclusif sur la table de transfert de réponse intersession, traite les enregistrements et annule le verrouillage.

S'il existe un grand nombre d'enregistrements non traités dans la table `xSessionResponse`, des problèmes de performance peuvent survenir lorsque le service `CrossSessionResponse` tente de traiter tous les enregistrements en même



temps. Pour améliorer les performances, les utilisateurs peuvent définir `Affinium|interact|services|crossSessionResponse|xsessionResponseBatchSize` sur une valeur de nombre entier positif. Le service `CrossSessionResponse` traite les enregistrements `xsessionResponseBatchSize` à la fois et réalise une boucle à travers la table `xSessionResponse` jusqu'à ce que tous les nouveaux enregistrements ou les nouvelles tentatives d'enregistrement soient traités.

## Configuration de la source de données du suivi de réponse intersession

Le suivi de réponse intersession Unica Interact fait correspondre les données de session de l'environnement d'exécution avec l'historique des contacts et des réponses Unica Campaign. Par défaut, le suivi de réponse intersession 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 Unica Interact.
- L'environnement d'exécution doit avoir accès aux tables de l'historique des contacts Unica Campaign. Vous pouvez configurer l'environnement d'exécution afin qu'il accède aux tables de l'historique des contacts Unica 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 de réponse intersession

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

Les tables d'historique des contacts et des réponses doivent être mappées correctement dans Unica Campaign avant d'effectuer ces étapes.

1. Exécutez le script SQL `aci_lrnfeature` dans le répertoire `interactDT/ddl/acifeatures` dans le répertoire d'installation de l'environnement de conception Unica Interact sur les tables `UA_DtlContactHist` et `UA_ResponseHistory` dans vos tables système Unica 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 niveaux d'audience. Editez le script si nécessaire pour exploiter la table correcte pour chacun de vos niveaux d'audience.

2. Si vous souhaitez copier les tables de l'historique des contacts dans l'environnement d'exécution, faites-le maintenant.

Si vous créez une copie des tables de l'historique des contacts Unica Campaign de accessibles à l'environnement d'exécution pour le suivi de réponse intersession, appliquez les directives suivantes :

- Le suivi de réponse intersession nécessite un accès en lecture seule à ces tables.
- Le suivi de réponse inter-sessions nécessite les tables suivantes dans l'historique des contacts Unica Campaign.

- `UA_DtlContactHist` (pour chaque niveau d'audience)
- `UA_Treatment`

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

3. Exécutez le script SQL `aci_crhtab` dans le répertoire `ddl` dans le répertoire d'installation de l'environnement de conception Unica 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`.

4. Créez une version de la table `UACI_XsessResponse` pour chaque niveau d'audience.

Pour améliorer les performances du suivi de réponse intersession, 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 Unica 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. Pour modifier le nombre de jours de conservation des données d'historique des contacts, ouvrez la propriété de configuration **Campaign | partitions | partitionn| Interact | contactAndResponseHistTracking** et définissez la valeur **daysBackInHistoryToLookupContact**.

Vous ne verrez pas les résultats du suivi de réponse intersession 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 Unica 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 de réponse intersession. 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.
Nom	varchar(64)	Nom du type de code de suivi. Il est transmis aux données de session à l'aide du paramètre réservé <code>UACI_TrackingCodeType</code> avec la méthode <code>post-Event</code> .
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 d'offre	Code d'offre de campagne UAC

## UACI\_XSessResponse

La table `UACI_XSessResponse` fait partie des tables de l'environnement d'exécution. Cette table est utilisée pour le suivi de réponse intersession.

Une instance de cette table pour chaque niveau d'audience doit exister dans la source de données de l'historique des contacts et des réponses disponibles pour le suivi de réponse intersession Unica Interact.

Colonne	Type	Description
<i>SeqNumber</i>	bigint	Identificateur de la ligne des données. Le service <code>CrossSessionResponse</code> traite tous les enregistrements dans l'ordre <i>SeqNumber</i> .
<i>ICID</i>	bigint	ID de canal interactif
<i>AudienceID</i>	bigint	ID de cette audience. Le nom de cette colonne doit correspondre à l>ID audience défini dans Unica Campaign. L'échantillon de table contient la colonne <i>CustomerID</i> .
<i>TrackingCode</i>	varchar(64)	Valeur transmise par le paramètre <code>UACIOfferTrackingCode</code> de la méthode <code>postEvent</code> .
<i>TrackingCodeType</i>	int	Représentation numérique d code de suivi. La valeur doit être une entrée valide de la table <code>UACI_TrackingType</code> .
<i>OfferID</i>	bigint	Id de l'offre défini dans Unica Campaign.
<i>Type de réponse</i>	int	Type de réponse de cet enregistrement. La valeur doit être une entrée valide de la table <code>UA_UsrResponseType</code> .
<i>ResponseTypeCode</i>	varchar(64)	Code du type de réponse de cet enregistrement. La valeur doit être une entrée valide de la table <code>UA_UsrResponseType</code> .
<i>ResponseDate</i>	date/heure	Date de la réponse.
<i>Mark</i>	bigint	La valeur de cette zone identifie l'état de l'enregistrement. <ul style="list-style-type: none"> <li>• 1 - En cours. Applicable lorsque la configuration "Effacer SQL" est utilisée. Dans ce cas, la requête correspondante définie par l'utili-</li> </ul>

Colonne	Type	Description
		<p>sateur est utilisée à la place de l'instruction SQL générée par le système.</p> <ul style="list-style-type: none"> <li>• Valeur Entier élevé aléatoire – Applicable lorsque l'instruction "Utiliser SQL généré par le système" est définie ou que la requête correspondante par défaut est utilisée. Le système met à jour les enregistrements à traiter à l'aide d'une valeur d'entier élevé unique afin d'identifier les enregistrements à traiter par cette unité d'exécution.</li> <li>• 2 - Réussi. Applicable lorsqu'une correspondance de l'historique de contact est trouvée et que la réponse du journal est exécutée avec succès.</li> <li>• NULL - Nouveaux enregistrements</li> <li>• 0 - Réessayez. Applicable lorsqu'aucune correspondance de l'historique de contact est trouvée et que l'enregistrement se trouve dans la base de données pendant moins de <code>purgeOrphanResponseThresholdInMinutes</code> minutes.</li> <li>• -1 - L'enregistrement est dans la base de données depuis plus de <code>purgeOrphanResponseThresholdInMinutes</code> minutes.</li> </ul> <p>Dans le cadre de la maintenance de l'administrateur de base de données 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</p>

Colonne	Type	Description
		sont automatiquement supprimés par le service CrossSessionResponse.
UsrDefinedFields	char(18)	Toutes les zones de personnalisation à inclure lors de la mise en 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 de réponse intersession

Cette procédure permet d'activer le suivi de réponse intersession.

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

Pour utiliser le suivi de réponse intersession, 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 Unica Campaign. Vous pouvez effectuer une lecture depuis les tables de l'historique des contacts et des réponses Unica Campaign dans l'environnement de conception, ou à partir d'une copie des tables dans les sources de données de l'environnement d'exécution. La configuration de l'environnement d'exécution pour obtenir l'accès en lecture aux tables de l'historique des contacts et des réponses est distincte 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 niveau d'audience.
4. Créez une catégorie `OverridePerAudience` pour chaque niveau d'audience.

## Mise en correspondance d'offre de réponse intersession

Par défaut, le suivi de réponse intersession 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 Unica 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 niveau d'audience, 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 échantillon de commande SQL générée par défaut qui établit une correspondance avec les codes de traitement. Unica 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 SQL généré par le système**.

```
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`, and `UA_ContactHistory` sont définies par vos paramètres dans Unica Interact. Par exemple, `UACI_XsessResponse` est défini par la propriété de configuration `Interact > profile > 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 niveau d'audience, 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 échantillon de commande SQL générée par défaut qui établit une correspondance avec les codes d'offre. Unica 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 SQL généré par le système**.

```
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`, and `UA_ContactHistory` sont définies par vos paramètres dans Unica Interact. Par exemple, `UACI_XsessResponse` est défini par la propriété de configuration `Interact > profile > 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 Unica 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 niveau d'audience, plus les colonnes `TreatmentCodeMatch` et `OfferIDMatch`. Vous pouvez éventuellement 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.

<b>Current® Etat</b>	<b>Description</b>
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.
TRAITE	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 devez 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 <code>ATTN</code> ou <code>ERROR</code> pour exécuter l'opération <code>RERUN</code> 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 échantillon 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

Cette procédure permet d'activer 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.

Unica Interact fournit des exemples de commande et des fichiers de contrôle dans le répertoire `loaderService` dans votre installation du serveur d'exécution Unica 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 `Interact > general > systemTablesDataSource` dans vos propriétés de configuration.
2. Définissez les propriétés de configuration `Interact > general > systemTablesDataSource > loaderProperties`.
3. Définissez la propriété `Interact > services > externalLoaderStagingDirectory`.
4. Réviser les propriétés de configuration `Interact > services > responseHist > fileCache` si nécessaire.
5. Réviser les propriétés de configuration `Interact > services > contactHist > fileCache` si nécessaire.
6. Redémarrez le serveur d'exécution.

## Processus ETL de modèle d'événement

Pour traiter d'importants volumes de données de modèle d'événement Unica Interact et pour les rendre disponibles en vue de l'établissement de requêtes et de génération de rapports, vous pouvez installer un processus ETL (Extract, Transform, Load) sur tous les serveurs pris en charge afin d'optimiser les performances.

Dans Interact, toutes les données de modèle d'événement d'un ID audience sont stockées sous forme de collection unique dans les tables de la base de données d'exécution. L'ID audience et les informations d'état du modèle sont stockées sous forme d'objet BLOB.

Pour que vous puissiez effectuer des requêtes SQL ou de la génération de rapports en fonction des modèles d'événement, ce nouveau processus ETL est nécessaire pour décomposer l'objet en tables dans une base de données cible. Pour ce faire, le processus ETL autonome prend les données de modèle d'événement dans les tables de la base de données d'exécution Unica Interact, les traite selon la planification que vous avez indiquée, puis les stocke dans la base de données cible où elles sont disponibles pour les requêtes SQL ou une génération de rapports supplémentaires.

Outre le transfert et la transformation des données de modèle d'événement dans la base de données cible, le processus ETL base de données autonome synchronise également les données dans la base de données cible avec les informations les plus récentes provenant de la base de données d'exécution d'Unica Interact. Par exemple, si vous supprimez un modèle d'événement dans la base de données d'exécution d'Unica Interact, les données traitées de ce modèle d'événement sont supprimées de la base de données cible lors de l'exécution suivante du processus ETL. Les informations d'état du modèle d'événement sont également mises à jour. Ainsi, les informations stockées concernant les modèles d'événement dans la base de données cible sont uniquement les données en cours et non des informations d'historique.

## Exécution du processus ETL autonome

Lorsque vous lancez le processus ETL autonome sur un serveur, il s'exécute de manière continue en arrière-plan jusqu'à ce qu'il soit arrêté. Le processus suit les instructions figurant dans les propriétés de configuration de Unica Platform afin de déterminer la fréquence, les connexions de base de données et d'autres détails pendant son fonctionnement.

Avant d'exécuter le processus ETL autonome, assurez-vous d'avoir effectué les tâches suivantes :

- Vous devez disposer des droits d'un rôle d'administrateur Interact.
- Vous devez avoir installé le processus sur un serveur et configuré les fichiers sur le serveur et dans Unica Platform correctement pour votre configuration.

**Remarque :**

Si vous exécutez le processus ETL sur Microsoft Windows pour une autre langue que l'anglais américain, utilisez `chcp` dans l'invite de commande afin de définir la page de codes pour la langue que vous employez. Par exemple, vous pouvez exécuter l'un des codes suivants : `ja_jp=932`, `zh_cn=936`, `ko_kr=949`, `ru_ru=1251` et pour `de_de`, `fr_fr`, `it_it`, `es_es`, `pt_br`, employez 1252. Pour garantir l'affichage correct des caractères, utilisez la commande `chcp` dans l'invite de commande Windows avant de lancer le processus ETL.

Une fois que vous avez installé et configuré le processus ETL autonome, vous êtes prêt à le lancer.

1. Ouvrez une invite de commande sur le serveur sur lequel le processus ETL est installé.
2. Accédez au répertoire `<Interact_home>/PatternStateETL/bin` qui contient les fichiers exécutables du processus ETL.
3. Pour exécuter le rapport ETL, ajoutez  

```
-Dcom.ibm.interact.logconfiglocation=PatternStateETL/bin/etl_log4j2.xml.
```
4. Exécutez le fichier `command.bat` (sur Microsoft Windows) ou le fichier `command.sh` (sur les systèmes d'exploitation UNIX) avec les paramètres suivants :
  - `-u <nomutilisateur>`. Il doit s'agir d'un utilisateur Unica Platform valide et vous devez avoir configuré cet utilisateur avec un accès aux sources de données **TargetDS** et **RuntimeDS** que le processus ETL utilisera.
  - `-p <mot_de_passe>` Remplacez `<password>` par le mot de passe de l'utilisateur indiqué. Si le mot de passe de cet utilisateur est vide, indiquez deux guillemets doubles (par exemple `-p ""`). Le mot de passe est facultatif lorsque vous exécutez le fichier de commandes. Si vous l'omettez avec la commande, vous êtes invité à le saisir à l'exécution de la commande.
  - `-c <nomProfil>`. Remplacez `<profileName>` par le nom exact que vous avez indiqué dans Unica Platform dans la configuration **Interact | PatternStateETL** que vous avez créée.

Le nom que vous saisissez ici doit correspondre à la valeur que vous avez indiquée dans la zone **Nouveau nom de la catégorie** lorsque vous avez créé la configuration.

- `start`. La commande `start` est requise pour démarrer le processus.

La commande complète permettant de démarrer le processus prend donc la forme suivante :

```
command.bat -u <username> -p <password> -c <profileName> start
```

Le processus ETL autonome s'exécute et continue à s'exécuter en arrière-plan jusqu'à ce que vous l'arrêtiez ou que le serveur soit redémarré.



#### Remarque :

La première fois que vous exécutez le processus, l'exécution des données de modèle d'événement accumulées peut prendre un temps considérable. Les exécutions ultérieures du processus ne seront effectuées qu'avec l'ensemble le plus récent des données de modèle d'événement et seront plus rapides.

Notez que vous pouvez aussi fournir l'argument `help` au fichier `command.bat` ou `command.sh` pour afficher toutes les options disponibles, comme dans l'exemple suivant :

```
command.bat help
```

## Arrêt du processus ETL autonome

Lorsque vous lancez le processus ETL autonome sur un serveur, il s'exécute de manière continue en arrière-plan jusqu'à ce qu'il soit arrêté.

1. Ouvrez une invite de commande sur le serveur sur lequel le processus ETL est installé.
2. Accédez au répertoire `<Interact_home>/PatternStateETL/bin` qui contient les fichiers exécutables du processus ETL.

3. Exécutez le fichier `command.bat` (sur Microsoft Windows) ou le fichier `command.sh` (sur les systèmes d'exploitation UNIX) avec les paramètres suivants :

- `-u <nomutilisateur>`. Il doit s'agir d'un utilisateur Unica Platform valide et vous devez avoir configuré cet utilisateur avec un accès aux sources de données **TargetDS** et **RuntimeDS** que le processus ETL utilisera.
- `-p <mot_de_passe>` Remplacez `<password>` par le mot de passe de l'utilisateur indiqué. Si le mot de passe de cet utilisateur est vide, indiquez deux guillemets doubles (par exemple `-p ""`). Le mot de passe est facultatif lorsque vous exécutez le fichier de commandes. Si vous l'omettez avec la commande, vous êtes invité à le saisir à l'exécution de la commande.
- `-c <nomProfil>`. Remplacez `<profileName>` par le nom exact que vous avez indiqué dans Unica Platform dans la configuration **Interact | PatternStateETL** que vous avez créée.

Le nom que vous saisissez ici doit correspondre à la valeur que vous avez indiquée dans la zone **Nouveau nom de la catégorie** lorsque vous avez créé la configuration.

- `stop`. La commande `stop` est requise pour arrêter le processus. Si vous utilisez cette commande, toutes les opérations ETL en cours se terminent avant l'arrêt du processus.

Pour arrêter le processus ETL sans attendre la fin des opérations en cours, utilisez la commande `forcestop` à la place de `stop`.

La commande complète permettant de démarrer le processus prend donc la forme suivante :

```
command.bat -u <username> -p <password> -c <profileName> stop
```

Le processus ETL autonome s'arrête.

# Chapitre 5. Présentation des offres

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

## Éligibilité d'une offre

Le but de Unica Interact est de présenter des offres éligibles. Pour l'expliquer simplement, Unica 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 Unica Interact de déterminer les offres les mieux adaptées à un client. Unica 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 fonctionnalités ne garantit qu'une offre est présentée à un client. Ces fonctionnalités influencent la probabilité qu'une offre puisse être présentée à un client. Vous pouvez utiliser le nombre de fonctionnalités de votre choix, en fonction de la meilleure solution à retenir pour votre environnement.

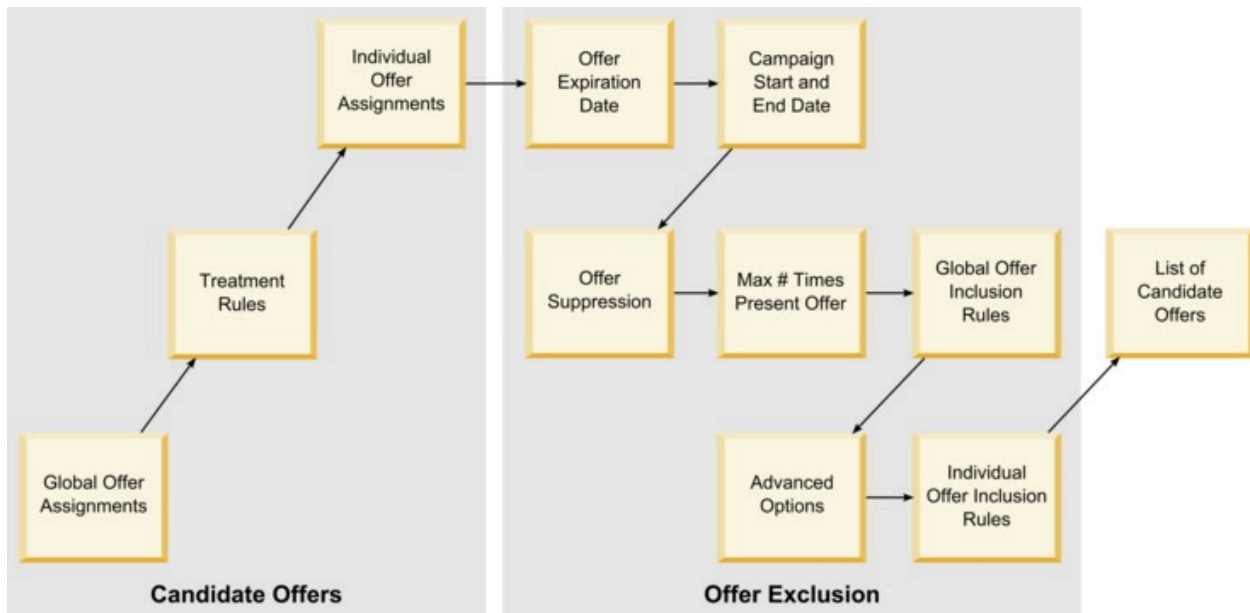
Il existe trois domaines principaux dans lesquels vous pouvez influencer sur l'éligibilité de l'offre : génération de la liste des offres candidates, détermination du score marketing et apprentissage.

## 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 qui vous permettent d'influer sur 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.





- **Affectations d'offres globales** - Vous pouvez définir des offres globales par niveau d'audience 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 Unica 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 Unica 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'une audience 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 présentations d'une offre à un client

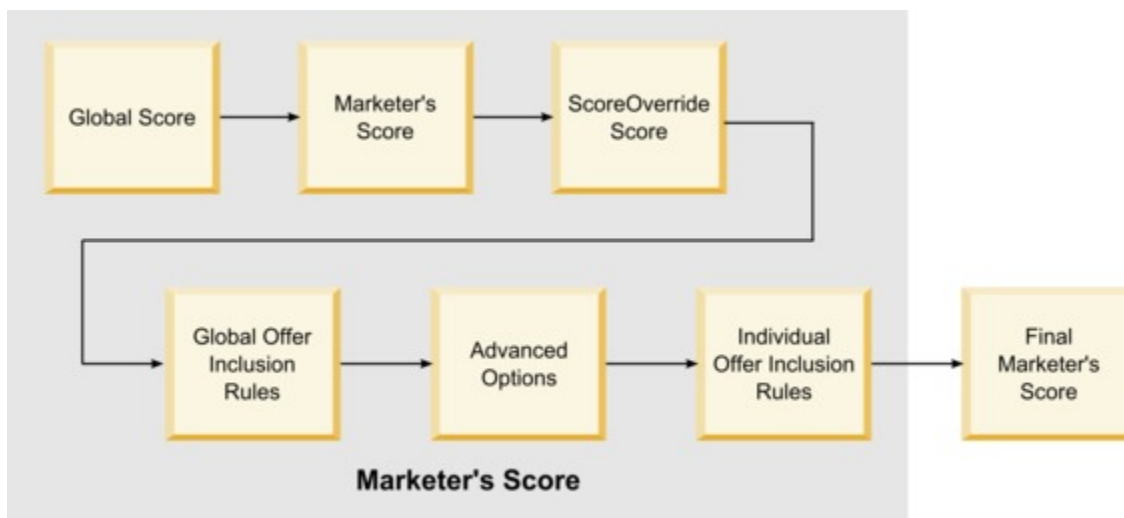
par session. Si le nombre de présentations de l'offre a déjà été atteint, l'environnement d'exécution exclut l'offre.

- **Règles d'inclusion d'offres globales** - Vous pouvez définir une expression booléenne pour filtrer les offres par niveau d'audience 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 les 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).

Ce diagramme 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 niveau d'audience à l'aide de la table des offres globales.
- **Score du spécialiste du marketing** - 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 niveau d'audience 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 Unica 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 fonctionnalité de script `aci_scoringfeature`. Pour utiliser correctement ces substitutions, vous avez besoin d'une compréhension approfondie de l'auto-apprentissage de Unica 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'auto-apprentissage 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.

## Supprimer des offres

Vous pouvez configurer l'environnement d'exécution afin qu'il supprime 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.

- Offres dont la date d'expiration est dépassée.
- Offres provenant de campagnes arrivées à expiration.
- Offres 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 Unica Interact. Si un client accepte ou refuse explicitement une offre, cette offre est supprimée lors de la session.

## Activation de la suppression des offres

La procédure ci-après permet d'activer la suppression des offres.

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

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

## Table de suppression des offres

La table de suppression des offres vous permet de supprimer une offre pour un ID audience 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 niveau d'audience doit exister dans votre base de données de profil de production. Vous pouvez créer un échantillon de table de suppression d'offre `UACI_Blacklist` en exécutant le script SQL `aci_usrtab` sur votre base de données de profil. Le script SQL `aci_usrtab` 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 audience ou votre code d'offre comprend plusieurs colonnes. Ces colonnes doivent correspondre aux noms de colonnes définis dans Unica 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 d'audience Unica Campaign. Si votre ID audience comprend plusieurs colonnes, vous pouvez les ajouter à cette table. Chaque ligne doit contenir l'ID audience auquel vous affectez l'offre par défaut, par exemple, <code>client1</code> .
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 <code>OfferCode2</code> et ainsi de suite.

## Ignorer la suppression d'offres

Le paramètre OfferSuppression pour une session peut être ignoré à l'aide des paramètres suivants :

### 1. UACIgnoreBlackList

TRUE – Lorsque ce paramètre est défini sur true, alors toutes les offres disponibles dans la table de liste noire seront affichées/retournées à l'utilisateur.

FALSE – Lorsqu'il est défini sur false, alors toutes les offres disponibles dans la table de liste noire ne seront pas affichées/retournées à l'utilisateur.

### 2. UACIgnoreSuppressionRules

TRUE – Lorsque ce paramètre est défini sur true, alors toutes les offres supprimées en temps réel seront affichées/retournées à l'utilisateur.

FALSE – Lorsque nous définissons ce paramètre sur false, alors toutes les offres supprimées en temps réel ne seront pas affichées/retournées conformément à la règle.

Si ces paramètres ne sont pas indiqués, alors ils sont considérés comme false par défaut.

Le paramètre n'affecte que les appels GetOffers dans cette session. Une fois le paramètre défini, Interact ne vérifie pas les règles de suppression lors du traitement des événements de contact et de réponse. Ainsi, un événement de contact/réponse peut être publié sur une offre supprimée.

Les règles de suppression définies sur les offres sont déclenchées par les activités en temps réel des utilisateurs.

La table de liste noire se rapporte à UACI\_BlackList ou à son équivalent, en fonction du nom donné par les clients, et ILPB peut remplir son contenu.

## 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 niveau d'audience et des affectations individuelles pour certains membres d'une audience. 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 d'offres globales ou individuelles, vous devez définir des codes de cible par défaut. `DefaultCellCode` est utilisé si aucun code de cible n'est défini dans une ligne particulière des offres par défaut ou des 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 Unica 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.

Définissez la propriété `DefaultCellCode` pour chaque niveau d'audience et type de table dans la catégorie `IndividualTreatment`.

## Définition d'offres non utilisées dans une règle de traitement

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 Unica Campaign.

Alimentez la table `UACI_ICBatchOffers` avec les codes d'offre que vous utilisez dans l'offre par défaut ou dans les tables de substitution de score. La table a le format suivant :

Nom de 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.
OfferCode4	varchar(64)	Quatrième partie du code de l'offre.
OfferCode5	varchar(64)	Cinquième partie du code de l'offre

## A propos de la table des offres globales

La table des offres globales vous permet de définir des traitements pour le niveau d'audience. 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 Unica Interact.

- Affectation d'offre globale
- Score global du spécialiste du marketing, désigné par un nombre ou une expression
- Expression booléenne de filtrage des offres



- Probabilité d'apprentissage et pondération, si vous utilisez l'auto-apprentissage Unica Interact
- Substitution d'apprentissage globale

## Affectation d'offres globales

Cette procédure permet de configurer l'environnement d'exécution afin qu'il affecte des offres globales à un niveau d'audience, 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.

Pour créer la table `UACI_DefaultOffers` avec les colonnes correctes, utilisez le fichier DDL `aci_usrtab`.

2. Définissez la propriété `Interact > profile > 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_usrtab` sur votre base de données de profil.


Le script SQL `aci_usrtab` se trouve dans le répertoire `ddl` du 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'auto-apprentissage au niveau d'audience.

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

Nom	Type	Description
Niveau d'audience	<code>varchar(64)</code>	(Obligatoire) Nom du niveau d'audience que vous affectez à l'offre par défaut, par exemple, <code>client</code>

Nom	Type	Description
		ou <i>foyer</i> . Ce nom doit correspondre au niveau d'audience, comme défini dans Unica Campaign.
OfferCode1	varchar(64)	<p>(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 <code>CodeOffre2</code> et ainsi de suite.</p> <p>Si vous ajoutez cette offre pour proposer une affectation d'offre globale, vous devez ajouter cette offre à la table <code>UACI_ICBatchOffers</code>.</p>
Note	float	Nombre définissant le score marketing de cette affectation d'offre.
OverrideTypeID	int	<p>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'offre globale.</p> <p>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.</p>
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 com-

Nom	Type	Description
		<p>portement de cette colonne dépend de la valeur de la colonne <code>EnableStateID</code>.</p> <ul style="list-style-type: none"> <li>• Si <code>EnableStateID</code> est 2, cette colonne fonctionne de la même manière que l'option <b>Considérer cette règle comme éligible si l'expression suivante est vraie...</b> 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.</li> </ul> <p>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.</p> <ul style="list-style-type: none"> <li>• Si <code>EnableStateID</code> a la valeur 3, cette colonne fonctionne de la même manière que l'option <b>Utiliser l'expression suivante comme score marketing...</b> 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 résolu.</li> <li>• Si <code>EnableStateID</code> a la valeur 1, Unica Interact ignore toute valeur dans cette colonne.</li> </ul> <p> <b>Remarque</b> : Pour attribuer un score à une offre, la séquence suivante est prise en compte.</p>

Nom	Type	Description
		 <ul style="list-style-type: none"> <li>• Champ de score final</li> <li>• Score [colonne]</li> <li>• Colonne de prédicat</li> </ul> <p>Si vous souhaitez attribuer le score depuis la colonne de prédicat, vous devez laisser la colonne de score vide.</p>
FinalScore	float	<p>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.</p>
CellCode	varchar(64)	<p>Code de cible d'un segment interactif déployé auquel vous souhaitez affecter cette offre par défaut. Si vos codes de cible comportent plusieurs zones, vous pouvez ajouter les colonnes supplémentaires.</p> <p>Vous devez fournir un code de cible si <code>OverrideTypeID</code> est 0 ou null. Si vous n'incluez pas de code de cible, l'environnement d'exécution ignore cette ligne de données.</p> <p>Si <code>OverrideTypeID</code> 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 <code>DefaultCellCode</code> pour ce niveau d'audience et cette table à des fins de génération de rapports.</p>

Nom	Type	Description
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.
EnableStateID	int	<p>La valeur de cette colonne définit le comportement de la colonne <code>Prédicat</code>.</p> <ul style="list-style-type: none"> <li>• <b>1</b> - N'utilisez pas la colonne <code>Prédicat</code>.</li> <li>• <b>2</b> - Utilisez <code>Prédicat</code> comme valeur booléenne pour filtrer l'offre. Cela suit les mêmes règles que l'option avancée <b>Considérer cette règle comme éligible si l'expression suivante est vraie...</b> dans une règle de traitement.</li> <li>• <b>3</b> - Utilisez <code>Prédicat</code> pour définir le score du vendeur. Cela suit les mêmes règles que l'option avancée <b>Utiliser l'expression suivante comme score marketing...</b> dans une règle de traitement.</li> </ul> <p>Toute ligne dans laquelle cette colonne a la valeur Null ou toute autre valeur que 2 ou 3 ignore la colonne <code>Prédicat</code>.</p>
LikelihoodScore	float	Cette colonne est utilisée uniquement pour influencer l'auto-apprentissage. Vous pouvez ajouter cette colonne avec le DDL <code>aci_scoringfeature</code> .
AdjExploreScore	float	Cette colonne est utilisée uniquement pour influencer l'auto-apprentissage. Vous pouvez ajouter cette colonne avec le DDL <code>aci_scoringfeature</code> .

Nom	Type	Description
Nombre de suppressions	int	Ce champ est destiné à la suppression d'offres exclusives. Le champ se situe sur la page Stratégie. Après avoir sauvegardé le nombre de suppressions dans l'option Avancée de la page Stratégie pour la règle, alors la valeur "Nombre de suppressions" est mise à jour dans cette colonne. La valeur par défaut est 0.
Score max	int	Par défaut, la valeur est false(0) et, sur la page Stratégie, après avoir sélectionné Score max en tant que règle et sauvegardé la stratégie, alors la valeur devient true(1).

## A propos de la table de substitution de score

La table de substitution de score vous permet de définir des traitements pour un ID d'audience 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 Unica Interact.

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

## Configuration des substitutions de score

Vous pouvez configurer Unica 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 niveau d'audience pour lequel vous souhaitez fournir des substitutions.

Pour créer un échantillon de table de substitution de score avec les colonnes correctes, utilisez le fichier DDL `aci_usrtab`.

2. Définissez la propriété `Interact > Profile > enableScoreOverrideLookup` sur **true**.
3. Définissez la propriété `scoreOverrideTable` sur le nom de la table de substitution de score pour chaque niveau d'audience pour lequel vous souhaitez fournir des substitutions.

Vous n'avez pas besoin de fournir une table de substitution de score pour chaque niveau d'audience.

## 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 une table de substitution de score échantillon, `UACI_ScoreOverride` en exécutant le script SQL `aci_usrtab` sur votre base de données de profil.

Le script SQL `aci_usrtab` 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'auto-apprentissage.

Nom	Type	Description
<i>AudienceID</i>	<code>varchar(64)</code>	(Obligatoire) Le nom de cette colonne doit correspondre au nom de la colonne définissant l'ID d'audience Unica Campaign. L'échantillon de table créé par le fichier <code>ddl aci_usrtab</code> crée cette colonne comme la colonne <code>CustomerID</code> . Si votre ID audience comprend plusieurs colonnes, vous pouvez les ajouter à cette table. Chaque ligne doit

Nom	Type	Description
		contenir l'ID audience auquel vous affectez l'offre individuelle, par exemple, <code>client1</code> . 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 <code>CodeOffre2</code> et ainsi de suite.  Si vous ajoutez cette offre pour proposer une affectation d'offre individuelle, vous devez ajouter cette offre à la table <code>UACI_ICBatchOffers</code> .
Note	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.
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



Nom	Type	Description
		<p>avancées pour les règles de traitement. Le comportement de cette colonne dépend de la valeur de la colonne <code>EnableStateID</code>.</p> <ul style="list-style-type: none"> <li>• Si <code>EnableStateID</code> est 2, cette colonne fonctionne de la même manière que l'option <b>Considérer cette règle comme éligible si l'expression suivante est vraie...</b> 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.</li> <li>• Si <code>EnableStateID</code> a la valeur 3, cette colonne fonctionne de la même manière que l'option <b>Utiliser l'expression suivante comme score marketing...</b> 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 résolu.</li> <li>• Si <code>EnableStateID</code> a la valeur 1, Unica Interact ignore toute valeur dans cette colonne.</li> </ul>

Nom	Type	Description
		<p> <b>Remarque</b> : Pour attribuer un score à une offre, la séquence suivante est prise en compte.</p> <ul style="list-style-type: none"> <li>• Champ de score final</li> <li>• Score [colonne]</li> <li>• Colonne de prédicat</li> </ul> <p>Si vous souhaitez attribuer le score depuis la colonne de prédicat, vous devez laisser la colonne de score vide.</p>
FinalScore	float	<p>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.</p>
CellCode	varchar(64)	<p>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.</p> <p>Vous devez fournir un code de cible si <code>OverrideTypeID</code> est 0 ou null. Si vous n'incluez pas de code de cible, l'environnement d'exécution ignore cette ligne de données.</p> <p>Si <code>OverrideTypeID</code> 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, défi-</p>

Nom	Type	Description
		ni dans la propriété <code>DefaultCellCode</code> pour ce niveau d'audience 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.
EnableStateID	int	<p>La valeur de cette colonne définit le comportement de la colonne <code>Prédicat</code>.</p> <ul style="list-style-type: none"> <li>• <b>1</b> - N'utilisez pas la colonne <code>Prédicat</code>.</li> <li>• <b>2</b> - Utilisez <code>Prédicat</code> comme valeur booléenne pour filtrer l'offre. Cela suit les mêmes règles que l'option avancée <b>Considérer cette règle comme éligible si l'expression suivante est vraie...</b> dans une règle de traitement.</li> <li>• <b>3</b> - Utilisez <code>Prédicat</code> pour définir le score du vendeur. Cela suit les mêmes règles que l'option avancée <b>Utiliser l'expression suivante comme score marketing...</b> dans une règle de traitement.</li> </ul> <p>Toute ligne dans laquelle cette colonne a la valeur Null ou toute autre valeur que 2 ou 3 ignore la colonne <code>Prédicat</code>.</p>
LikelihoodScore	float	Cette colonne est utilisée uniquement pour influencer l'auto-apprentissage. Vous pouvez ajouter cette colonne avec le DDL <code>aci_scoringfeature</code> .

Nom	Type	Description
AdjExploreScore	float	Cette colonne est utilisée uniquement pour influencer l'auto-apprentissage. Vous pouvez ajouter cette colonne avec le DDL <code>aci_scoringfeature</code> .
Nombre de suppressions	int	Ce champ est destiné à la suppression d'offres exclusives. Le champ se situe sur la page Stratégie. Après avoir sauvegardé le nombre de suppressions dans l'option Avancée de la page Stratégie pour la règle, alors la valeur "Nombre de suppressions" est mise à jour dans cette colonne. La valeur par défaut est 0.
Score max	int	Par défaut, la valeur est false(0) et, sur la page Stratégie, après avoir sélectionné Score max en tant que règle et sauvegardé la stratégie, alors la valeur devient true(1).

## Présentation de l'auto-apprentissage Unica 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 temps réel.

Toutefois, ces données ne sont pas des données en temps réel.

Unica 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 bayésien 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.

## Module d'apprentissage Unica Interact

Le module d'apprentissage Unica Interact surveille les réponses des visiteurs aux offres, ainsi que les attributs des visiteurs.

### Modes du module d'apprentissage

Le module d'apprentissage a deux modes généraux :

- Exploration : le module d'apprentissage présente les offres afin de pouvoir collecter suffisamment de données de réponse pour optimiser l'estimation utilisée en mode 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 utilise deux propriétés pour alterner entre les modes exploration et exploitation. Il s'agit des propriétés suivantes :

- un niveau de fiabilité que vous configurez avec la propriété `confidenceLevel`,
- une probabilité de présentation par le module d'apprentissage d'une offre aléatoire que vous configurez avec la propriété `percentRandomSelection`.

### Propriété du niveau de fiabilité

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. Une fois que 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 suffisante 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. Pour calculer les pondérations, il applique une formule qui prend comme base de départ le niveau de confiance configuré, les données historisées d'acceptation et les données de la session en cours. La formule établit un équilibre inhérent entre exploration et exploitation et renvoie la pondération adéquate.

## Propriété de sélection aléatoire

Pour éviter que le système favorise trop l'offre optimale dès les premières phases, Unica Interact présente une offre aléatoire en appliquant la propriété `percentRandomSelection` qui correspond à un pourcentage de temps. Ce pourcentage d'offres aléatoires 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, pendant 5 % du temps, le module d'apprentissage présente une offre aléatoire et ajoute les données de réponse à ses calculs.

Vous pouvez définir **%Random** pour indiquer le pourcentage de chance pour qu'une offre renvoyée soit sélectionnée de manière aléatoire, sans que les scores soient pris en compte, pour chaque zone de l'onglet Points d'interaction de la fenêtre Canal interactif.

## Mode de détermination des offres par le module d'apprentissage

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 ces informations. 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.



**Remarque :** Plusieurs événements de réponse par rapport à un événement de contact unique décale le score d'apprentissage.

## Propriétés du facteur de pondération

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 à affecter aux données récentes. `recencyWeightingPeriod` correspond à la durée considérée comme récente. Par exemple, vous configurez `recencyWeightingFactor` sur 0,30 et `recencyWeightingPeriod` sur 24. Ces paramètres signifient que les 24 heures de données précédentes représentent 30 % de toutes les données prises en compte. Pour 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.

## Données de la table de transfert consignées

Chaque session écrit les données suivantes dans une table de transfert 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 de transfert, 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 changeant une propriété de configuration.

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

Propriété de configuration	Paramètre
<code>optimizationType</code>	<b>BuiltInLearning</b>

## Attributs d'apprentissage

Le module d'apprentissage s'enrichit et évolue à l'aide des attributs visiteur, des états des modèles d'événement 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, y compris un paramètre d'événement collecté en temps réel.

Les attributs des tables dimensionnelles ne sont pas pris en charge dans l'apprentissage.

Vous pouvez configurer le nombre d'attributs de votre choix à surveiller, mais HCL 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 attributs 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. Vous pouvez aussi définir des binaires dans Interact et le système d'apprentissage réalisera automatiquement le mappage.

- 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 Unica 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 `Campaign > partitions > partition1 > Interact > learning > maxAttributeNames` définit le nombre maximal d'attributs visiteur suivis. La propriété `maxAttributeValues` définit le nombre maximum de valeurs suivies par attribut. Toutes les autres valeurs sont affectées 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	\$1 000
2	Carte Gold	\$9 000
3	Carte Bronze	\$1 000
4	Carte Bronze	\$9 000

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 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 des visiteurs 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.

Les **états de modèle d'événement** peuvent désormais être utilisés dans l'apprentissage. Le nom des modèles d'événement, ayant comme valeur de préfixe spécifiée dans le paramètre de configuration `Affinium|Campaign|partitions|partition1|Interact|`

diagramme: `eventPatternPrefix`, peut être ajouté à des attributs du modèle d'apprentissage et à des attributs d'apprentissage global.

Ils sont traités de la même manière que les attributs de profil.

Les valeurs d'un modèle d'événement peuvent être les suivantes :

0 - condition non remplie

1 - condition remplie

-1 - arrivé à expiration

-2 - désactivé

-3 - pas encore activé

## Définition d'un attribut d'apprentissage

La procédure ci-après permet de définir un attribut d'apprentissage.

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

(*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

Dans Unica Platform, pour l'environnement de conception, éditez les propriétés de configuration suivantes dans la catégorie `Campaign > partitions > partitionn > Interact > learning`.

Propriété de configuration	Paramètre
<code>attributeName</code>	<code>attributeName</code> 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 remplir 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 <code>AttributeNameCol</code> .
AttributeNameCol	Nom qualifié complet de la colonne (structure hiérarchique, à partir de la table de profils) dans laquelle se trouve <code>AttributeName</code> . 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 profils) dans laquelle se trouve la valeur associée à <code>AttributeName</code> .

Par exemple, examinez la table de profils suivante et la table des dimensions associée.

**Tableau 7. MaTableDeProfil**

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

**Tableau 8. MaTableDeDimension**

KeyField	CardType	CardBalance
Key1	Carte Gold	1000
Key2	Carte Gold	9 000
Key3	Carte Bronze	1000

**Tableau 8. MaTableDeDimension (suite)**

KeyField	CardType	CardBalance
Key4	Carte Bronze	9 000

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

**Tableau 9. UACI\_AttributeList**

AttributeName	AttributeNameCol	AttributeValueCol
Carte Gold	MaTableDeProfil.MaTableDeDimension. CardType	MaTableDeProfil.MaTableDeDimension. CardBalance
Carte Bronze	MaTableDeProfil.MaTableDeDimension. CardType	MaTableDeProfil.MaTableDeDimension. CardBalance

## Unica Interact AutoBinning

Dans Interact, l'algorithme d'apprentissage intégré fonctionne en partie en sauvegardant et en analysant les valeurs des attributs de profil au moment de l'appel et de la réponse de l'offre. Certains attributs peuvent avoir un nombre virtuellement illimité de valeurs uniques. Cependant, en l'absence de ressources illimitées dans le système Interact, vous ne pouvez enregistrer qu'un petit nombre de ces valeurs. En outre, il est souvent plus raisonnable de réaliser l'examen en se basant sur des plages de valeurs. Vous pouvez utiliser cette fonctionnalité pour créer des binaires dans Interact et le sous-système d'apprentissage réalisera automatiquement le mappage.

Vous pouvez créer des définitions binaires dans **Interact > Apprentissage global > Page Toutes les définitions binaires**. Tout en ajoutant ou en éditant une définition binaire, vous pouvez sélectionner des attributs de profil dans la liste qui regroupe tous les attributs des tables de profil mappées. La définition binaire peut être de type Plage ou Liste. Le type "Plage" peut uniquement utiliser des opérateurs mathématiques, tandis que le type "Liste" peut uniquement recourir à des opérateurs "contient" et constitue une liste de valeurs.

Exemple de binaire de type "Plage" :

Faibles revenus < =30000

30000 < revenus moyens < =60000

Revenus élevés > 60000

Exemple de binaire de type "Liste" :

Nouvelle-Angleterre : MA, NH, CT

Nord-Ouest : MI, IL

Une définition binaire est un ensemble de données générales concernant tous les canaux interactifs et tous les modèles d'apprentissage.

Toutes les définitions binaires seront déployées dans le cadre des données de déploiement générales. Vous pouvez les déployer dans n'importe quel canal interactif, pour l'un et pour tous. Ensuite, les nouvelles définitions binaires seront sauvegardées dans la mémoire cache, laquelle est uniquement visible dans le sous-système d'auto-apprentissage.

Lors de la publication d'un contact ou d'une réponse, la valeur d'un attribut de profil est mappée à un binaire si celui-ci existe. Les valeurs "binaires" sont utilisées lors de la journalisation des tables d'apprentissage. Si des binaires sont définis pour l'attribut et que la valeur d'attribut ne fait partie d'aucune définition binaire, la valeur d'attribut sera journalisée en tant que AUTRE dans les tables d'apprentissage.

## Configuration de l'environnement d'exécution pour qu'il reconnaisse les modules d'apprentissage externes

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 Unica Platform.

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

1. Dans Unica 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
<code>optimizationType</code>	<b>ExternalLearning</b>
<code>externalLearningClass</code>	Nom de classe de l'apprentissage externe
<code>externalLearningClassPath</code>	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 Unica Platform, une copie de la classe ou des fichiers JAR doit être présente au même emplacement sur chaque serveur.

2. Redémarrez le serveur d'exécution Unica Interact pour que les changements prennent effet.

# Chapitre 6. Compréhension de l'API Unica Interact

Unica 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 Unica 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.

L'API Unica Interact permet également les communications entre le client utilisateur final et le serveur via JavaScript.

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

Par défaut, tous les paramètres sont stockés dans la session actuelle. Dès lors, toutes les API ultérieures sont affectées. Les paramètres `UACIPreRemoveParameter` et `UACIPostRemoveParameter` peuvent servir à supprimer des paramètres indésirables de la session.

Un indicateur transitoire est ajouté à tous les paramètres d'API. Si la valeur d'un paramètre est `INVOCATION (1)`, ce paramètre est alors effectif uniquement lors du processus d'invocation de cette API. La valeur par défaut est `SESSION (0)`.



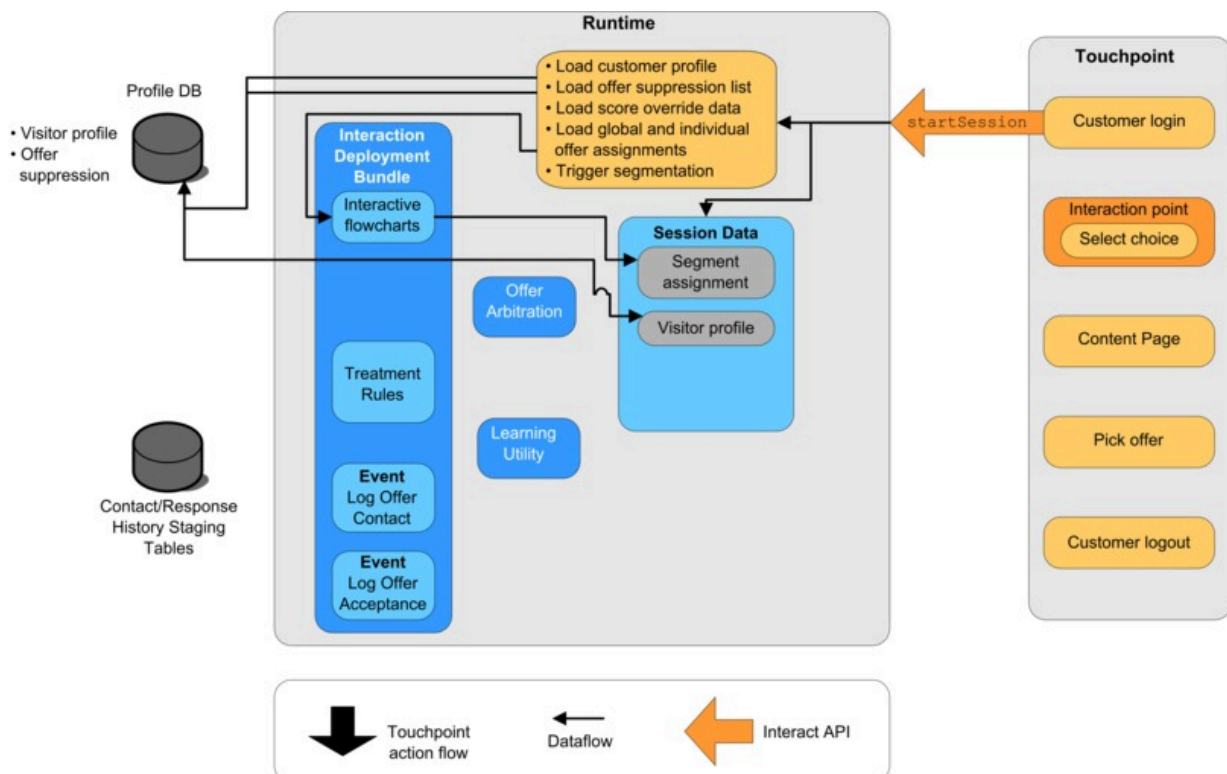
## Flux de données de l'API Unica Interact

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

Ce diagramme montre une implémentation simple de l'API Unica 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. L'interaction est simple, mais plusieurs événements se produisent à la fois dans le point de contact et sur le serveur d'exécution :

1. Démarrage d'une session
2. Accès à une page
3. Sélection d'une offre
4. Fermeture de la session



## Démarrage de la session

Lorsque le visiteur se connecte, il déclenche une méthode `startSession`.

La méthode `startSession` effectue quatre opérations :

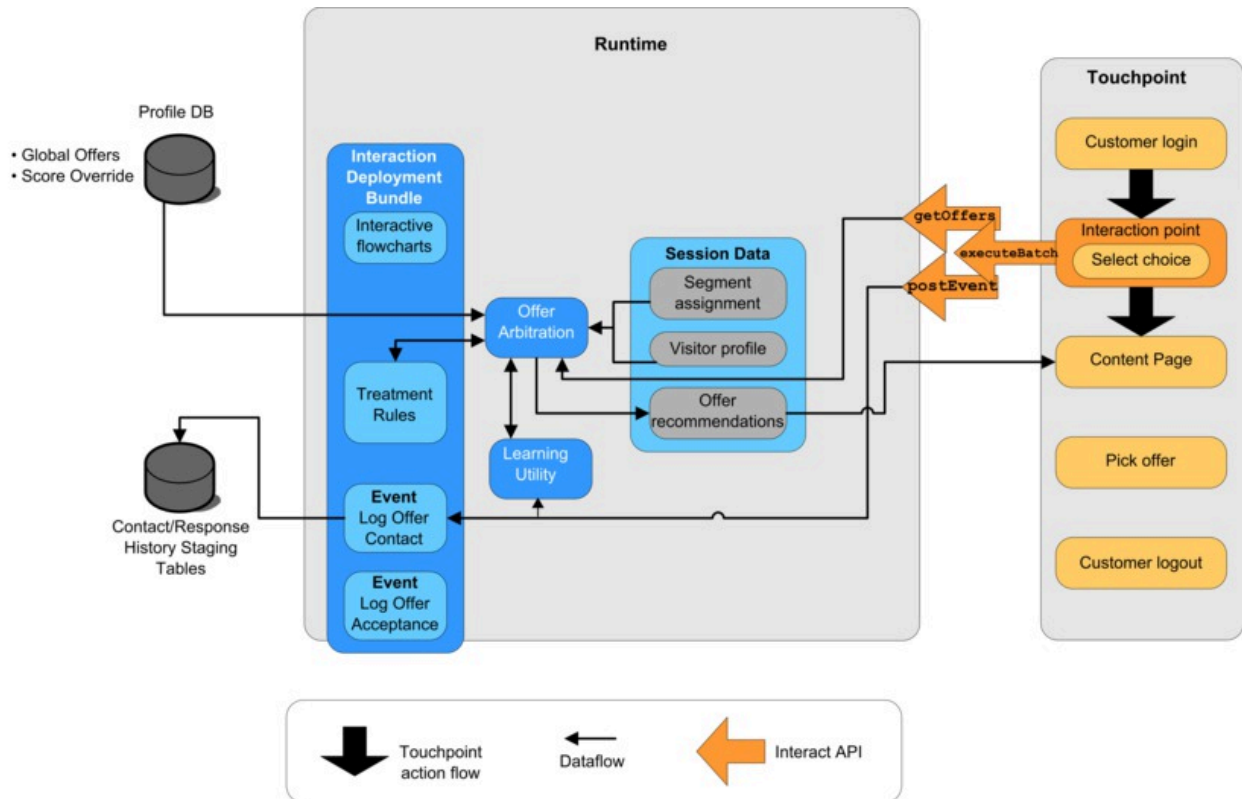
1. Elle crée une session d'exécution
2. Elle envoie une requête de chargement des données du profil client dans la session
3. Elle envoie une requête d'utilisation des données du profil et démarre un diagramme temps réel pour placer le client dans des segments. Cette exécution de diagramme s'exécute en mode asynchrone.
4. Le serveur d'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 la session.

## Accès à une page

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` permettant de sélectionner une offre.

## Sélection d'une offre

Ce diagramme illustre l'appel d'API qui déclenche la méthode `executeBatch`.

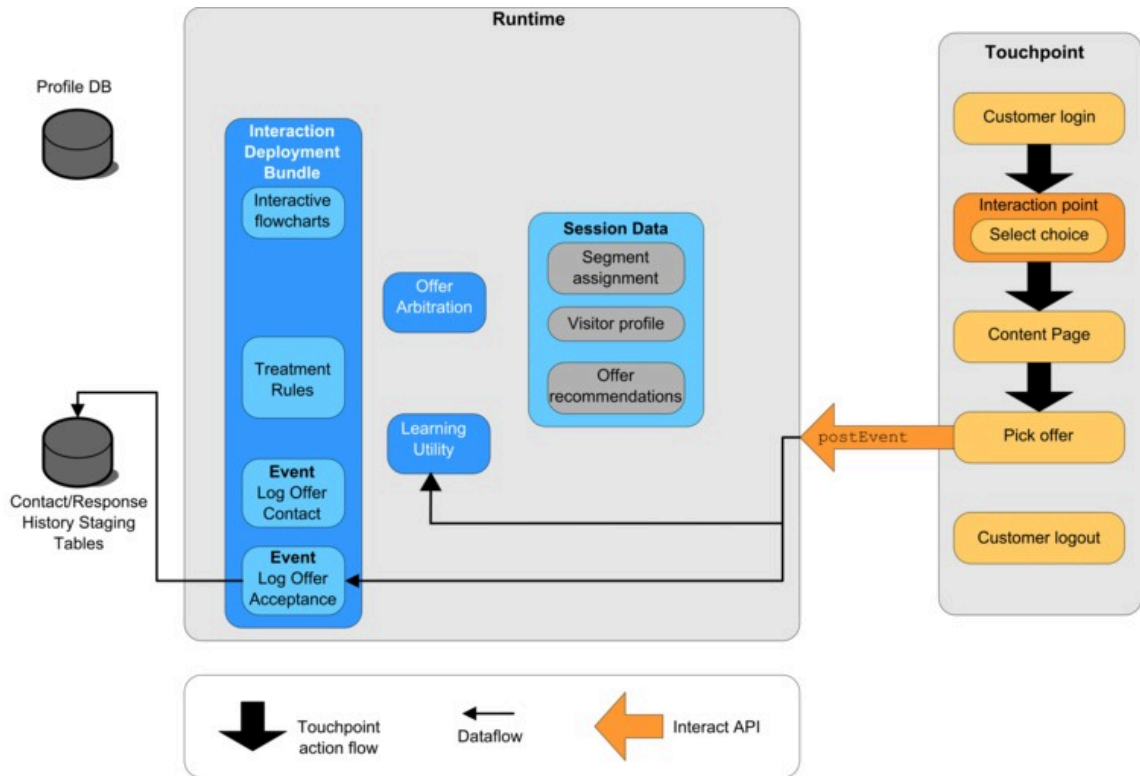


La méthode `executeBatch` vous permet d'appeler plusieurs méthodes dans un seul appel au serveur d'exécution. Cette méthode `executeBatch` appelle deux autres méthodes, `getOffers` et `postEvent`. La méthode `getOffers` demande une liste d'offres. Le serveur d'exécution utilise les données de segmentation, la liste de suppression des offres, les règles de traitement et le module d'apprentissage pour proposer un ensemble d'offres. Le serveur d'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'une des offres (offre Sélect.)

Ce diagramme illustre la méthode `postEvent`.

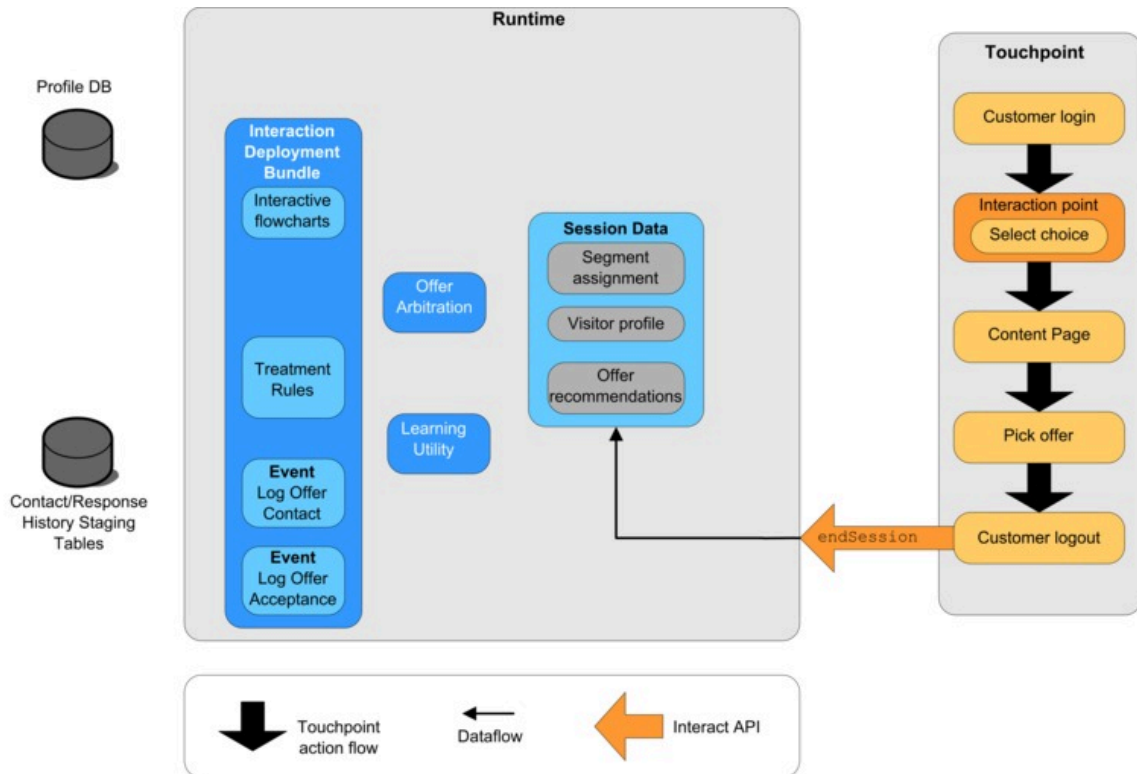


La commande d'interface utilisateur associée à la sélection de l'offre est configurée pour envoyer une autre méthode `postEvent`. Cet événement envoie une demande de journalisation de l'acceptation de l'offre dans l'historique des réponses.

## Fermeture de la session

Une fois que le visiteur a sélectionné l'offre, il en a terminé avec le site Web et il se déconnecte. La commande de déconnexion est liée à la méthode `endSession`.

Ce diagramme illustre la méthode `endSession`.



La méthode `endSession` termine la session. Si le visiteur oublie de se déconnecter, un délai d'attente de session configurable permet de faire prendre fin à toutes les sessions. Si vous souhaitez conserver des données passées à la session, telles que les informations incluses dans les paramètres des méthodes `startSession` ou `setAudience`, collaborez avec la personne chargée de créer les diagrammes temps réel. 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 le diagramme temps réel qui contient le processus d'instantané.

## Exemple simple de planification d'interaction

Dans cet exemple, vous créez une interaction pour le site Web d'une société de téléphonie cellulaire. Vous créez trois offres différentes, configurez la consignation de ces offres, affectez des codes de traitement à l'offre et afficher une série d'images associées aux offres.

## Processus de conception

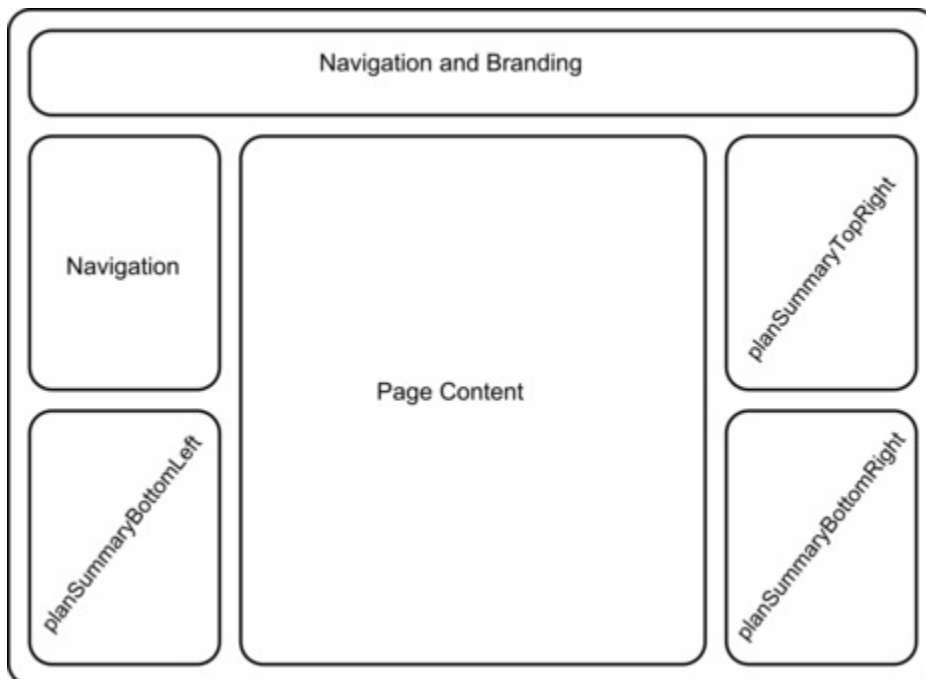
Pour concevoir une interaction pour ce client, vous :

1. Identifiez les exigences de la page de récapitulatif du client
2. Créez des points d'interaction pour les exigences des offres
3. Configurez la consignation des offres
4. Création de codes de traitement
5. Associez une série d'images en alternance aux offres

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

## Identification des exigences de la page de récapitulatif du plan prévu pour un téléphone

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 :

Exigence	Implémentation
<p>L'une des offres doit être affichée dans une zone dédiée aux offres de mise à niveau</p> <p>La zone de la page qui affiche l'offre de mise à niveau doit être définie. En outre, une fois qu'Unica Interact a sélectionné une offre à afficher, les informations doivent être journalisées.</p>	<ul style="list-style-type: none"> <li>• Point d'interaction : <code>ip_planSummaryBottomRight</code></li> <li>• Événement : <code>evt_logOffer</code></li> </ul>
<p>Deux offres pour les mises à niveau du téléphone</p> <p>Chaque zone de la page qui affiche les mises à niveau du téléphone doit être définie.</p>	<ul style="list-style-type: none"> <li>• Point d'interaction : <code>ip_planSummaryTopRight</code></li> <li>• Point d'interaction : <code>ip_planSummaryBottomLeft</code></li> </ul>
<p>Pour l'analyse, vous devez journaliser les offres qui sont acceptées et celles qui sont refusées.</p>	<ul style="list-style-type: none"> <li>• Événement : <code>evt_offerAccept</code></li> <li>• Événement : <code>evt_offerReject</code></li> </ul>
<p>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.</p>	<p><code>NameValuePair</code></p>
<p>Affichez trois images en alternance sur la page. Associez ces images aux offres.</p>	

## Création de points d'interaction

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 affiche 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)
```

Cet appel de réponse renvoie un objet réponse qui inclut 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 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 = " */
        }
    }
}
```

## Configuration de la consignation

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);

```

Il n'est pas nécessaire de définir `UACIOfferTrackingCode` dans cet exemple. Le serveur d'exécution d'Unica Interact journalise automatiquement la dernière liste recommandée de traitements sous forme de contacts si vous ne spécifiez pas `UACIOfferTrackingCode`.

## Création de codes de traitement

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);
```

## Association d'images à des offres

Pour la deuxième zone de la page qui affiche une mise à niveau du téléphone, vous avez écrit quelque chose pour modifier l'image affichée toutes les 30 secondes. Vous décidez 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 offering attribute value and store somewhere;
            // this will be the first image to display
        }
        else if(x==1)
        {
            // grab offering attribute value and store somewhere;
            // this will be the second image to display
        }
        else if(x==2)
```

```

{
    // grab offering 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 possède 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), journalisez l'offre acceptée et l'offre refusée. (Dans ce scénario, les offres 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`.

## Conception de l'intégration de l'API Unica Interact

La génération de votre intégration de l'API Unica Interact à votre point de contact requiert un certain nombre de tâches de conception avant de commencer la mise en oeuvre. 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 récapitulatif 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 Unica Interact pour définir les détails. Un auteur de diagramme doit concevoir des diagrammes temps réel 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 passées 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 d'indicateur requises pour déterminer quand vous devez vous journaliser les des contacts et les réponses des offres.

## Points à prendre en compte

Lorsque vous concevez une interaction, gardez à l'esprit les effets qu'aucune offre éligible, qu'un serveur d'exécution inaccessible et que le temps de traitement ont sur l'interaction. Soyez spécifique lorsque vous définissez des rejets d'offre. Etudiez les fonctionnalités facultatives du produit qui peuvent améliorer l'interaction.

Lorsque vous concevez votre interaction :

### **Création d'un contenu de remplissage par défaut**

Créez un contenu de remplissage par défaut, un message de marque bénin ou un contenu vide, pour chaque point d'interaction où les offres peuvent être présentées. Ce contenu de remplissage est utilisé au cas où aucune offre éligible ne peut être proposée au visiteur en cours dans la situation actuelle. Vous affectez ce contenu de remplissage par défaut en tant que la chaîne par défaut du point d'interaction.

## Inclusion d'une autre méthode de présentation de contenu

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.

## Examen du temps requis pour l'exécution des diagrammes

Lorsque vous déclenchez des événements qui resegmentent votre visiteur, y compris `postEvent` et `setAudience`, n'oubliez pas que l'exécution de diagrammes demande un certain temps. La méthode `getOffers` attend que la segmentation soit terminée avant que la méthode `getOffers` s'exécute. Une resegmentation trop fréquente peut dégrader les performances des réponses à l'appel `getOffers`.

## Détermination de ce que signifie un "rejet d'offre".

Plusieurs rapports, comme le rapport de récapitulatif des performances des offres du canal, présentent le nombre de rejets d'une offre. Ce rapport indique le nombre de fois qu'une méthode `postEvent` a déclenché une action Journaliser le refus de l'offre. Vous devez déterminer si l'action Journaliser le refus de l'offre correspond à un rejet réel, par exemple un clic sur un lien intitulé **Non, merci** ou à une offre qui a été ignorée, comme une page qui affiche trois bannières publicitaires différentes, dont aucune n'est sélectionnée.

## Détermination des fonctionnalités de sélection des offres à utiliser

Plusieurs fonctionnalités facultatives vous permettent d'améliorer la sélection des offres Unica Interact. Il s'agit des fonctionnalités suivantes :

- Apprentissage
- Suppression d'offre
- Affectations d'offres individuelles
- Autres éléments du service d'offres

Vous devez déterminer combien, le cas échéant, de ces fonctionnalités facultatives amélioreraient vos interactions.

## Authentification API

Cette fonctionnalité vous permet d'utiliser l'option permettant d'activer l'authentification des appels d'API Unica Interact. Lorsqu'elle est activée, Unica Interact RT vérifie si une requête API entrante dispose d'un jeton d'authentification valide. Si ce n'est pas le cas, il l'identifie à l'aide du nom d'utilisateur et du mot de passe fournis avec Unica Platform ou LDAP. Cette requête est rejetée si le jeton n'est pas valide et si le nom d'utilisateur et le mot de passe ne sont pas valides. Chaque jeton qui dispose d'une durée de vie non prolongeable du cache de session Unica Interact expire. Le jeton est lié à l'identifiant de session, fourni lors de l'appel d'API `startSession()`. Par conséquent, tous les autres appels d'API valident ce jeton avant de poursuivre. Cette fonction est désactivée par défaut.



### Remarque :

Des applications tierces qui interceptent ou modifient les demandes réseau (par ex. : équilibreurs de charge, pare-feux, serveurs proxy, filtres de requêtes, antivirus, etc.) peuvent interférer avec l'API/le déploiement/les requêtes d'authentification d'Interact.

L'authentification d'Interact peut échouer lors du déploiement, malgré des données d'identification correctes, si des applications tierces (équilibreurs de charge, pare-feux, serveurs proxy, filtres de requêtes, antivirus, etc.) sont configurées de façon à intercepter ou à réécrire des messages sur le réseau ou le serveur d'applications. Par conséquent, la longueur du contenu des messages de déploiement d'Interact peut être définie sur 0 par un tiers, même si le contenu de la réponse d'authentification est correct. Campaign a proposé une solution de contournement qui lit le contenu, peu importe l'altération de la longueur du contenu, afin de restaurer des réponses d'authentification de déploiement correctes jusqu'à ce que la raison de cette altération soit connue. Il ne s'agit pas d'un problème lié à Interact et vous devez contacter le support de Campaign afin de corriger le problème.

# Chapitre 7. Gestion de l'API Unica Interact

Chaque fois que vous utilisez la méthode `startSession`, vous créez une session d'exécution Unica 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 Unica Interact avec votre point de contact.

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

## Paramètres régionaux et API Unica Interact

Vous pouvez utiliser Unica 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`. Si la bibliothèque client (`interact_client.jar`) est utilisée pour connecter l'application client aux serveurs d'exécution Interact, un proxy HTTP peut être configuré avec une authentification entre l'exécution de l'application du client et d'Interact. Pour activer le proxy pour les API Interact, ajoutez les paramètres JVM ci-dessous et redémarrez le serveur d'applications sur lequel l'application est déployée.

`-Dcom.hcl.interact.http.proxyHost=<Adresse IP du serveur proxy>`

`-Dcom.hcl.interact.http.proxyHost=<Port d'écoute du serveur proxy>`

Voici ci-dessous les paramètres à utiliser lorsque l'authentification est requise pour le serveur proxy.

`-Dcom.hcl.interact.http.proxyUsername= <Nom d'utilisateur pour se connecter au serveur proxy. A ne pas indiquer si aucune authentification n'est requise>`



-Dcom.hcl.interact.http.proxyPassword=<Mot de passe pour se connecter au serveur proxy.  
A ne pas indiquer si aucune authentification n'est requise>

## A propos de la surveillance JMX

Unica 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.

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



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

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

Cette procédure permet de configurer Unica Interact pour utiliser la surveillance JMX avec le protocole RMI.

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

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

Propriété de configuration	Paramètre
<code>protocole</code>	<b>RMI</b>
<code>port</code>	Numéro de port du service JMX
<code>enableSecurity</code>	<b>Faux</b> La mise en oeuvre Unica Interact du protocole RMI ne prend pas en charge la sécurité.

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

Cette procédure permet de configurer Unica Interact pour utiliser la surveillance JMX avec le protocole JMXMP.

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

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

L'adresse par défaut pour la surveillance pour le protocole JMXMP est :

`service:jmx:jmxmp://RuntimeServer:port.`

1. Vérifiez que les bibliothèques `InteractJMX.jar` et `jmxremote_optional.jar` sont spécifiées dans le chemin d'accès aux classes, dans l'ordre. Si elles ne s'y trouvent pas, ajoutez-les.
2. Dans Unica Platform pour l'environnement d'exécution, éditez les propriétés de configuration suivantes dans la catégorie `Interact > monitoring`.

Propriété de configuration	Paramètre
<code>protocole</code>	<b>JMXMP</b>
<code>port</code>	Numéro de port du service JMX
<code>enableSecurity</code>	<b>False</b> pour désactiver la sécurité, ou <b>True</b> pour activer la sécurité

## Configuration d'Unica Interact afin d'utiliser les scripts jconsole pour la surveillance JMX

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

Le script jconsole utilise par défaut le protocole JMXMP pour la surveillance. Les paramètres par défaut du fichier `jconsole.bat` sont les suivants :

### 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%
```

### 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
```

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 `HOSTE` sur le nom d'hôte du serveur d'exécution que vous souhaitez surveiller.
4. Définissez `PORT` sur le port sur lequel JMX est configuré pour écouter avec la propriété `Interact > monitoring > port`.
5. **Facultatif** : Si vous utilisez le protocole RMI pour la surveillance, ajoutez un commentaire avant la connexion JMXMP et supprimez le commentaire avant la connexion RMI.

## Attributs JMX

Plusieurs attributs sont disponibles pour la surveillance JMX. Les attributs de l'environnement de conception incluent la surveillance ETL de l'historique des contacts et des réponses. Les attributs de l'environnement d'exécution inclut les exceptions, plusieurs attributs de diagramme différents, l'environnement local, le consignateur et les statistiques du pool d'unités d'exécution. Plusieurs attributs des statistiques de service sont également disponibles. 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.

### Attributs du moniteur ETL de l'historique des réponses et des contacts

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 10. 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 10. Moniteur ETL de l'historique des réponses des contacts (suite)**

Attribut	Description
ErrorCount	Nombre de 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 d'historique des réponses. Cette valeur est calculée uniquement pour les opérations

**Tableau 10. Moniteur ETL de l'historique des réponses des contacts (suite)**

Attribut	Description
	qui ont réussi et pour lesquelles au moins un enregistrement a été écrit dans la table d'historique des réponses.
LastExecutionAverage	Nombre de millisecondes nécessaire au module de l'historique des contacts et des réponses pour exécuter chaque 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 réussi. 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 <code>UACI_CHStaging</code> qui sont déplacés 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

**Tableau 10. Moniteur ETL de l'historique des réponses des contacts (suite)**

Attribut	Description
	des réponses est en cours d'exécution.
NumberOfResponseHistoryRecordsMarked	Nombre d'enregistrements de l'historique des réponses dans la table <code>UACI_RHStaging</code> qui sont déplacés 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.

### Attributs d'exception

Les attributs d'exception font partie de l'environnement d'exécution.

**Tableau 11. 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.

### Attributs des statistiques du moteur de diagramme

Les attributs des statistiques du moteur de diagramme font partie de l'environnement d'exécution.



**Tableau 12. 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 tâches en attente d'exécution par les unités d'exécution de processus du diagramme.
CurrentJobsInSchedulerQueue	Nombre de tâches en attente d'exécution 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.
numExecutionsCompleted	Nombre total d'exécutions de diagramme terminées.

**Tableau 12. Statistiques du moteur de diagramme (suite)**

Attribut	Description
numExecutionsStarted	Nombre total d'exécutions de diagramme qui ont démarré.

### Attributs des diagrammes spécifiques par canal interactif

Les attributs des diagrammes spécifiques par canal interactif font partie de l'environnement d'exécution.

**Tableau 13. 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.

### Attributs d'environnement local

Les attributs d'environnement local font partie de l'environnement d'exécution.

**Tableau 14. Environnement local**

Attribut	Description
locale	Paramètres régionaux du client JMX.

### Attributs de configuration du consignateur

Les attributs de configuration du consignateur font partie de l'environnement d'exécution.

**Tableau 15. Configuration du consignateur**

Attribut	Description
category	Modifiez la catégorie de journal dans laquelle le niveau de journalisation peut être manipulé.

### Attributs des statistiques du pool d'unités d'exécution des services

Les attributs des statistiques du pool d'unités d'exécution des services font partie de l'environnement d'exécution.

**Tableau 16. 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 16. 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 InFlushCachetoDB-Queue	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 tâches dans la file d'attente pour le pool d'unités d'exécution utilisé pour l'historique des

**Tableau 16. Statistiques du pool d'unités d'exécution des services (suite)**

Attribut	Description
	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.
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'éligi-

**Tableau 16. Statistiques du pool d'unités d'exécution des services (suite)**

Attribut	Description
	bilité, les activités de l'événement et les statistiques par défaut.

## Attributs des statistiques de service

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 des données personnalisées en vue de les écrire dans une table (événement qui utilise le paramètre d'événement `UACICustomLoggerTableName`).
- `Statistiques par défaut` - Service qui collecte les statistiques relatives au nombre d'utilisations de la chaîne par défaut du point d'interaction.
- `Statistiques d'éligibilité` - Service qui écrit les statistiques relatives aux offres éligibles.
- `Statistiques de l'activité de l'événement` - 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 mémoire cache de l'historique des réponses` - Service qui écrit dans les tables de transfert d'historique des réponses.
- `Statistiques de réponse intersession` - Service qui collecte les données de suivi de réponse intersession.

**Tableau 17. Statistiques de service**

Attribut	Description
Nombre	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

**Tableau 17. Statistiques de service (suite)**

Attribut	Description
	une grande différence entre ExecTimeInsidMutex et ExecTimeMillis, il peut être nécessaire d'éditer 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 consacrée 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.
TotalRowsInserted	Nombre total de lignes insérées dans la base de données.

### **Attributs des statistiques de service - utilitaire de chargement de la base de données**

Les attributs des statistiques de service - utilitaire de chargement de la base de données font partie de l'environnement d'exécution.

**Tableau 18. Statistiques de service - Utilitaire de chargement de la base de données**

Attribut	Description
ExecTimeOfWriteToCache	Durée en millisecondes consacrée 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.



**Tableau 18. Statistiques de service - Utilitaire de chargement de la base de données (suite)**

Attribut	Description
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.

### Attributs des statistiques d'API

Les attributs des statistiques d'API font partie de l'environnement d'exécution.

**Tableau 19. Statistiques d'API**

Attribut	Description
endSessionCount	Nombre d'appels de l'API <code>endSession</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
endSessionAverage	Temps écoulé pour chaque appel de l'API <code>endSession</code> , en millisecondes.


**Tableau 19. Statistiques d'API (suite)**

Attribut	Description
executeBatchCount	Nombre d'appels de l'API <code>executeBatch</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
executeBatchAverage	Temps écoulé pour chaque appel de l'API <code>executeBatch</code> , en millisecondes.
getOffersCount	Nombre d'appels de l'API <code>getOffers</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
getOffersAverage	Temps écoulé pour chaque appel de l'API <code>getOffer</code> , en millisecondes.
getProfileCount	Nombre d'appels de l'API <code>getProfile</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
getProfileAverage	Temps écoulé pour chaque appel de l'API <code>getProfileAverage</code> , en millisecondes.
getVersionCount	Nombre d'appels de l'API <code>getVersion</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
getVersionAverage	Temps écoulé pour chaque appel de l'API <code>getVersion</code> , en millisecondes.
loadOfferSuppressionAverage	Temps écoulé pour chaque appel de l'API <code>loadOfferSuppression</code> .
LoadOffersBySQLCount	Nombre d'appels de l'API <code>LoadOffersBySQL</code> depuis la dernière réinitialisation ou le dernier démarrage du système.

**Tableau 19. Statistiques d'API (suite)**

Attribut	Description
LoadOffersBySQLAverage	Temps écoulé pour chaque appel de l'API <code>LoadOffersBySQL</code> , en millisecondes.
loadProfileAverage	Temps écoulé pour chaque appel de l'API <code>loadProfile</code> , en millisecondes.
loadScoreOverrideAverage	Temps écoulé pour chaque appel de l'API <code>loadScoreOverride</code> , en millisecondes.
postEventCount	Nombre d'appels de l'API <code>postEvent</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
postEventAverage	Temps écoulé pour chaque appel de l'API <code>postEvent</code> , en millisecondes.
runSegmentationAverage	Temps écoulé pour chaque appel de l'API <code>runSegmentation</code> , en millisecondes.
setAudienceCount	Nombre d'appels de l'API <code>setAudience</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
setAudienceAverage	Temps écoulé pour chaque appel de l'API <code>setAudience</code> , en millisecondes.
setDebugCount	Nombre d'appels de l'API <code>setDebug</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
setDebugAverage	Temps écoulé pour chaque appel de l'API <code>setDebug</code> , en millisecondes.

**Tableau 19. Statistiques d'API (suite)**

Attribut	Description
startSessionCount	Nombre d'appels de l'API <code>startSession</code> depuis la dernière réinitialisation ou le dernier démarrage du système.
startSessionAverage	Temps moyen écoulé pour le dernier appel de l'API <code>startSession</code> .
ActiveSessionCount	<p>Nombre de sessions actuellement actives dans l'instance d'exécution Interact.</p> <p> <b>Remarque :</b> l'attribut <code>ActiveSessionCount</code> dans JMX MBean <code>com.unicacorp.interact:type=api,group=Statistics</code> ne prend pas en compte les événements arrivés à expiration et peut par conséquent indiquer un nombre de sessions actives incorrect.</p>

### Attributs des statistiques de l'optimiseur d'apprentissage

Les attributs des statistiques de l'optimiseur d'apprentissage font partie de l'environnement d'exécution.

**Tableau 20. Statistiques de l'optimiseur d'apprentissage**

Attribut	Description
LearningOptimizerAcceptCalls	Nombre événements d'acceptation transmis au module d'apprentissage.

**Tableau 20. Statistiques de l'optimiseur d'apprentissage (suite)**

Attribut	Description
LearningOptimizer AcceptTrackingAverage	Nombre total de millisecondes consacré à la journalisation des événements d'acceptation dans le module d'apprentissage.
LearningOptimizerContactCalls	Nombre événements de contact transmis au module d'apprentissage.
LearningOptimizer ContactTrackingAverage	Nombre total de millisecondes consacré à la journalisation 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.
LearningOptimizer LogOtherTrackingAverage	Durée en millisecondes consacrée à la journalisation des autres événements (ni contact, ni acceptation) dans le module d'apprentissage.
LearningOptimizer NonRandomCalls	Nombre d'applications de la mise en œuvre d'apprentissage configurée.
LearningOptimizer RandomCalls	Nombre de fois où la mise en œuvre d'apprentissage configuré a été contournée et où la sélection aléatoire a été appliquée.
LearningOptimizer RecommendCalls	Nombre de demandes de recommandation passées au module d'apprentissage.
LearningOptimizer RecommendAverage	Durée totale en millisecondes consacrée à la logique de recommandation d'apprentissage.

## Attributs des statistiques d'offre par défaut

Les attributs des statistiques d'offre par défaut font partie de l'environnement d'exécution.

**Tableau 21. Statistiques d'offre par défaut**

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

## Attributs des répartiteurs de messages déclenchés

Les attributs des répartiteurs de messages déclenchés font partie de l'environnement d'exécution.

**Tableau 22. Répartiteurs de messages déclenchés**

Attribut	Description
NumRequested	Nombre total d'offres demandées pour la répartition à l'aide de ce répartiteur.
NumDispatched	Nombre total d'offres réparties par ce répartiteur.
AvgExecutionTime	Durée moyenne en millisecondes utilisée par ce répartiteur pour répartir une offre. Seules les offres réparties sur les passerelles sont comptabilisées dans le calcul.
CurrentQueueSize	Nombre d'offres en attente de répartition.

**Tableau 22. Répartiteurs de messages déclenchés (suite)**

Attribut	Description
GatewayInvocation	Nombre d'offres et durée de répartition moyenne en millisecondes lors de la répartition sur chaque passerelle par ce répartiteur. Le format de cette valeur est <code>{gateway name=[number of offers, average dispatching time]}</code> .

### Attributs des passerelles de messages déclenchés

Les attributs des passerelles de messages déclenchés font partie de l'environnement d'exécution.

**Tableau 23. Passerelles de messages déclenchés**

Attribut	Description
NumValidationRequested	Nombre total d'offres demandées par cette passerelle pour la validation.
NumValidated	Nombre total d'offres validées par cette passerelle.
AvgValidationTime	Durée moyenne en millisecondes utilisée par cette passerelle pour valider une offre. Seules les offres validées sont comptabilisées dans le calcul.
NumDeliveryRequested	Nombre total d'offres demandées par cette passerelle pour la distribution.

**Tableau 23. Passerelles de messages déclenchés (suite)**

Attribut	Description
NumDelivered	Nombre total d'offres distribuées par cette passerelle.
AvgDeliveryTime	Durée moyenne en millisecondes utilisée par cette passerelle pour distribuer une offre. Seules les offres distribuées sont comptabilisées dans le calcul.

### Attributs des messages de messages déclenchés

Les attributs des messages de messages déclenchés font partie de l'environnement d'exécution.

**Tableau 24. Messages de messages déclenchés**

Attribut	Description
ProcessSuccessCount	Nombre total de fois que ce message déclenché a été exécuté.
AvgSuccessProcessTime	Durée moyenne en millisecondes passée par ce message déclenché pour chaque exécution réussie.
ProcessErrorCount	Nombre total de fois que ce message déclenché a été exécuté incorrectement.
AvgErrorProcessTime	Durée moyenne en millisecondes passée par ce message déclenché pour chaque exécution non réussie.
SelectBranchCount	Nombre total de fois que la sélection de branches a été exécutée lors



**Tableau 24. Messages de messages déclenchés (suite)**

Attribut	Description
	du traitement des messages déclenchés.
AvgSelectBranchTime	Durée moyenne en millisecondes passée par l'exécution de la sélection de branches lors du traitement des messages déclenchés.
SelectOfferCount	Nombre total de fois que la sélection d'offres a été exécutée lors du traitement des messages déclenchés.
AvgSelectOfferTime	Durée moyenne en millisecondes passée par l'exécution de la sélection d'offres lors du traitement des messages déclenchés.
SelectChannelCount	Nombre total de fois que la sélection de canaux a été exécutée lors du traitement des messages déclenchés.
AvgSelectChannelTime	Durée moyenne en millisecondes passée par l'exécution de la sélection de canaux lors du traitement des messages déclenchés.
FlowchartWaitCount	Nombre total de fois que ce message déclenché a attendu la fin de la segmentation.
AvgFlowchartWaitTime	Durée moyenne en millisecondes pendant lequel ce message déclen-

**Tableau 24. Messages de messages déclenchés (suite)**

Attribut	Description
	ché a attendu la fin de la segmentation dans chaque exécution.
WaitFlowchartTimeoutCount	Nombre total de fois que ce message déclenché est arrivé à expiration alors qu'il attendait la fin de la segmentation.

**Tableau 25. Informations des passerelles Activity Orchestrator**

Attribut	Description
NumReceived	Nombre de messages reçus par Activity Orchestrator Receiver.
NumProcessed	Nombre de messages traités pour Activity Orchestrator Receiver.
AvgProcessTime	Durée de traitement moyenne du nombre de messages traités pour Activity Orchestrator Receiver.

**Tableau 26. Informations sur les statistiques Kafka**

Attributs	Description
NumMessagesInTopic	Nombre de messages enregistrés dans la rubrique, en plus du nom de la rubrique.
Exécution en cours	Statut d'exécution du serveur Kafka.
TopicDetails	Les détails des rubriques, en plus des responsables et des suiveurs.
ListOfTopics	Liste des rubriques du serveur Kafka, excepté la rubrique par défaut.

**Tableau 27. Informations sur les statistiques Zookeeper**

Attribut	Description
Exécution en cours	Statut d'exécution du serveur Zookeeper.

## Opérations JMX

Plusieurs opérations sont disponibles pour la surveillance 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 <code>Interact/conf/interact_log4j.properties</code> sur débogage.
Configuration du consignateur	activateError	Définissez le niveau de journalisation pour le fichier journal défini dans <code>Interact/conf/interact_log4j.properties</code> sur erreur.
Configuration du consignateur	activateFatal	Définissez le niveau de journalisation pour le fichier journal défini dans <code>Interact/conf/interact_log4j.properties</code> sur fatal.
Configuration du consignateur	activateInfo	Définissez le niveau de journalisation pour le fichier journal défini dans <code>Interact/conf/interact_log4j.properties</code> sur info.
Configuration du consignateur	activateTrace	Définissez le niveau de journalisation pour le fichier journal défini dans <code>Inter-</code>

Groupe	Attribut	Description
		ract/conf/interact_log4j.properties sur trace.
Configuration du consigneur	activateWarn	Définissez le niveau de journalisation pour le fichier journal défini dans Interact/conf/interact_log4j.properties sur avertissement.
Environnement local	changeLocale	Modifiez les paramètres régionaux du client JMX. Les environnements locaux Unica Interact pris en charge sont de, en, es et fr.
ContactResponseHistory ETLMonitor	Réinitialiser	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 le nombre d'offres par défaut qu'au démarrage.

## Surveillance des threads

Afin d'aider à surveiller les activités du système, le lien "Informations sur le fil" est ajouté à la page Admin de l'interface utilisateur d'exécution d'Interact. Le lien Thread Info affiche les threads qui s'exécutent actuellement sur le serveur d'applications hébergeant cette instance d'exécution avec les informations suivantes.

- ID : ID de ce fil de discussion.
- Nom du fil : Nom de ce fil.
- Alive : spécifie si ce thread est actif.

- Démon : spécifie si ce thread est un thread démon.
- Temps CPU : Le temps CPU total en millisecondes que ce thread a consommé.
- Temps utilisateur : Le temps CPU total de l'utilisateur en millisecondes que ce thread a consommé.
- Temps d'attente : Le temps total en millisecondes que ce thread a passé en état d'attente.
- Wait Count : Le nombre d'occurrences où ce thread est mis en état d'attente.
- Temps de blocage : Le temps total en millisecondes pendant lequel ce thread est dans l'état bloqué.
- Block Count : Le nombre d'occurrences où ce thread est mis dans l'état bloqué.
- État : L'état actuel de ce fil.
- Verrou attendu: le verrou que ce thread attend. Il est vide s'il n'attend aucun verrou.
- Moniteurs suspendus: le moniteur (verrou) que ce thread détient actuellement.
- Stack Trace: la trace de pile actuelle de ce thread. Par défaut, il n'affiche que l'entrée du haut, et cliquer dessus se développe pour afficher la pile complète.

# Chapitre 8. Classes et méthodes de l'API Java, SOAP et REST d'Unica Interact

Les sections suivantes listent les exigences et d'autres informations que vous devez connaître avant d'utiliser l'API Unica 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 Unica Interact dispose d'un adaptateur client Java™ qui utilise une sérialisation Java™ via HTTP. En outre, Unica Interact fournit un WSDL pour la prise en charge des clients SOAP. Le WSDL présente le même ensemble de fonctions que l'adaptateur de client Java™, de sorte que les sections suivantes, à l'exception des exemples, continuent de s'appliquer.



**Remarque :** Les occurrences multiples de tout paramètre dans un même appel d'API ne sont pas prises en charge.

## Unica Interact API Classes

L'API Unica Interact se base sur la classe `InteractAPI`.

Il existe 6 interfaces de prise en charge.

- `AdvisoryMessage`
- `BatchResponse`
- `NameValuePair`
- `Offre`
- `OfferList`
- `Réponse`

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

## Méthodes pour indiquer les paramètres d'authentification si l'authentification d'API est activée avant les appels d'API

Si l'authentification d'API est activée, alors vous pouvez utiliser les méthodes suivantes pour indiquer les paramètres d'authentification, comme les identifiants ou le jeton, avant tout appel d'API.

```
setAuthenticationParameter
```

Cette méthode configure le paramètre d'authentification sur l'appel d'API demandé.

```
setAuthenticationParameter(String username, String password)
```

- nom d'utilisateur : Unica Platformusername
- mot de passe

```
setAuthenticationParameter(String token)
```

token: jeton obtenu depuis le serveur

## Prérequis de la sérialisation Java™ via HTTP

L'adaptateur client Java™ utilise la sérialisation Java™ via HTTP.

Les prérequis d'utilisation de l'adaptateur client Java™ pour la sérialisation Java™ via HTTP sont les suivants :

1. Ajoutez le fichier suivant à votre variable `CLASSPATH` :

```
Unica Interact_Home/lib/interact_client.jar
```

2. Tous les objets échangés entre le client et le serveur se trouvent dans le module `com.unicacorp.interact.api`. Pour plus de détails, consultez la documentation Javadoc de l'API Unica Interact installée sur le serveur d'exécution dans le répertoire `Unica Interact_Home/docs/apiJavaDoc`. Vous pouvez l'afficher en ouvrant le fichier `index.html` qui se trouve dans ce répertoire dans n'importe quel navigateur.
3. Pour obtenir une instance de la classe `InteractAPI`, appelez la méthode statique `getInstance` avec l'adresse URL du serveur d'exécution d'Unica Interact.

## Prérequis SOAP

Pour pouvoir accéder au serveur d'exécution avec SOAP, vous devez effectuer plusieurs tâches prérequisées pour configurer votre environnement.



**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 Unica Interact API à l'aide du kit d'outils SOAP de votre choix.

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

Lorsque vous configurez SOAP à l'aide des fichiers XML de WSDL, vous devez remplacer vos URL par le nom d'hôte et le port du serveur d'exécution.

Le texte du WSDL est disponible à la fin du document Unica Interact - Guide d'administration.



## 2. Installez et configurez le serveur d'exécution.

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

## 3. Vérifiez que vous utilisez la version appropriée de SOAP.

Unica Interact utilise axis2 1,3 comme infrastructure SOAP sur les serveurs d'exécution Unica 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](#)

Unica Interact a été testé avec les clients [axis2](#), XFire, JAX-WS-Ri, DotNet, SOAPUI et IBM® RAD SOAP.

## Configuration requise pour REST

Pour appeler l'API Unica Interact, vous pouvez utiliser des appels au format JSON (JavaScript™ Object Notation) sur HTTP, désignés dans le présent document par le terme API REST. Les performances de l'API REST sont meilleures que celles de SOAP, bien que l'adaptateur de sérialisation Java™ reste la méthode la plus rapide pour appeler l'API Unica Interact.

Avant de vous servir de l'API REST, prenez connaissance des points suivants :

- L'adresse URL qui prend en charge les appels REST de l'API Unica Interact est : `http://Unica_Interact_Runtime_Server:PORT/interact/servlet/RestServlet`, dans laquelle vous devez indiquer l'adresse IP ou le nom d'hôte réel du serveur d'exécution Unica Interact et le port sur lequel Unica Interact est déployé.
- Il existe deux classes Unica Interact propres à l'API REST : `RestClientConnector`, qui sert de classe auxiliaire pour la connexion à une instance d'exécution Unica Interact via REST au format JSON, et `RestFieldConstants`, qui décrit le format sous-jacent du message JSON utilisé pour les demandes et les réponses de l'API.
- Un exemple de client REST est fourni dans le répertoire `Unica Interact _Home/samples/javaApi/InteractRestClient.java`. Cet exemple de code est simple, mais il constitue un bon point de départ pour comprendre comment l'API REST est utilisée.

- Pour une description complète des classes de l'API REST et d'autres informations relatives à l'API Unica Interact, consultez la documentation JavaDoc installée sur le serveur d'exécution dans le répertoire `Unica Interact_Home/docs/apiJavaDoc`.
- L'API REST renvoie des valeurs SessionID et des messages au format HTML avec échappement et non au format Unicode.
- L'authentification de l'API est activée. Les identifiants ou le jeton doivent être indiqués dans l'en-tête de la requête.
  - Paramètres d'en-tête d'entrée
    - Références d'identification
      - `m_user_name`
      - Paramètre d'en-tête - Nom d'utilisateur de la plateforme
      - `m_user_password`
      - Paramètre d'en-tête - Mot de passe de l'utilisateur de la plateforme
    - Jeton
      - `m_tokenId`
      - Paramètres d'en-tête - jeton
  - Paramètres d'en-tête de sortie
    - `m_tokenId`
    - Paramètres d'en-tête - jeton

Enfin, l'API REST prend en charge toutes les méthodes qui sont prises en charge par les autres protocoles pour l'utilisation de l'API Unica Interact.

## API JavaDoc

En complément du document Unica Interact - Guide d'administration, la documentation Javadoc de l'API Unica Interact est installée avec le serveur d'exécution. Elle est installée pour référence dans le répertoire `Unica Interact_Home/docs/apiJavaDoc`.

## Exemples d'API

Tous les exemples contenus dans ce guide ont été créés à l'aide de la sérialisation Java™ via l'adaptateur HTTP. Les classes générées à partir du WSDL peuvent varier en fonction du

kit d'outils SOAP et des options que vous sélectionnez. Si vous utilisez SOAP, ces exemples peuvent ne pas fonctionner 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 surensemble 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 appelez 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 Unica 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 passer un paramètre d'événement avec le type personnalisé `NameValuePair`. Dans la classe `NameValuePair`, vous pouvez

utiliser les méthodes `setName` et `setValueAsNumeric` pour changer 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 d'instantané dans votre diagramme temps réel pour écrire les données dans une table de base de données. Si vous envisagez d'utiliser des processus d'instantané, 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 Unica Interact prend en charge les méthodes de cette classe.

Vous devez compiler votre mise en œuvre par rapport à `interact_client.jar` situé dans le répertoire `lib` de votre installation d'environnement d'exécution Unica Interact. `interact_client.jar` dépend de `log4j-api`, `log4j-core`, `commons-lang`, `commons-lang3` et `commons-httpclient`. Ces dépendances doivent être installées manuellement et leur emplacement doit être ajouté manuellement au chemin d'accès aux classes de l'application client qui utilise `interact_client.jar`.

## Terminer la session

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.

```
endSession(String sessionID, NameValuePair[] parameters)
```

- **sessionID** - Chaîne unique identifiant la session.
- **parameters** - Objets NameValuePair identifiant tous les paramètres à transmettre avec la requête d'API.

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 :

- ID session
- 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

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

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

- **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());
    }
}
}
```



## Écriture de demandes XML `executeBatch()` pour l'API SOAP d'Interact

Utilisez cette procédure pour écrire les demandes XML `executeBatch()` destinées à l'API SOAP d'Unica Interact.

La demande XML pour les appels d'API SOAP d'une seule opération (`startSession`, `getOffers`, `setAudience`, `endSession`, etc.) ne doit pas être copiée et collée directement dans l'appel `executeBatch()` de plusieurs opérations. Les structures des demandes WSDL et XML des sous-commandes contenues dans les appels `executeBatch()` diffèrent légèrement de celles des appels d'API pour une seule opération. Les différences de structure génèrent des réponses d'erreur de la part du serveur si les éléments XML sont copiés et collés depuis les demandes d'API pour une seule opération dans des demandes `executeBatch` pour plusieurs opérations.

Exemples de réponse d'erreur :

```

** XML Response Element:
  <ns0:faultstring>org.apache.axis2.databinding.ADBException:
Unexpected subelement audienceID</ns0:faultstring>
** Interact Server Exception: java.lang.Exception:
  org.apache.axis2.databinding.
ADBException: Unexpected subelement audienceID at
*** ...
  com.unicacorp.interact.api.soap.service.v1.xsd.CommandImpl$Factory.parse
(CommandImpl.java:1917) at

```

Utilisez cette procédure pour écrire une demande XML `executeBatch()`. Pour les valeurs des paramètres, vous pouvez faire référence dans cette procédure à des demandes d'appel d'API destinées à une seule opération, mais ne copiez et collez pas les éléments XML.

1. Utilisez un outil de traitement WSDL (SoapUI par exemple) pour créer une demande XML `executeBatch()` au bon format à partir du fichier WSDL d'Unica Interact.
2. Ajoutez des sous-commandes à la demande, après la définition WSDL des éléments enfants d'`executeBatch()`.
3. Entrez les arguments des sous-commandes après la définition WSDL des éléments enfants d'`executeBatch()`.

## getInstance

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

```
getInstance(String URL)
```



**Important :** Chaque application que vous créez à l'aide de l'API Unica 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 mise en œuvre du client SOAP a sa propre façon de créer l'URL du nœud 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");
```

## Obtenir des offres

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

```
getOffers(String sessionId, String interactionPoint, int numberOfOffers,
    NameValuePair[] parameters)
```

- **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.
- **parameters** - Objets `NameValuePair` identifiant tous les paramètres à transmettre avec la requête d'API.

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`
- `ID session`
- `StatusCode`
- `NameValuePair`

## 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 = "Overview Page Banner 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());

```

Les positions décimales des scores d'offre sont retournées dans la réponse `getOffer` dans la paire `NameValue`. Lorsque les offres sont retournées vers les canaux entrants à l'origine de la requête, les canaux utilisent les scores pour hiérarchiser les offres. Les chiffres décimaux ne sont pas supprimés, ainsi le canal sait quelle offre détient le score le plus élevé si des nombres décimaux sont retournés.

## Obtenir des offres pour des points d'interaction multiples

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

```
getOffersForMultipleInteractionPoints(String sessionID, String requestStr,
    NameValuePair[] parameters)
```

- **sessionID** - Chaîne identifiant la session en cours.
- **requestStr** - Chaîne fournissant un tableau d'objets `GetOfferRequest`.
- **parameters** - Objets `NameValuePair` identifiant tous les paramètres à transmettre avec la requête d'API.

Chaque objet `GetOfferRequest` spécifie 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 incluses dans cette instance de `GetOfferRequest` (c'est-à-dire que Unica Interact appliquera le dédoublement).
- **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 préséance 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 Unica 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 dédoublement 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
- ID session
- 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 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/dd/yyyy 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.

## Obtenir le profil

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

```
getProfile(String sessionID, NameValuePair[] parameters)
```

- **sessionID** - Chaîne identifiant l'ID session.
- **parameters** - Objets `NameValuePair` identifiant tous les paramètres à transmettre avec la requête d'API.

## 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
- ID session
- 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());

```

## Obtenir la version

La méthode `getVersion` renvoie la version de la mise en oeuvre actuelle du serveur d'exécution Unica Interact.

```
getVersion()
```

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

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

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

- **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 transmettre avec l'événement.

Si cet événement déclenche la resegmentation, vous devez veiller à ce que toutes les données requises par les diagrammes temps réel 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 profils), vous devez inclure un `eventParameter` 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 temps réel.

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



**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, Unica 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, Unica 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.

La resegmentation n'efface pas les résultats de la segmentation précédente pour le niveau d'audience en cours. Vous pouvez utiliser le paramètre `UACIExecuteFlowchartByName` pour définir des diagrammes spécifiques à exécuter. La méthode `getOffers` attend la fin de la resegmentation avant de s'exécuter. Par conséquent, un retard est possible si vous appelez une méthode `postEvent`, qui déclenche une resegmentation juste avant un appel `getOffers`.

## Valeur de retour

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

- `AdvisoryMessages`
- `ApiVersion`
- `ID session`
- `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());
```



## Définir l'audience

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

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

- **sessionID** : chaîne identifiant l'ID session.
- **audienceID** - Tableau d'objets `NameValuePairImpl` définissant l'ID audience.
- **audienceLevel** - Chaîne définissant le niveau d'audience.
- **parameters** - Objets `NameValuePairImpl` identifiant tous les paramètres à transmettre 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 surensemble 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 audience. 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`

- ID session
- StatusCode

## Exemple

Dans cet exemple, le niveau d'audience 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 niveau d'audience 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());

```

## Définir le débogage

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

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

- **sessionID** - Chaîne identifiant l'ID session.
- **debug** - Valeur booléenne qui active ou désactive les informations de débogage. Les valeurs valides sont `true` ou `false`. Si elle est `true`, Unica 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`

- ID session
- StatusCode

## Exemple

L'exemple suivant illustre le changement 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());
```

## Démarrer la session

La méthode `startSession` crée et définit une session d'exécution.

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

`startSession` peut déclencher cinq actions au maximum :

- Créer une session d'exécution.
- Charger les données de profil du visiteur correspondant au niveau d'audience 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 temps réel correspondant au niveau d'audience 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.

- **elyOnExistingSession** - Valeur booléenne qui définit si cette session utilise une session nouvelle ou existante. Les valeurs valides 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, toutes 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 valides sont `true` ou `false`. Si elle est `true`, Unica 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 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 Unica Campaign.
- **audienceID** - Tableau d'objets `NameValuePairImpl` dans lequel les noms doivent correspondre aux noms de colonne physique de toute table contenant l'ID audience.
- **audienceLevel** - Chaîne définissant le niveau d'audience.
- **parameters** - Objets `NameValuePairImpl` identifiant tous les paramètres à transmettre avec `startSession`. Ces valeurs peuvent être utilisées pour la segmentation.

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

Si `audienceID` ou `audienceLevel` ne sont pas valides et si `relyOnExistingSession` a la valeur `false`, l'appel `startSession` échoue. Si le `interactiveChannel` est non 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, Unica Interact crée une nouvelle session.

Si `relyOnExistingSession` a la valeur `true` 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`
- ID session
- `StatusCode`



**Remarque :** En raison des limitations des nombres à virgule flottante IEEE 754, il n'est pas possible de représenter avec exactitude toutes les valeurs numériques, y compris AudienceID, SessionID et les paramètres de session, dans Interact, même s'il est possible de le faire dans la table de profils. En outre, des nombres entiers supérieurs à  $2^{53}$  (9007199254740992) ne peuvent pas être utilisés en tant que valeurs d'ID audience.

## 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` est une méthode qui gère l'objet de réponse renvoyé par `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());
}
```

## Dédoublonnage des offres dans les attributs d'offre

Grâce à l'API Unica Interact, deux appels API fournissent les offres : `getOffers` et `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` peut empêcher le retour des offres en double au niveau d'OfferID, mais ne peut pas

dédoubler les offres dans la catégorie d'offre. Ainsi, pour qu'Unica Interact ne renvoie qu'une seule offre de chaque catégorie d'offre, vous devez auparavant faire appel à une solution de contournement. Grâce à l'introduction de deux nouveaux paramètres dans l'appel API `startSession`, il est désormais possible de dédoubler les offres dans les attributs d'offre, comme la catégorie.

Cette liste récapitule les paramètres qui ont été ajoutés dans l'appel API `startSession`. Pour plus d'informations sur ces paramètres ou sur tout aspect de l'API Unica Interact, reportez-vous au document *Unica Interact - Guide d'administration* ou aux fichiers Javadoc inclus avec votre installation d'Unica Interact, dans `<Unica Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Pour créer un appel API `startSession` avec un dédoublonnage des offres, afin que les appels `getOffer` suivants ne renvoient qu'une seule offre de chaque catégorie, vous devez inclure le paramètre `UACIOfferDedupeAttribute` dans l'appel API. Vous pouvez indiquer un paramètre au format `name, value, type`, comme indiqué ici :

```
UACIOfferDedupeAttribute, <attributeName>, string
```

Dans cet exemple, vous remplacez `<attributeName>` par le nom de l'attribut d'offre que vous souhaitez utiliser comme critère pour le dédoublonnage, comme `Category`.



**Remarque :** Unica Interact examine les offres dotées de la même valeur d'attribut indiquée (par exemple `Category`) et effectue le dédoublonnage pour tout supprimer sauf l'offre dotée du score le plus élevé. Si les offres dotées de l'attribut en double ont aussi des scores identiques, Unica Interact renvoie une sélection aléatoire parmi les offres correspondantes.

- `UACINoAttributeDedupeIfFewerOf`. Lorsque vous incluez `UACIOfferDedupeAttribute` dans l'appel `startSession`, vous pouvez aussi définir ce paramètre `UACINoAttributeDedupeIfFewerOf` afin d'indiquer le comportement lorsque la liste des offres après dédoublonnage ne contient plus assez d'offres pour satisfaire la demande d'origine.

Par exemple, si vous avez défini `UACIOfferDedupeAttribute` de telle sorte qu'il utilise la catégorie d'offre afin de dédoubler les offres, et que l'appel `getOffers` suivant demande le renvoi de huit offres, le dédoublonnage peut renvoyer un nombre d'offres éligibles inférieur à 8. Dans ce cas, si vous indiquez la valeur `true` pour le paramètre `UACINoAttributeDedupeIfFewerOf`, vous ajoutez certaines offres dédoublonnées dans la liste éligible pour satisfaire au nombre d'offres demandé. Dans cet exemple, si vous attribuez la valeur `false` au paramètre, le nombre d'offres renvoyé est inférieur au nombre demandé.

`UACINoAttributeDedupeIfFewerOf` est défini par défaut sur `true` par défaut.

Par exemple, supposons que vous indiquez pour le paramètre `startSession` que le critère de dédoublonnage est la catégorie de l'offre, comme indiqué ici :

```
UACIOfferDedupeAttribute, Category, string;UACINoAttributeDedupeIfFewerOffer,
0, string
```

Ensemble, ces paramètres indiquent à Unica Interact de dédoubler les offres en fonction de l'attribut d'offre "Category," et de ne renvoyer que les offres dédoublonnées même si le nombre d'offres renvoyées est inférieur au nombre demandé (`UACINoAttributeDedupeIfFewerOffer` a pour valeur `false`).

Lorsque vous émettez un appel API `getOffers`, l'ensemble d'origine des offres éligibles peut inclure les offres suivantes :

- `Category=Electronics` : Offre A1 avec un score égal à 100 et Offre A2 avec un score égal à 50.
- `Category=Smartphones` : Offre B1 avec un score égal à 100, Offre B2 avec un score égal à 80 et Offre B3 avec un score égal à 50.
- `Category=MP3Players` : Offre C1 avec un score égal à 100, Offre C2 avec un score égal à 50.

Dans ce cas, il y avait deux offres en double qui correspondaient à la première catégorie, trois offres en double qui correspondaient à la deuxième catégorie et deux offres en double qui correspondaient à la troisième catégorie. Les offres renvoyées sont les offres dotées des scores les plus élevés de chaque catégorie, à savoir Offre A1, Offre B1 et Offre C1.

Si l'appel API `getOffers` a demandé six offres, cet exemple a attribué la valeur `false` à `UACINoAttributeDedupeIfFewerOffer` et seulement trois offres sont renvoyées.

Si l'appel API `getOffers` a demandé six offres et que cet exemple a omis le paramètre `UACINoAttributeDedupeIfFewerOffer`, ou lui a spécifiquement attribué la valeur `true`, certaines offres en double sont incluses dans le résultat pour satisfaire au nombre demandé.

## Paramètres réservés

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

### paramètres API

Fonction	Paramètre	La description
Connectez-vous à la table personnalisée (Périmètre-session)	<code>UACICustomLoggerTableName</code>	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 <code>NameValuePair</code> 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 <code>NULL</code> . Vous pouvez gérer le processus qui écrit dans la base de données avec les propriétés de configuration <code>customLogger</code> .
Se connecter à un fichier	<code>UACILogSeparationFileName</code>	Nom de fichier journal personnalisé : ce paramètre permet à l'environnement d'exécution d'écrire tous les journaux appartenant

Fonction	Paramètre	La description
spécifique (Périmètre-session)		<p>à cette session dans le fichier journal spécifié. Le fichier avec le nom spécifié est créé à l'emplacement du journal par défaut, s'il est indisponible.</p> <p>Dans le cas d'un simulateur, le paramètre est automatiquement ajouté. Le format du nom du fichier journal est <code>&lt; Simulator-Basic- {scenarioName}.log &gt;</code>, <code>&lt; Simula- tor-Advanced- {scenarioName}.log &gt;</code> et <code>&lt; Simulator-Coverage- {scenarioName}.log &gt;</code> pour les scénarios de base, avancés et de couverture, respectivement.</p>
Types de réponse multiples (Périmètre-invocation)	UACILogToLearning	<p>Nombre entier ayant la valeur 1 ou 0. 1 indique que l'environnement d'exécution doit journaliser l'événement comme une acceptation dans le système d'apprentissage ou autoriser la suppression de l'offre dans une session. 0 indique que l'environnement d'exécution ne doit ni journaliser l'événement dans le système d'apprentissage ni autoriser la suppression de l'offre dans une session. Ce paramètre vous permet de créer plusieurs méthodes <code>postEvent</code> enregistrant différents types de réponse sans influencer 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 conjointement avec <code>UACIRe- sponseTypeCode</code>. Si vous ne définissez pas</p>

Fonction	Paramètre	La description
		UACILOGTOLEARNING, l'environnement d'exécution prend la valeur par défaut de 0 (sauf si l'événement déclenche un contact de journal, une acceptation ou un rejet).
	UACIResponseTypeCode	Valeur représentant un code de type de réponse. La valeur doit être une entrée valide dans la table UA_UsrResponseType
Suivi des réponses (Périmètre-invocation)	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 des réponses intersessions, vous pouvez utiliser le paramètre UACIOfferTracking-codeType pour définir le type de code de suivi que vous utilisez autre que le code de traitement.
Suivi des réponses inter-sessions	UACIOfferTrackingCode-Type	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_-

Fonction	Paramètre	La description
(Périmètre- invocation)		TrackingType. Vous pouvez ajouter d'autres codes personnalisés dans cette table.
Exécution de diagramme spécifique (Périmètre- invocation)	UACIExecuteFlowchart- ByName	Si vous définissez ce paramètre pour une méthode qui déclenche la segmentation ( <code>startSession</code> , <code>setAudience</code> ou un <code>postEvent</code> qui déclenche la resegmentation), au lieu d'exécuter tous les diagrammes pour le niveau d'audience actuel, Unica Interact exécute uniquement les diagrammes nommés. Vous pouvez fournir une liste des diagrammes séparés par un caractère de barre verticale ( ).
Possibilité de limiter le champ d'offre dans l'API get-Offers à Interact RunTime (Périmètre- Session)	UACIOfferFields et UACIExcludeOfferFields	Vous pouvez inclure/exclure les attributs UACIOfferFields et UACIExcludeOfferFields d'une offre en transmettant ces attributs dans le paramètre de <code>startSession</code> , <code>setAudience</code> ou un <code>postEvent</code> .
Réinitialiser le modèle d'événement (Périmètre- Invocation)	<ul style="list-style-type: none"> <li>• ResetEventPattern. Ce paramètre réinitialise les états de modèle d'événement pour tous les modèles pour l'ID audience.</li> </ul>	Vous pouvez réinitialiser les états d'un modèle spécifique ou de tous les modèles pour un ID audience particulier associé à la session. Vous devez afficher un événement avec un paramètre spécial. L'événement ne doit avoir aucune action explicite associée.



Fonction	Paramètre	La description
	<ul style="list-style-type: none"> <li>• <code>PatternToReset</code>. Ce paramètre réinitialise les états de modèle d'événement pour des modèles d'événement spécifiques. Il est possible de réinitialiser ID modèle ou Nom du modèle. Ce paramètre peut uniquement être transmis avec le paramètre <code>ResetEventPattern</code>.</li> </ul>	
Réinitialiser la suppression de l'offre (Périmètre-Invocation)	<ul style="list-style-type: none"> <li>• <code>ResetOfferSuppression</code>. Ce paramètre réinitialise les règles de suppression d'offre pour toutes les offres pour l'ID d'audience.</li> <li>• <code>OfferToResetSuppression</code>. Ce paramètre réinitialise les règles de suppression d'offre pour des offres spécifiques. Il est possible de réini-</li> </ul>	Vous pouvez réinitialiser les états d'une règle de suppression d'offre spécifique ou toutes les règles de suppression d'offre pour un ID audience particulier associé à la session. Vous devez afficher un événement avec un paramètre spécial. L'événement ne doit avoir aucune action explicite associée.

Fonction	Paramètre	La description
	<p>tialiser l'ID offre ou le code d'offre. Pour réinitialiser la suppression d'offre pour des offres ayant un code d'offre à plusieurs parties, le code d'offre doit être séparé par des virgules. Par exemple, Code d'offre1, Code d'offre2, Code d'offre3, etc. Ce paramètre peut uniquement être transmis avec le paramètre ResetOfferSuppression.</p>	
Supprimer le paramètre de session avant l'appel de l'API (Périmètre-Session)	UACIPreRemoveParameter. Noms séparés par des virgules des paramètres que vous souhaitez supprimer de la session avant l'appel d'API.	Ce paramètre est utile lorsque vous souhaitez supprimer certains paramètres déjà définis dans la session par de précédents appels d'API. Plusieurs noms de paramètres peuvent être transmis et séparés par des virgules.
Supprimer le paramètre de session	UACIPostRemoveParameter. Noms séparés par des virgules des paramètres que vous	Ce paramètre est utile lorsque vous souhaitez supprimer certains paramètres qui ne sont pas requis en vue d'un traitement ultérieur. Plusieurs noms de paramètres peu-

Fonction	Paramètre	La description
après l'appel de l'API (Périmètre-Session)	souhaitez supprimer de la session après l'exécution de l'appel d'API.	vent être transmis et séparés par des virgules.
Exécution synchrone du diagramme (Périmètre-Session)	UACIWaitForSegmentation	<p>Si vous définissez ce paramètre pour une méthode déclenchant la segmentation (startSession, setAudience, ou un postEvent qui déclenche la resegmentation), les diagrammes du niveau d'audience en cours sont exécutés de manière synchrone. Le délai d'attente pour cette exécution synchrone est défini par la configuration de la plate-forme au chemin <code>Affinium interact offerserving segmentationMaxWaitTimeInMS</code></p> <p><b>Exemple :</b> <code>UACIWaitForSegmentation true string</code></p>
Récupérer les offres en cache (Périmètre-Invocation)	UACICachedOffers	<p>Si vous définissez ce paramètre pour l'appel d'API getOffers, les offres de l'appel getOffers précédent sont renvoyées sans exécution de la logique d'arbitrage. Si la liste d'offres précédente est vide, l'arbitrage des offres a lieu et un nouvel ensemble d'offres est renvoyé. La fonctionnalité de mise en cache des offres est activée lorsque la configuration de la plate-forme au chemin <code>Affinium interact offerserving enableOfferCaching</code> est définie sur True. <b>Exemple :</b> <code>UACICachedOffers true string</code></p>

Fonction	Paramètre	La description
Inclure les offres des ID d'audience associés (Périmètre-Session)	Supplément UACISAudience	Les ID d'audience supplémentaires dont les offres éligibles doivent être incluses lors de l'obtention d'offres pour l'ID d'audience de la session. Il a le format <code>audienceLevel1,audienceId1Field1=value1[,audienceId1Field N =value N ]</code> . Plusieurs occurrences de ce paramètre peuvent être utilisées si plusieurs ID d'audience sont nécessaires pour obtenir des offres.
Sessions connexes (Périmètre-Invocation)	UACIEmbeddedSession	Spécifie s'il faut exécuter ce lot dans une session distincte. 1 - Oui, 0 - Non. Il n'est effectif que dans <code>startSession</code> et <code>setAudience</code> lorsque cette demande d'API est la première commande d'un appel <code>executeBatch</code> .
Sessions connexes (Périmètre-Invocation)	UACIIncludeArbitration	Spécifie s'il faut inclure le résumé de l'arbitrage des offres dans la réponse des requêtes <code>getOffers</code> et <code>getOffersForMultipleInteractionPoints</code> . Si l'historisation de l'API est activée, les mêmes informations sont également enregistrées avec celle-ci. 1 - Oui, 0 - Non.
Suivi des contacts (Périmètre-Invocation)	UACIContactStatusCode	Le paramètre est une chaîne représentant un code de type de contact. La valeur du paramètre doit être une entrée valide dans la table <code>UA_ContactStatus</code> .
Suivi des réponses (Périmètre-Invocation)	UACIResponseTypeCode	Valeur représentant un code de type de réponse. La valeur doit être une entrée valide dans la table <code>UA_UsrResponseType</code> .

Fonction	Paramètre	La description
(Périmètre-Session)	<code>UACIPurgePriorWhiteListOnLoad</code>	<p>Ce paramètre, lorsqu'il est utilisé lors de l'appel de la méthode <code>setAudience</code>, recharge le tableau de la liste blanche dans une session existante. Si vous définissez <code>UACIPurgePriorWhiteListOnLoad = 1</code>, le contenu précédemment chargé de la liste blanche sera supprimé de cette session.</p>
(Périmètre-Session)	<code>UACIPurgePriorBlackListOnLoad</code>	<p>Ce paramètre, lorsqu'il est utilisé lors de l'appel de la méthode <code>setAudience</code>, recharge la table de liste noire dans une session existante. Si vous définissez <code>UACIPurgePriorWhiteListOnLoad = 1</code>, le contenu précédemment chargé de la liste blanche sera supprimé de cette session.</p>
Offre Déduplication (Périmètre-Session)	<code>UACINoAttributeDedupeIfFewerOffer</code>	<p>Lorsque vous incluez <code>UACIOfferDedupeAttribute</code> dans l'appel <code>startSession</code>, vous pouvez également définir le paramètre <code>UACINoAttributeDedupeIfFewerOffer</code> pour spécifier le comportement dans les cas où la liste d'offres après déduplication ne contient plus suffisamment d'offres pour satisfaire la demande d'origine.</p> <p>Par exemple, si vous définissez <code>UACIOfferDedupeAttribute</code> pour utiliser la catégorie d'offres pour dédupliquer les offres et que cet appel <code>getOffers</code> demande que huit offres soient renvoyées, la déduplication peut entraîner moins de huit offres éligibles. Dans ce cas, la définition du paramètre <code>UACINoAttributeDedupeIfFewerOffer</code> sur <code>true</code> en-</p>

Fonction	Paramètre	La description
		<p>traînerait l'ajout de certaines des offres dupliquées à la liste éligible pour satisfaire le nombre d'offres demandé.</p> <p><code>UACINoAttributeDedupeIfFewerOffer</code> est défini sur true par défaut.</p>
Suppression d'offre (Périmètre-Session)	<code>UACIignoreBlackList</code>	<p>VRAI - Lorsque nous transmettons ce paramètre comme vrai, le contenu du tableau de la liste noire ne sera pas utilisé pour supprimer les offres autrement éligibles.</p> <p>FAUX - Lorsque cela est transmis comme faux, toutes les offres disponibles dans le tableau de la liste noire ne seront pas affichées/renvoyées à l'utilisateur. Il s'agit du comportement par défaut.</p>
Suppression d'offre (Périmètre-Session)	<code>UACIignoreSuppression-Rules</code>	<p>VRAI - Lorsque nous transmettons ce paramètre à vrai, toutes les règles de suppression d'offre en temps réel seront ignorées lors de l'arbitrage de l'offre.</p> <p>FALSE – Lorsque nous définissons ce paramètre sur false, alors toutes les offres supprimées en temps réel ne seront pas affichées/retournées conformément à la règle. Il s'agit du comportement par défaut.</p>
Filtrage des offres (Périmètre-Invocation)	<code>UACIEnableOfferMappingFilter</code>	Ce paramètre est défini avec le nom du filtre lors de l'exécution pendant <code>getOffers</code> . Le filtre particulier sera appliqué pour obtenir des offres à partir de la table de mappage <code>FlexOffers</code> .

Fonction	Paramètre	La description
Filtrage des offres (Périmètre-Invocation)	<code>UACIDisableOfferMap-pingFilter</code>	Ce paramètre est défini avec le nom du filtre lors de l'exécution pendant <code>getOffers</code> . Le filtre particulier ne sera pas appliqué lors de l'obtention d'offres de la table
États de modèle d'événement (Périmètre-Session)	<code>UACIMergePatternStates</code>	<p>Si ce paramètre est défini avec une valeur de 1, lorsque l'ID d'audience d'une session passe d'anonyme (introuvable dans le profil) à connu (existe dans le profil), les activités d'événement survenues dans la session en cours seront conservées et fusionnées aux activités existantes de l'identifiant d'audience connu.</p> <p>Si sa valeur est définie sur 0 (valeur par défaut), les activités d'événement de l'ID d'audience anonymout seront supprimées.</p> <p>Ce paramètre, lorsqu'il est spécifié, remplace le paramètre de configuration <code>Affinium interact services eventPattern:mergeUnknownUserInSessionStates</code>.</p>
États de modèle d'événement (Périmètre-Session)	<code>UACISavePatternStates</code>	<p>Lorsque ce paramètre est défini avec une valeur de 1, si l'identifiant d'audience d'une session est anonyme (non trouvé dans le profil), les états des modèles d'événements mis à jour dans la session en cours seront enregistrés à la fin de cette session.</p> <p>Lorsque ce paramètre est défini avec une valeur de 0 (valeur par défaut), si l'ID d'audience d'une session est anonyme (introuvable dans le profil), les états des modèles</p>

Fonction	Paramètre	La description
		<p>d'événements mis à jour dans la session en cours seront supprimés à la fin de cette session.</p> <p>Ce paramètre, lorsqu'il est spécifié, remplace le paramètre de configuration suivant <code>Affinium interact services eventPattern:persistUnknownUserStates</code>.</p>
États de modèle d'événement (Périmètre-Session)	UACIShowPatternStates	Pour inclure les états des modèles d'événement appartenant à l'ID d'audience actuel dans la réponse de l'appel d'API <code>postEvent</code> .
Filtrage de profil (Périmètre-Session)	UACIProfileFields	Chaîne séparée par des barres verticales ( ), chaque composant étant un nom de champ de profil. Si ce paramètre est passé et n'est pas vide, seules les valeurs des champs spécifiés seront renvoyées lors de l'appel de la méthode <code>getProfile</code> .
Filtrage de profil (Périmètre-Session)	UACIGetProfileShowDimensionalFields	Si ce paramètre est passé avec une valeur égale à 1, les champs dimensionnels (généralement les champs des tables de profil dimensionnel) seront renvoyés lors de l'appel <code>getProfile</code> . Sinon, seuls les champs de niveau supérieur sont renvoyés. La valeur par défaut est 0.
Filtrage des offres	UACIQueryOffersBySQL	Pour détecter si le moteur doit ou non exécuter la logique <code>offersBySQL</code> lors de l'appel <code>getOffers</code> . Pour l'activer, définissez <code>Affini-</code>



Fonction	Paramètre	La description
(Périmètre-Session)		um interact profile Niveaux d'audience Client Offres par SQL brut enableOffersByRawSQL sur true.
Filtrage des offres (Périmètre-Session)	UACIOffersBySQLTemplate	Pour récupérer le SQL de l'offre à partir de la configuration ( Affinium interact profile Audience Levels Customer Offers By Raw SQL <SQL Template name> ). Le SQL référencé par ce paramètre remplace le SQL par défaut ( Affinium interact profile Audience Levels  Customer Offers By Raw SQL defaultSQLTemplate ) spécifié dans la configuration de la fonctionnalité Offer By Raw SQL.
Attributs de l'offre (sortie)	UACICellCode	Si un attribut d'offre contient un attribut correspondant à " UACICellCode ", le code de cellule pour le traitement sera renvoyé comme valeur pour cet attribut d'offre lors d'un appel getOffers.
Attributs de l'offre (sortie)	UACICellName	Si un attribut d'offre contient un attribut correspondant à " UACICellName ", le nom de cellule pour le traitement sera renvoyé comme valeur pour cet attribut d'offre lors d'un appel getOffers.
Attributs de l'offre (sortie)	UACIZoneID	Si un attribut d'offre contient un attribut correspondant à " UACIZoneID ", l'ID de zone pour le traitement sera renvoyé comme valeur pour cet attribut d'offre lors d'un appel getOffers.

Fonction	Paramètre	La description
Attributs de l'offre (sortie)	<code>UACISegmentID</code>	Si un attribut d'offre contient un attribut correspondant à " <code>UACISegmentID</code> ", l'ID de segment pour le traitement sera renvoyé comme valeur pour cet attribut d'offre lors d'un appel <code>getOffers</code> .
Attributs de l'offre (sortie)	<code>UACIOfferID</code>	Si un attribut d'offre contient un attribut correspondant à " <code>UACIOfferID</code> ", l'ID d'offre pour le traitement sera renvoyé comme valeur pour cet attribut d'offre lors d'un appel <code>getOffers</code> .
Attributs de l'offre (sortie)	<code>UACIOfferListID</code>	Si l'offre de retour est sélectionnée dans une liste d'offres configurée dans une règle, cet attribut est inclus dans la réponse pour indiquer l'ID de la liste d'offres pour le traitement.
Attributs d'offre (sortie)	<code>UACIABTestBranchName</code>	Si l'offre renvoyée est le résultat d'un test A/B, cet attribut est inclus dans la réponse pour indiquer le nom de la branche de test A/B sélectionnée.

## Paramètres réservés à 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.

- `UACIResponseTypeCode`
- `UACIContactStatusCode`
- `UACILogToLearning`
- `UACIOfferTrackingCode`
- `UACIOfferTrackingCodeType`
- `ResetEventPattern`

- `PatternToReset`
- `ResetOfferSuppression`
- `OfferToResetSuppression`
- `UACICustomLoggerTableName`
- `UACIExecuteFlowchartByName`
- `UACIOffersBySQLTemplate`
- `UACIQueryOffersBySQL`
- `UACIGetProfileShowDimFields`
- `UACIProfileFields`
- `UACIOffreFields`
- `UACIExcludeOfferFields`
- `UACIPurgePriorBlackListOnLoad`
- `UACIPurgePriorWhiteListOnLoad`
- `UACIMergePatternStatesUACIMergePatternStates`
- `UACIResetMergedPatternStatesUACIResetMergedPatternStates`
- `UACISavePatternStates`
- `UACIShowPatternStates`
- `UACIOfferDedupeAttributeUACIOfferDedupeAttribute`
- `UACINoAttributeDedupeIfFewerOffer`
- `UACIIgnoreBlackList`
- `UACIIgnoreSuppressionRules`
- `UACILogSeparationFileName`
- `UACICachedOffers`
- `UACIWaitForSegmentation`
- `UACIPreRemoveParameterUACIPreRemoveParameter`
- `UACIPostRemoveParameter`
- `UACIInteractiveChannelName`
- `UACIInteractiveChannelID`
- `UACIInteractionPointName`
- `UACIInteractionPointID`
- `UACIEventName`
- `UACIEventID`

- `UACISessionID`
- `UACIEnableOfferMappingFilter`
- `UACIDisableOfferMappingFilter`
- `UACICellCode`
- `UACICellName`
- `UACIOfferID`
- `UACISegmentID`
- `UACIZoneID`
- `UACIABTestBranchName`
- `UACIEmbeddedSession`
- `UACIEventTime`
- `UACIIncludeArbitration`
- `UACIIncludeAttributeMetadata`

## 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 Unica Interact renseigne l'objet de message de recommandation. L'objet de message de recommandation contient les attributs suivants :

- **etailMessage** - Description prolixue 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

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.

```
getDetailMessage()
```

### 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

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

```
getMessage()
```

### 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

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`).

```
getMessageCode()
```

## 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

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

```
getStatusLevel()
```

### 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

Vous extrayez les codes du message de recommandation avec la méthode `getMessageCode`.

Cette table répertorie et décrit les codes des messages de recommandation.

Code	Texte du message	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 <code>endSession</code> .
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.



Code	Texte du message	Description
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 appel <code>executeBatch</code> est non 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 l'audience, 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.

<b>Code</b>	<b>Texte du message</b>	<b>Description</b>
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	Une erreur s'est produite 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	Une erreur s'est produite pendant la tentative d'exécution du diagramme.
20	FLOWCHART_FAILED	Une exécution de diagramme a échoué.
21	FLOWCHART_ABORTED	Une exécution de diagramme a été abandonnée.
22	FLOWCHART_NEVER_RUN	Un diagramme spécifié n'a jamais été exécuté.
23	FLOWCHART_STILL_RUNNING	Un diagramme est toujours en cours d'exécution.
24	ERROR_WHILE_READING_PARAMETERS	Une erreur s'est produite 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	Une erreur s'est produite lors de la journalisation des statistiques de texte par défaut (nombre d'affichages de la

Code	Texte du message	Description
		chaîne par défaut du point d'interaction).
27	SCORE_OVERRIDE_LOAD_FAILED	Le chargement de la table de substitution de score a échoué.
28	NULL_AUDIENCE_ID	L'identificateur d'audience est vide.
29	UNRECOGNIZED_AUDIENCE_LEVEL	Un niveau d'audience non reconnu a été spécifié.
30	MISSING_AUDIENCE_FIELD	Une zone d'audience est manquante.
31	INVALID_AUDIENCE_FIELD_TYPE	Un type de zone d'audience non valide a été spécifié.
32	UNSUPPORTED_AUDIENCE_FIELD_TYPE	Type de zone d'audience non prise en charge
33	TIMEOUT_REACHED_ON_GET_OFFERS_CALL	L'appel <code>getOffers</code> a expiré sans renvoyer d'offres.
34	INTERACT_INITIALIZATION_NOT_COMPLETED_SUCCESSFULLY	L'initialisation du serveur d'exécution n'a pas abouti.
35	SESSION_ID_UNDEFINED	L'identificateur de session est non défini.
36	INVALID_NUMBER_OF_OFFERS_REQUESTED	Nombre d'offres demandées non valide.
37	NO_SESSION_EXIST_BUT_WILL_CREATE_NEW_ONE	Il n'existait aucune session, mais une session a été créée.
38	AUDIENCE_ID_NOT_FOUND_IN_PROFILE_TABLE	L'identificateur d'audience spécifié ne figure pas dans la table de profil.
39	LOG_CUSTOM_LOGGER_EVENT_EXCEPTION	Une exception s'est produite lors de la tentative de soumission de l'événement

Code	Texte du message	Description
		des données de journalisation personnalisées.
40	SPECIFIED_FLOWCHART_FOR_EXECUTION_DOES_NOT_EXIST	Le diagramme spécifié ne peut pas être exécuté car il n'existe pas.
41	AUDIENCE_NOT_DEFINED_IN_CONFIGURATION	L'audience spécifiée n'est pas définie dans la configuration.
42	Impossible de charger les offres à partir d'un SQL brut	Le SQL brut ne parvient pas à charger des offres.
43	Stratégie de duplication non valide	La stratégie de duplication pour l'id de session n'est pas valide.
44	Erreur lors de la sauvegarde des états de modèle d'événement pour l'ID d'audience	Une erreur s'est produite lors de la sauvegarde des états de modèle d'événement pour l'ID d'audience.
45	Expression incorrecte	L'expression n'est pas valide.
46	Expression vide	L'expression est vide.
47	Erreur lors d'une tentative d'évaluation d'un modèle d'événement	Une erreur s'est produite lors d'une tentative d'évaluation d'un modèle d'événement.
48	Erreur lors du chargement de l'état de modèle d'événement pour l'ID d'audience	Une erreur s'est produite lors du chargement de l'état de modèle d'événement pour l'ID d'audience.
49	Erreur lors du chargement/de la mise à jour de la session	Une erreur s'est produite lors du chargement ou de la mise à jour de la session.
50	Erreur lors du chargement/de la mise à jour de l'état de modèle pour l'ID audience	Une erreur s'est produite lors du chargement ou de la mise à jour de l'état

<b>Code</b>	<b>Texte du message</b>	<b>Description</b>
		de modèle d'événement pour l'ID d'audience.
51	Erreur lors de la création du service de phase d'exécution OpDetection	Une erreur s'est produite lors de la création du service de phase d'exécution OpDetection.
52	Erreur lors de l'extraction des états de modèle d'événement les plus récents à partir de OpDetection pour IDSession	Une erreur s'est produite lors de l'extraction des états de modèle d'événement les plus récents à partir de OpDetection pour IDSession.
53	Erreur lors de l'affichage de l'événement sur OpDetection pour IDSession	Une erreur s'est produite lors de l'affichage de l'événement sur OpDetection pour IDSession.
54	Erreur lors du contact du service OpDetection pour le serveur	Une erreur s'est produite lors du contact du service OpDetection pour le serveur.
55	Le service OpDetection ne sera pas contacté durant les {0} prochaines minutes.	Le service OpDetection ne peut pas être contacté durant les {0} prochaines minutes.
56	Paramètre(s) requis manquant(s) {0} pour l'événement {1}.	Le(s) paramètre(s) requis {0} est/sont manquant(s) pour l'événement {1}.
57	L'événement est configuré pour se connecter comme contact, mais aucune offre n'a été recommandée	L'événement est configuré pour se connecter comme contact, mais aucune offre n'a été recommandée.
58	Connexion au contact demandé, mais treatmentCode non valide	La connexion au contact est demandée, mais treatmentcode n'est pas valide pour sessionCode.

<b>Code</b>	<b>Texte du message</b>	<b>Description</b>
59	Impossible d'obtenir les réponses par lot pour l'exécution du simulateur de scénario avancé. Voir le journal sur le serveur pour plus de détails.	Impossible d'obtenir les réponses par lot pour l'exécution du simulateur de scénario avancé. Voir le journal sur le serveur pour plus de détails.
60	Connexion à la réponse demandée mais code de suivi non valide pour sessionId	La connexion à la réponse est demandée, mais code de suivi n'est pas valide pour sessionId.
61	Connexion au contact demandé, mais code d'état du contact non valide pour sessionId : {0} code de statut du contact : {1}	La connexion au contact est demandée, mais le code de statut du contact n'est pas valide pour sessionId.
62	Connexion à la réponse demandée, mais code de type de réponse non valide pour sessionId : {0} code de type de réponse : {1}	connexion à la réponse demandée, mais code de type de réponse non valide pour sessionId.
63	Echec de l'authentification. Le jeton n'est pas disponible.	L'authentification a échoué, car le jeton n'est pas disponible.
64	Echec de l'authentification. L'ID de session n'est pas disponible.	L'authentification a échoué, car l'ID de session n'est pas disponible.
65	Echec de l'authentification. Le jeton n'est pas associé à l'ID de session demandé.	L'authentification a échoué, car le jeton n'est pas associé à l'ID de session demandé.
66	Demande non autorisée. Jeton ou données d'identification non valides.	La demande n'a pas été autorisée, car le jeton ou les données d'identification ne sont pas valides.
67	Erreur lors de la génération du jeton.	Une erreur s'est produite lors de la génération du jeton.

<b>Code</b>	<b>Texte du message</b>	<b>Description</b>
68	Erreur : Impossible d'obtenir les réponses par lot pour l'exécution du simulateur de scénario de base. Voir le journal sur le serveur pour plus de détails.	Impossible d'obtenir les réponses par lot pour l'exécution du simulateur de scénario de base.
69	Aucune licence valide	La licence n'est pas valide.
70	Impossible de créer ou de rechercher une session	Impossible de créer ou de rechercher une session
71	Échec du diagramme pour l'ID de session : {0}	Le diagramme a échoué pour la session.
72	Dépassement du délai d'exécution du diagramme pour l'ID de session : {0}	Le délai d'exécution du diagramme a été dépassé pour la session.
73	Echec de l'API de Campaign permettant d'obtenir les segments statiques pour l'ID de session : {0}	L'API de Campaign permettant d'obtenir les segments statiques a échoué pour la session.
74	Dépassement du délai pour l'API de Campaign permettant d'obtenir les segments statiques pour l'ID de session : {0}	Le délai de l'API de Campaign permettant d'obtenir les segments statiques a été dépassé pour la session.
101	Réponse à la commande	Réponse à la commande
102	Code de statut	Code de statut
103	Version d'API	Version d'API
104	ID de session	ID de session
105	Message de recommandation	Message de recommandation
106	Code du message	Code du message
107	Message	Message

<b>Code</b>	<b>Texte du message</b>	<b>Description</b>
108	Message détaillé	Message détaillé
109	Niveau du statut	Niveau du statut
110	Aucun message de recommandation	Aucun message de recommandation
111	Aucun enregistrement de profil	Aucun enregistrement de profil
112	Offres du point d'interaction	Offres du point d'interaction
113	Texte par défaut	Texte par défaut
114	Offre recommandée	Offre recommandée
123	Score de l'offre	Score de l'offre
116	Nom de l'offre	Nom de l'offre
117	Description de l'offre	Description de l'offre
118	Code de l'offre	Code de l'offre
119	Paramètres de l'offre	Paramètres de l'offre120
120	Code de traitement	Code de traitement
121	Aucune offre recommandée	Aucune offre recommandée
122	Aucune liste d'offres	Aucune liste d'offres
123	Nom	Nom
130	Type	Type
125	Valeur de date	Valeur de date
126	Valeur de chaîne	Valeur de chaîne
170	Valeur numérique	Valeur numérique
128	Destinataire valide et/ou passerelle valide introuvable(s)	Destinataire valide et/ou passerelle valide introuvable(s)



<b>Code</b>	<b>Texte du message</b>	<b>Description</b>
129	Destinataire non valide : {0}	Le destinataire n'est pas valide.
130	Passerelle non valide : {0}	La passerelle n'est pas valide.
131	L'envoi du message au destinataire {0} pour la passerelle {1} a abouti	Le message envoyé au destinataire pour la passerelle a abouti.
193	Aucun destinataire valide disponible	Il n'y a pas de destinataire valide disponible.
133	Format du paramètre non valide - {0}	Le format du paramètre n'est pas valide.
134	Format de l'ID d'audience non valide	Le format de l'ID d'audience n'est pas valide.
135	Paramètre non valide {0}, valeur non valide {1}	Le paramètre et sa valeur ne sont pas valides.
136	ID d'audience non valide {0}, valeur non valide {1}	L'ID d'audience et sa valeur ne sont pas valides.
137	Paramètre non valide {0}, type de données non valide {1}	Le paramètre et son type de données ne sont pas valides.
138	ID d'audience non valide {0}, type de données non valide {1}	L'ID d'audience et son type de données ne sont pas valides.
139	Paramètre non valide {0}, type de données {1} attendu, mais {2} reçu.	Paramètre non valide {0}, type de données {1} attendu, mais {2} reçu.
140	Impossible de trouver l'offre pour réinitialiser les règles de suppression : {0}.	Impossible de trouver l'offre pour réinitialiser les règles de suppression.
141	Impossible de trouver le schéma d'événement à réinitialiser : {0}	Impossible de trouver le schéma d'événement à réinitialiser.
142	Format de la requête non valide, paramètre manquant : {0}	Le format de la requête n'est pas valide et un paramètre est manquant.

Code	Texte du message	Description
201	Les entrées fournies pour l'API ne sont pas valides.	Les entrées fournies pour l'API ne sont pas valides.
202	Une erreur interne s'est produite. Veuillez consulter les fichiers journaux pour plus de détails.	Une erreur interne s'est produite. Voir les journaux pour plus de détails.
203	ID ou nom d'IC non valide : {0}	L'ID ou le nom d'IC n'est pas valide.
204	Erreur lors de la recherche du nom d'IC : {0} dans le cache du déploiement.	Une erreur s'est produite lors de la recherche du nom d'IC : {0} dans le cache du déploiement.
205	Le paramètre d'entrée {0} est manquant.	Le paramètre d'entrée est manquant.
206	Echec de l'authentification	Echec de l'authentification
207	Le nom d'utilisateur est requis	Le nom d'utilisateur est requis.

## 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** - Valeur la plus élevée pour le code de statut pour toutes les réponses demandées par la méthode `executeBatch`.
- **Responses** - Tableau des objets Response demandés par la méthode `executeBatch`.

### getBatchStatusCode

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

```
getBatchStatusCode()
```

## 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

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

```
getResponses()
```

## 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 passer un tableau d'objets qui implémentent l'interface de commande. Vous devez utiliser la mise en oeuvre par défaut, `CommandImpl` pour passer 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	Terminer la session	Aucun.
COMMAND_GETOFFERS	Obtenir des offres	<ul style="list-style-type: none"> <li>• <code>setInteractionPoint</code></li> <li>• <code>setNumberRequested</code></li> </ul>
COMMAND_GETPROFILE	Obtenir le profil	Aucun.
COMMAND_GETVERSION	Obtenir la version	Aucun.
COMMAND_POSTEVENT	<code>postEvent</code>	<ul style="list-style-type: none"> <li>• <code>setEvent</code></li> <li>• <code>setEventParameters</code></li> </ul>
COMMAND_SETAUDIENCE	Définir l'audience	<ul style="list-style-type: none"> <li>• <code>setAudienceID</code></li> <li>• <code>setAudienceLevel</code></li> <li>• <code>setEventParameters</code></li> </ul>
COMMAND_SETDEBUG	Définir le débogage	Définir le débogage

Commande	Méthode API Interact	Méthodes de l'interface de commande
COMMAND_STARTSESSION	Démarrer la session	<ul style="list-style-type: none"> <li>• setAudienceID</li> <li>• setAudienceLevel</li> <li>• Définir le débogage</li> <li>• setEventParameters</li> <li>• setInteractiveChannel</li> <li>• setRelyOnExistingSession</li> </ul>

## setAudienceID

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

```
setAudienceID(audienceID)
```

- **audienceID** - Tableau d'objets `NameValuePair` définissant `AudienceID`.

## Valeur de retour

Aucun.

## 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

La méthode `setAudienceLevel` définit le niveau d'audience pour les commandes `setAudience` et `startSession`.

```
setAudienceLevel(audienceLevel)
```

- `audienceLevel` - Chaîne contenant le niveau d'audience.



**Important** : Le nom de *audienceLevel* doit correspondre exactement au niveau d'audience défini dans Unica Campaign. Il est sensible à la casse.

## Valeur de retour

Aucun.

## 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);
```

## Définir le débogage

La méthode `setDebug` définit le niveau de débogage pour la commande `startSession`.

```
setDebug(debug)
```

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

Aucun.

## 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

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

```
setEvent(event)
```

- **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. Il est sensible à la casse.

## Valeur de retour

Aucun.

## 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

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

```
setEventParameters(eventParameters)
```

- **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

Aucun.

## 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

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

```
setGetOfferRequests(numberRequested)
```

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

## Valeur de retour

Aucun.

## 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 getOffersMultiIPCcmd = new CommandImpl();
getOffersMultiIPCcmd.setGetOfferRequests(new GetOfferRequest[]
{request1,
    request2, request3});

```

## setInteractiveChannel

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

```
setInteractiveChannel(interactiveChannel)
```

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



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

## Valeur de retour

Aucun.

## 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

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

```
setInteractionPoint(interactionPoint)
```

- **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. Il est sensible à la casse.

## Valeur de retour

Aucun.

## 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

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

```
setMethodIdentifier(methodIdentifier)
```

- **methodIdentifier** - Chaîne contenant le type de commande.

Les valeurs valides sont les suivantes :

- **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

Aucun.



## 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

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

```
setNumberRequested(numberRequested)
```

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

## Valeur de retour

Aucun.

## 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

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

```

setRelyOnExistingSession(relyOnExistingSession)

```

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

Aucun.

### 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 Unica Interact renvoient des objets paires nom-valeur ou nécessitent de passer des objets paires nom-valeur en tant qu'arguments. Lors de la transmission en tant qu'arguments dans une méthode, vous devez utiliser la mise en oeuvre par défaut `NameValuePairImpl`.

### getName

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

```
getName()
```

### 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

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

```
getValueAsDate()
```

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

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

```
getValueAsNumeric()
```

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

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

```
getValueAsString()
```

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

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

```
getValueDataType()
```

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.

Les valeurs valides sont les suivantes :

- **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

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

```
setName(name)
```

- **name** - Chaîne contenant le composant nom d'un objet `NameValuePair`.

### Valeur de retour

Aucun.

### Exemple

L'exemple suivant montre comment définir le composant nom d'un objet `NameValuePair`.

```
NameValuePair custId = new NameValuePairImpl();
custId.setName("CustomerId");
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
```

## setValueAsDate

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

```
setValueAsDate(valueAsDate)
```

- **valueAsDate** - Date contenant la valeur de date et d'heure d'un objet `NameValuePair`.

### Valeur de retour

Aucun.

## 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

La méthode `setValueAsNumeric` définit la valeur d'un objet NameValuePair.

```
setValueAsNumeric(valueAsNumeric)
```

- **valueAsNumeric** - Double contenant la valeur numérique d'un objet NameValuePair.

## Valeur de retour

Aucun.

## 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

La méthode `setValueAsString` définit la valeur d'un objet NameValuePair.



```
setValueAsString(valueAsString)
```

- **valueAsString** - Chaîne contenant la valeur d'un objet NameValuePair

## Valeur de retour

Aucun.

## 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

La méthode `setValueDataType` définit le type de données d'un objet NameValuePair.

```
getValueDataType(valueDataType)
```

Les valeurs valides sont les suivantes :

- **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

Aucun.

## 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);
```

## setScope (portée)

La méthode `setScope` définit la durée de vie de cet objet `NameValuePair`.

Les valeurs admises sont les suivantes.

- `Scope.INVOCATION` : L'objet `NameValuePair` est effectif lors du traitement de cet appel d'API. Sa valeur n'est pas enregistrée dans la session.
- `Scope.SESSION` : L'objet `NameValuePair` est effectif dès le début de cet appel d'API jusqu'à sa suppression. Sa valeur est enregistrée dans la session. Il s'agit de la valeur par défaut.

## Valeur de retour

Aucun

## getScope()

La méthode `getScope` permet d'obtenir la portée de la méthode `NameValuePair`.

Les valeurs admises sont les suivantes.

- `Scope.INVOCATION` : L'objet `NameValuePair` est effectif lors du traitement de cet appel d'API. Sa valeur n'est pas enregistrée dans la session.
- `Scope.SESSION` : L'objet `NameValuePair` est effectif dès le début de cet appel d'API jusqu'à sa suppression. Sa valeur est enregistrée dans la session.

### Valeur de retour

La méthode `getScope` renvoie une valeur d'énumération de la portée.

## A propos de la classe Offer

La classe `Offer` contient des méthodes qui définissent un objet `Offer`. Cet objet `Offer` contient un grand nombre de propriétés d'une offre dans Unica Campaign.

L'objet `Offer` contient les attributs suivants :

- **AdditionalAttributes** - `NameValuePairs` contenant les attributs d'offre personnalisée que vous avez définis dans Unica 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 score défini par `ScoreOverrideTable` si la propriété `enableScoreOverrideLookup` a la valeur `true`.

## getAdditionalAttributes

La méthode `getAdditionalAttributes` renvoie les attributs d'offre personnalisés définis dans Unica Campaign.

```
getAdditionalAttributes()
```

### Valeur de retour

La méthode `getAdditionalAttributes` renvoie un tableau d'objets paires nom-valeur.

### 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

La méthode `getDescription` renvoie la description de l'offre définie dans Unica Campaign.

```
getDescription()
```

## 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

La méthode `getOfferCode` renvoie le code de l'offre comme défini dans Unica Campaign.

```
getOfferCode()
```

## 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

La méthode `getOfferName` renvoie le nom de l'offre tel qu'il est défini dans Unica Campaign.

```
getOfferName()
```

## 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

La méthode `getScore` renvoie un score, basé sur les offres que vous avez configurées.

```
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'auto-apprentissage, cette méthode renvoie le score marketing de l'offre, tel qu'il est défini dans l'onglet Stratégie d'interaction.
- Si vous avez activé les offres par défaut ou la table de substitution de score, mais non l'auto-apprentissage, 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 spécialiste du marketing et la table de substitution de score.
- Si vous avez activé l'auto-apprentissage, cette méthode renvoie le score final utilisé par l'auto-apprentissage pour commander 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

La méthode `getTreatmentCode` renvoie le code de traitement l'offre comme défini dans Unica Campaign.

```
getTreatmentCode()
```

Etant donné que Unica 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 Unica Campaign.

## getDefaultString

La méthode `getDefaultString` renvoie la chaîne par défaut du point d'interaction, comme défini dans Unica Campaign.

```
getDefaultString()
```

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. Unica 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 aucune 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

La méthode `getRecommendedOffers` renvoie un tableau d'objets `Offer` demandés par la méthode `getOffers`.

```
getRecommendedOffers()
```

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 de NameValuePairs contenant des 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

La méthode `getAdvisoryMessages` renvoie un tableau de messages de recommandation à partir de l'objet de réponse.

```
getAdvisoryMessages()
```

### 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

La méthode `getApiVersion` renvoie la version de l'API d'un objet de réponse.

```
getApiVersion()
```

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

La méthode `getOfferList` renvoie l'objet `OfferList` d'un objet de réponse.

```
getOfferList()
```

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

La méthode `getAllOfferLists` renvoie un tableau de tous les Offerlists d'un objet de réponse.

```
getAllOfferLists()
```

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

La méthode `getProfileRecord` renvoie les enregistrements de profil de la session en cours sous la forme d'un tableau d'objets paires nom-valeur. Ces enregistrements de profil incluent également les `eventParameters` ajouté plus tôt au cours de la session d'exécution.

```
getProfileRecord()
```

La méthode `getProfile` renseigne l'enregistrement du profil des objets paires nom-valeur d'un objet de réponse.

### Valeur de retour

L'objet de réponse renvoie un tableau d'objets paires nom-valeur.

### 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

La méthode `getSessionID` renvoie l'ID de session.

```
getSessionID()
```

### 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

La méthode `getStatusCode` renvoie la code de statut d'un objet de réponse.

```
getStatusCode()
```

## 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 9. Classes et méthodes de l'API JavaScript d'Unica Interact

Les sections suivantes contiennent les conditions préalables et les informations que vous devez connaître avant d'utiliser l'API JavaScript Unica Interact.

L'API Unica Interact prend en charge un flavor javascript pour autoriser les communications entre le client utilisateur final (navigateur) et le serveur.



**Remarque :** Cette section suppose que vous soyez familiarisé les API JavaScript.



**Remarque :** Les occurrences multiples de tout paramètre dans un même appel d'API ne sont pas prises en charge.

## Conditions requises pour JavaScript

Avant d'utiliser l'API JavaScript d'Unica Interact sur un site Web, vous devez intégrer le fichier `interactapi.js` à vos pages Web.

## 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 surensemble 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 Unica 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 passer un paramètre d'événement avec le type personnalisé `NameValuePair`. Dans la classe `NameValuePair`, vous pouvez utiliser les méthodes `setName` et `setValueAsNumeric` pour changer 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 d'instantané dans votre diagramme temps réel pour écrire les données dans une table de base de données. Si vous envisagez d'utiliser des processus d'instantané, 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.

## Utilisation du paramètre de rappel

La fonction de rappel est un paramètre supplémentaire des méthodes de l'API JavaScript d'Unica Interact.

Le processus principal du navigateur est une boucle d'événements à une seule unité d'exécution. Une opération longue exécutée dans ce type de boucle bloque le processus. Celui-ci attend la fin de l'opération pour traiter d'autres événements. Pour éviter les blocages dus aux opérations longues, l'objet XMLHttpRequest fournit une interface asynchrone. Vous lui transmettez un rappel à exécuter à la fin de l'opération, et pendant qu'il le traite, il rend la main à la boucle d'événements principale au lieu de bloquer le processus.

Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

Ainsi, pour afficher des offres sur votre page Web, vous pourriez utiliser la méthode `getOffers`, et le rappel pour l'affichage sur la page. La page Web fonctionne normalement et n'attend pas qu'Unica Interact renvoie les offres. Lorsqu'Unica Interact le fait, la réponse est renvoyée dans la paramètre de rappel. Vous pouvez analyser les données du rappel et afficher les offres sur la page.

Vous pouvez utiliser un rappel générique pour toutes les fonctions, ou des rappels spécifiques pour des fonctions particulières.

Vous pouvez utiliser `var callback = InteractAPI.Callback.create(onSuccess, onError);` pour créer une fonction de rappel générique.

Vous pouvez utiliser la fonction suivante pour créer une fonction de rappel spécifique pour la méthode `getOffers`.

```
var callbackforGetOffer = InteractAPI.Callback.create(onSuccessofGetOffer,
onErrorofGetOffer);
```

## 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 Unica Interact prend en charge les méthodes de cette classe.

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

## Démarrer la session

La méthode `startSession` crée et définit une session d'exécution.

```
function callStartSession(commandsToExecute, callback) {  
  
    //read configured start session  
    var ssId = document.getElementById('ss_sessionId').value;  
    var icName = document.getElementById('ic').value;  
    var audId = document.getElementById('audienceId').value;  
    var audLevel = document.getElementById('audienceLevel').value;  
    var params = document.getElementById('ss_parameters').value;  
    var relyOldSs = document.getElementById('relyOnOldSession').value;  
    var debug = document.getElementById('ss_isDebug').value;  
  
    InteractAPI.startSession(ssId, icName,  
                             getNameValuePairs(audId), audLevel,  
                             getNameValuePairs(params), relyOldSs,  
                             debug, callback) ;  
  
}
```

`startSession` peut déclencher cinq actions au maximum :

- Créer une session d'exécution.
- Charger les données de profil du visiteur correspondant au niveau d'audience 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 temps réel correspondant au niveau d'audience 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.

- **elyOnExistingSession** - Valeur booléenne qui définit si cette session utilise une session nouvelle ou existante. Les valeurs valides 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, toutes 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 valides sont `true` ou `false`. Si elle est `true`, Unica 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 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 Unica Campaign.
- **audienceID** - Tableau d'objets NameValuePairImpl dans lequel les noms doivent correspondre aux noms de colonne physique de toute table contenant l'ID audience.
- **audienceLevel** - Chaîne définissant le niveau d'audience.
- **parameters** - Objets NameValuePairImpl identifiant tous les paramètres à transmettre avec `startSession`. Ces valeurs sont stockées dans les données de session et peuvent être utilisées pour la segmentation.

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

- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

Si `audienceID` ou `audienceLevel` ne sont pas valides et si `relyOnExistingSession` a la valeur `false`, l'appel `startSession` échoue. Si le `interactiveChannel` est non 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, Unica Interact crée une nouvelle session.

Si `relyOnExistingSession` a la valeur `true` 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`
- ID session
- `StatusCode`

## Dédoublonnage des offres dans les attributs d'offre

Grâce à l'API Unica Interact, deux appels API fournissent les offres : `getOffers` et `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` peut empêcher le retour des offres en double au niveau d'`OfferID`, mais ne peut pas dédoubler les offres dans la catégorie d'offre. Ainsi, pour qu'Unica Interact ne renvoie qu'une seule offre de chaque catégorie d'offre, vous devez auparavant faire appel à une solution de contournement. Grâce à l'introduction de deux nouveaux paramètres dans l'appel API `startSession`, il est désormais possible de dédoubler les offres dans les attributs d'offre, comme la catégorie.

Cette liste récapitule les paramètres qui ont été ajoutés dans l'appel API `startSession`. Pour plus d'informations sur ces paramètres ou sur tout aspect de l'API Unica Interact, reportez-vous au document `Unica Interact - Guide d'administration` ou aux fichiers Javadoc inclus avec votre installation d'Unica Interact, dans `<Unica Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Pour créer un appel API `startSession` avec un dédoublonnage des offres, afin que les appels `getOffer` suivants ne renvoient qu'une seule offre de chaque catégorie, vous devez inclure le paramètre



`UACIOfferDedupeAttribute` dans l'appel API. Vous pouvez indiquer un paramètre au format `name, value, type`, comme indiqué ici :

```
UACIOfferDedupeAttribute, <attributeName>, string
```

Dans cet exemple, vous remplacez `<attributeName>` par le nom de l'attribut d'offre que vous souhaitez utiliser comme critère pour le dédoublement, comme `Category`.



**Remarque :** Unica Interact examine les offres dotées de la même valeur d'attribut indiquée (par exemple `Category`) et effectue le dédoublement pour tout supprimer sauf l'offre dotée du score le plus élevé. Si les offres dotées de l'attribut en double ont aussi des scores identiques, Unica Interact renvoie une sélection aléatoire parmi les offres correspondantes.

- `UACINoAttributeDedupeIfFewerOffer`. Lorsque vous incluez `UACIOfferDedupeAttribute` dans l'appel `startSession`, vous pouvez aussi définir ce paramètre `UACINoAttributeDedupeIfFewerOffer` afin d'indiquer le comportement lorsque la liste des offres après dédoublement ne contient plus assez d'offres pour satisfaire la demande d'origine.

Par exemple, si vous avez défini `UACIOfferDedupeAttribute` de telle sorte qu'il utilise la catégorie d'offre afin de dédoubler les offres, et que l'appel `getOffers` suivant demande le renvoi de huit offres, le dédoublement peut renvoyer un nombre d'offres éligibles inférieur à 8. Dans ce cas, si vous indiquez la valeur `true` pour le paramètre `UACINoAttributeDedupeIfFewerOffer`, vous ajoutez certaines offres dédoublées dans la liste éligible pour satisfaire au nombre d'offres demandé. Dans cet exemple, si vous attribuez la valeur `false` au paramètre, le nombre d'offres renvoyé est inférieur au nombre demandé.

`UACINoAttributeDedupeIfFewerOffer` est défini par défaut sur `true` par défaut.

Par exemple, supposons que vous indiquez pour le paramètre `startSession` que le critère de dédoublement est la catégorie de l'offre, comme indiqué ici :

```
UACIOfferDedupeAttribute, Category, string;
```

```
UACINoAttributeDedupeIfFewerOffer, 1, string
```

Par défaut, `UACIOfferDedupeAttribute` ne dédouble pas les offres si un nombre d'offres inférieur au nombre demandé est renvoyé. Toutefois, pour que le dédoublement ait lieu lorsqu'un nombre d'offres inférieur au nombre demandé est renvoyé, fournissez le paramètre `UACINoAttributeDedupeIfFewerOffer` et associez-le à la valeur 1.

Ensemble, ces paramètres indiquent à Unica Interact de dédoubler les offres en fonction de l'attribut d'offre "Category," et de ne renvoyer que les offres dédoublées même si le nombre d'offres renvoyées est inférieur au nombre demandé (`UACINoAttributeDedupeIfFewerOffer` a pour valeur false).

Lorsque vous émettez un appel API `getOffers`, l'ensemble d'origine des offres éligibles peut inclure les offres suivantes :

- Category=Electronics : Offre A1 avec un score égal à 100 et Offre A2 avec un score égal à 50.
- Category=Smartphones : Offre B1 avec un score égal à 100, Offre B2 avec un score égal à 80 et Offre B3 avec un score égal à 50.
- Category=MP3Players : Offre C1 avec un score égal à 100, Offre C2 avec un score égal à 50.

Dans ce cas, il y avait deux offres en double qui correspondaient à la première catégorie, trois offres en double qui correspondaient à la deuxième catégorie et deux offres en double qui correspondaient à la troisième catégorie. Les offres renvoyées sont les offres dotées des scores les plus élevés de chaque catégorie, à savoir Offre A1, Offre B1 et Offre C1.

Si l'appel API `getOffers` a demandé six offres, cet exemple a attribué la valeur false à `UACINoAttributeDedupeIfFewerOffer` et seulement trois offres sont renvoyées.

Si l'appel API `getOffers` a demandé six offres et que cet exemple a omis le paramètre `UACINoAttributeDedupeIfFewerOffer`, ou lui a spécifiquement attribué la valeur true, certaines offres en double sont incluses dans le résultat pour satisfaire au nombre demandé.

## postEvent

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

```
function callPostEvent(commandsToExecute, callback) {

    var ssId = document.getElementById('pe_sessionId').value;
    var ev = document.getElementById('event').value;
    var params = document.getElementById('parameters').value;

    InteractAPI.postEvent(ssId, ev, getNameValuePairs(params),
callback);

}
```

- **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 transmettre 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 temps réel 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 profils), vous devez inclure un `eventParameter` 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 temps réel.

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



**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, Unica 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, Unica Interact renvoie une erreur.

- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.
- 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.

La resegmentation n'efface pas les résultats de la segmentation précédente pour le niveau d'audience en cours. Vous pouvez utiliser le paramètre `UACIExecuteFlowchartByName` pour définir des diagrammes spécifiques à exécuter. La méthode `getOffers` attend la fin de la resegmentation avant de s'exécuter. Par conséquent, un retard est possible si vous appelez une méthode `postEvent`, qui déclenche une resegmentation juste avant un appel `getOffers`.

## Valeur de retour

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

- AdvisoryMessages
- ApiVersion
- OfferList
- Profil
- ID session
- StatusCode

## Obtenir des offres

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

```
function callGetOffers(commandsToExecute, callback) {  
  
    var ssId = document.getElementById('go_sessionId').value;  
    var ip = document.getElementById('go_ipoint').value;  
    var nofRequested = 5 ;  
    var nreqString = document.getElementById('offersRequested').value;  
  
    InteractAPI.getOffers(ssId, ip, nofRequested, callback);  
  
}
```

- **session ID** - Chaîne identifiant la session en cours.
- **Interaction point** - 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.

- **nofRequested** - Entier identifiant le nombre d'offres demandées.
- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

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`
- `Profil`
- `ID session`
- `StatusCode`
- `NameValuePair`

Les positions décimales des scores d'offre sont retournées dans la réponse `getOffer` dans la paire `NameValue`. Lorsque les offres sont retournées vers les canaux entrants à l'origine de la requête, les canaux utilisent les scores pour hiérarchiser les offres. Les chiffres décimaux ne sont pas supprimés, ainsi le canal sait quelle offre détient le score le plus élevé si des nombres décimaux sont retournés.

## Obtenir des offres pour des points d'interaction multiples

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

```
function callGetOffersForMultipleInteractionPoints(commandsToExecute,
callback) {
```

```

var ssId = document.getElementById('gop_sessionId').value;
var requestDetailsStr =
document.getElementById('requestDetail').value;

//trim string
var trimmed = requestDetailsStr.replace(/\{/g, "");
var parts = trimmed.split("{}");

//sanitize strings
for(i = 0; i < parts.length; i += 1) {
    parts[i] = parts[i].replace(/^\s+|\s+$/g, "");
}

//build get offer requests
var getOffReqs = [];
for(var i = 0; i < parts.length; i += 1) {
    var getofReqObj = parseGetOfferReq(parts[i]);
    if (getofReqObj) {
        getOffReqs.push(getofReqObj);
    }
}

InteractAPI.getOffersForMultipleInteractionPoints
(ssId, getOffReqs, callback);
}

```

- **session ID** - Chaîne identifiant la session en cours.
- **requestDetailsStr** - Chaîne fournissant un tableau d'objets `GetOfferRequest`.

Chaque objet `GetOfferRequest` spécifie 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 incluses dans cette instance de `GetOfferRequest` (c'est-à-dire que Unica Interact appliquera le dédoublement).
  - **ALLOW\_DUPLICATION** (valeur ID = 2). Toutes les offres répondant aux exigences indiquées dans cette instance de `GetOfferRequest` seront incluses. Les offres qui ont été incluses dans les précédentes instances de `GetOfferRequest` ne seront pas rapprochées.
- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

L'ordre des demandes dans le paramètre de tableau est également l'ordre de préséance 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 Unica 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 dédoublement 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`
- `Profil`
- `ID session`
- `StatusCode`

## Définir l'audience

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

```
function callSetAudience(commandsToExecute, callback) {

    var ssId = document.getElementById('sa_sessionId').value;
    var audId = document.getElementById('sa_audienceId').value;
    var audLevel = document.getElementById('sa_audienceLevel').value;
    var params = document.getElementById('sa_parameters').value;

    InteractAPI.setAudience(ssId, getNameValuePairs(audId), audLevel,
        getNameValuePairs(params), callback);

}
```

- **sessionID** : chaîne identifiant l'ID session.
- **audienceID** - Tableau d'objets `NameValuePairImpl` définissant l'ID audience.
- **audienceLevel** - Chaîne définissant le niveau d'audience.

- **parameters** - Objets `NameValuePairImpl` identifiant tous les paramètres à transmettre 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 surensemble 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.

- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

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 audience. Vous pouvez utiliser la méthode `setAudience` pour forcer un rechargement des mêmes données de profil chargées par la méthode `startSession`.

La méthode `setAudience` recharge la table de liste blanche et la table de liste noire dans une session existante. Vous pouvez utiliser la méthode `setAudience` avec les paramètres `UACIPurgePriorWhiteListOnLoad` et `UACIPurgePriorBlackListOnLoad` pour recharger la table de liste blanche et la table de liste noire dans une session existante.

Par défaut, lorsque la méthode `setAudience` est appelée, tout le contenu de la liste noire est supprimé. Vous pouvez définir les paramètres `UACIPurgePriorWhiteListOnLoad` et `UACIPurgePriorBlackListOnLoad` dans l'appel `setAudience` comme suit :

- Si vous définissez `UACIPurgePriorBlackListOnLoad= 0`, tout le contenu de la table de liste blanche est conservé.
- Si vous définissez `UACIPurgePriorWhiteListOnLoad= 1`, le contenu de la table est supprimé et le contenu de la liste blanche ou de la liste noire pour l'ID audience est chargé depuis la base de données. Une fois l'opération terminée, une nouvelle segmentation est lancée.

## 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`
- `OfferList`
- `Profil`
- `ID session`
- `StatusCode`

## Obtenir le profil

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

```
function callGetProfile(commandsToExecute, callback) {  
  
    var ssId = document.getElementById('gp_sessionId').value;  
  
    InteractAPI.getProfile(ssId, callback);  
  
}
```

- **session ID** - Chaîne identifiant l'ID session.
- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

## Valeur de retour

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

- AdvisoryMessages
- ApiVersion
- OfferList
- ProfileRecord
- ID session
- StatusCode

## Terminer la session

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.

```
function callEndSession(commandsToExecute, callback) {  
  
    var ssId = document.getElementById('es_sessionId').value;  
  
    InteractAPI.endSession(ssId, callback);  
  
}
```

- **session ID** - Chaîne unique identifiant la session.
- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

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 :

- ID session
- ApiVersion
- OfferList

- Profil
- StatusCode
- AdvisoryMessages

## Définir le débogage

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

```
function callSetDebug(commandsToExecute, callback) {  
  
    var ssId = document.getElementById('sd_sessionId').value;  
    var isDebug = document.getElementById('isDebug').value;  
  
    InteractAPI.setDebug(ssId, isDebug, callback);  
  
}
```

- **sessionId** - Chaîne identifiant l'ID session.
- **debug** - Valeur booléenne qui active ou désactive les informations de débogage. Les valeurs valides sont `true` ou `false`. Si elle est `true`, Unica Interact journalise les informations de débogage dans les journaux du serveur d'exécution.
- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

### Valeur de retour

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

- AdvisoryMessages
- ApiVersion
- OfferList
- Profil

- ID session
- StatusCode

## Obtenir la version

La méthode `getVersion` renvoie la version de la mise en oeuvre actuelle du serveur d'exécution Unica Interact.

```
function callGetVersion(commandsToExecute, callback) {  
  
    InteractAPI.getVersion(callback);  
  
}
```

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

- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

## 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
- OfferList
- Profil
- ID session
- StatusCode

## executeBatch

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

```
function callExecuteBatch(commandsToExecute, callback) {  
  
    if (!commandsToExecute)  
        return ;  
  
    InteractAPI.executeBatch(commandsToExecute.ssid,  
        commandsToExecute.commands, callback);  
}
```

- **session ID** - Chaîne identifiant l'ID session. Cet ID de session est utilisé pour toutes les commandes exécutées par cet appel de méthode.
- **commands** - Tableau d'objets `command`, un pour chaque commande que vous souhaitez exécuter.
- **callback** - Si la méthode aboutit, la fonction de rappel appelle `onSuccess`. Si elle échoue, la fonction de rappel appelle `onError`.

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 d'API JavaScript

```
function isJavaScriptAPISelected() {
    var radios = document.getElementsByName('api');
    for (var i = 0, length = radios.length; i < length; i++) {
        if (radios[i].checked) {
            if (radios[i].value === 'JavaScript')
                return true ;
            else // only one radio can be logically checked
                break;
        }
    }
    return false;
}

function processFormForJSInvocation(e) {

    if (!isJavaScriptAPISelected())
        return;

    if (e.preventDefault) e.preventDefault();

    var serverurl = document.getElementById('serviceUrl').value ;
    InteractAPI.init( { "url" : serverurl } );

    var commandsToExecute = { "ssid" : null, "commands" : [] };
    var callback = InteractAPI.Callback.create(onSuccess, onError);

    callStartSession(commandsToExecute, callback);
}
```



```

    callGetOffers(commandsToExecute, callback);
    callGetOffersForMultipleInteractionPoints(commandsToExecute,
callback);
    callPostEvent(commandsToExecute, callback);
    callSetAudience(commandsToExecute, callback);
    callGetProfile(commandsToExecute, callback);
    callEndSession(commandsToExecute, callback);
    callSetDebug(commandsToExecute, callback);
    callGetVersion(commandsToExecute, callback);

    callExecuteBatch(commandsToExecute, callback);

    // You must return false to prevent the default form behavior
    return false;
}

function callStartSession(commandsToExecute, callback) {

    //read configured start session
    var ssId = document.getElementById('ss_sessionId').value;
    var icName = document.getElementById('ic').value;
    var audId = document.getElementById('audienceId').value;
    var audLevel = document.getElementById('audienceLevel').value;
    var params = document.getElementById('ss_parameters').value;
    var relyOldSs = document.getElementById('relyOnOldSession').value;
    var debug = document.getElementById('ss_isDebug').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {

```

```

        commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createStartSessionCmd(
                icName, getNameValuePairs(audId),
                audLevel, getNameValuePairs(params),
                relyOldSs, debug));
    }
    else {
        InteractAPI.startSession(ssId, icName,
            getNameValuePairs(audId), audLevel,
            getNameValuePairs(params), relyOldSs,
            debug, callback) ;
    }
}

function callGetOffers(commandsToExecute, callback) {

    var ssId = document.getElementById('go_sessionId').value;
    var ip = document.getElementById('go_ipoint').value;
    var nofRequested = 5 ;
    var nreqString = document.getElementById('offersRequested').value;
    if (!nreqString && nreqString!=='')
        nofRequested = Number(nreqString);

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createGetOffersCmd(ip,
                nofRequested));
    }
}

```

```

    }
    else {
        InteractAPI.getOffers(ssId, ip, nofRequested, callback);
    }
}

function callPostEvent(commandsToExecute, callback) {

    var ssId = document.getElementById('pe_sessionId').value;
    var ev = document.getElementById('event').value;
    var params = document.getElementById('parameters').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.
            CommandUtil.createPostEventCmd
            (ev, getNameValuePair(params)));
    }
    else {
        InteractAPI.postEvent(ssId, ev, getNameValuePair(params),
callback);
    }
}

function callGetOffersForMultipleInteractionPoints
(commandsToExecute, callback) {

    var ssId = document.getElementById('gop_sessionId').value;

```

```
var requestDetailsStr =
document.getElementById('requestDetail').value;

//trim string
var trimmed = requestDetailsStr.replace(/\{/g, "");
var parts = trimmed.split("{}");

//sanitize strings
for(i = 0; i < parts.length; i += 1) {
    parts[i] = parts[i].replace(/^\s+|\s+$/g, "");
}

//build get offer requests
var getOffReqs = [];
for(var i = 0; i < parts.length; i += 1) {
    var getofReqObj = parseGetOfferReq(parts[i]);
    if (getofReqObj) {
        getOffReqs.push(getofReqObj);
    }
}

if (commandsToExecute && !commandsToExecute.ssid) {
    commandsToExecute.ssid = ssId;
}

if (commandsToExecute && commandsToExecute.commands) {
    commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                    createGetOffersForMultiple
                                    InteractionPointsCmd(getOffReqs));
}
else {
    InteractAPI.getOffersForMultipleInteractionPoints
```

```

        (ssId, getOffReqs, callback);
    }
}

function parseGetOfferReq(ofReqStr) {

    if (!ofReqStr || ofReqStr=="")
        return null;

    var posIp = ofReqStr.indexOf(',');
    var ip = ofReqStr.substring(0,posIp);
    var posNmReq = ofReqStr.indexOf(',', posIp+1);
    var numReq = ofReqStr.substring(posIp+1,posNmReq);
    var posDup = ofReqStr.indexOf(',', posNmReq+1);
    var dupPolicy = null;
    var reqAttributes = null;

    if (posDup===-1)
        dupPolicy = ofReqStr.substring(posNmReq+1);
    else
        dupPolicy = ofReqStr.substring(posNmReq+1,posDup);

    //check if request string has attributes
    var reqAttrPos = ofReqStr.search(/\(/g);
    if (reqAttrPos!==-1) {
        var reqAttributesStr = ofReqStr.substring(reqAttrPos);
        reqAttributesStr = trimString(reqAttributesStr);
        reqAttributesStr = removeOpenCloseBrackets(reqAttributesStr);
        reqAttributes = parseReqAttributes(reqAttributesStr);
    }

    return InteractAPI.GetOfferRequest.create(ip, parseInt(numReq),

```

```

        parseInt(dupPolicy), reqAttributes);
    }

    //trim string
    function trimString(strToTrim) {
        if (strToTrim)
            return strToTrim.replace(/^\s+|\s+$/g, "");
        else
            return null;
    }

    function trimStrArray(strArray) {
        if (!strArray) return ;
        for(var i = 0; i < strArray.length; i += 1) {
            strArray[i] = trimString(strArray[i]);
        }
    }

    //remove open and close brackets in the end
    function removeOpenCloseBrackets(strToUpdate) {
        if (strToUpdate)
            return strToUpdate.replace(/^\^(+|\)+$/g, "");
        else
            return null;
    }

    function parseReqAttributes(ofReqAttrStr) {

        //sanitize string
        ofReqAttrStr = trimString(ofReqAttrStr);
        ofReqAttrStr = removeOpenCloseBrackets(ofReqAttrStr);
    }

```

```

if (!ofReqAttrStr || ofReqAttrStr==="")
    return null;

//get the number requested
var pos = ofReqAttrStr.indexOf(",");
var numRequested = ofReqAttrStr.substring(0,pos);
ofReqAttrStr = ofReqAttrStr.substring(pos+1);

//first part will be attribute and rest will be child attributes
var parts = [];
pos = ofReqAttrStr.indexOf(",");
if (pos!==-1) {
    parts.push(ofReqAttrStr.substring(0,pos));
    parts.push(ofReqAttrStr.substring(pos+1));
}
else {
    parts.push(ofReqAttrStr);
}

for(var i = 0; i < parts.length; i += 1) {
    //sanitize string
    parts[i] = trimString(parts[i]);
    parts[i] = removeOpenCloseBrackets(parts[i]);
    parts[i] = trimString(parts[i]);
}

//build list of attributes
var attributes = [];
var idx = 0;
if (parts[0]) {
    var attParts = parts[0].split(";");
    for (idx=0; idx<attParts.length; idx++) {

```

```

        attParts[idx] = trimString(attParts[idx]);
        attParts[idx] = removeOpenCloseBrackets(attParts[idx]);
        attParts[idx] = trimString(attParts[idx]);

        var atrObj = parseAttribute(attParts[idx]);
        if (atrObj) attributes.push(atrObj);
    }
}

//build list of child attributes
var childAttributes = [];
if (parts[1]) {
    var childAttParts = parts[1].split("");
    for (idx=0; idx<childAttParts.length; idx++) {

        childAttParts[idx] = trimString(childAttParts[idx]);
        childAttParts[idx] =
removeOpenCloseBrackets(childAttParts[idx]);
        childAttParts[idx] = trimString(childAttParts[idx]);

        //get the number requested
        var noReqPos = childAttParts[idx].indexOf(",");
        var numReqAt = childAttParts[idx].substring(0,noReqPos);
        childAttParts[idx] =
childAttParts[idx].substring(noReqPos+1);
        childAttParts[idx] = trimString(childAttParts[idx]);

        var atrObjParsed = parseAttribute(childAttParts[idx]);
        if (atrObjParsed) {

```



```

        var childReq =
InteractAPI.OfferAttributeRequirements.create
        (parseInt(numReqAt), [atrObjParsed], null);
        childAttributes.push(childReq);
    }
}
}

return
InteractAPI.OfferAttributeRequirements.create(parseInt(numRequested),
    attributes, childAttributes);
}

function parseAttribute(attStr) {

    attStr = trimString(attStr);

    if (!attStr || attStr=="")
        return null;

    var pos1 = attStr.indexOf("=");
    var pos2 = attStr.indexOf("|");
    var nvp = InteractAPI.NameValuePair.create
        ( attStr.substring(0,pos1),
          attStr.substring(pos1+1, pos2),
          attStr.substring(pos2+1));

    return nvp;
}

function callSetAudience(commandsToExecute, callback) {
    if (!document.getElementById('checkSetAudience').checked)
        return ;
}

```

```
var ssId = document.getElementById('sa_sessionId').value;
var audId = document.getElementById('sa_audienceId').value;
var audLevel = document.getElementById('sa_audienceLevel').value;
var params = document.getElementById('sa_parameters').value;

if (commandsToExecute && !commandsToExecute.ssid) {
    commandsToExecute.ssid = ssId;
}

if (commandsToExecute && commandsToExecute.commands) {
    commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createSetAudienceCmd
        (getNameValuePairs(audId), audLevel,
getNameValuePairs(params)));
}
else {
    InteractAPI.setAudience(ssId, getNameValuePairs(audId),
        audLevel, getNameValuePairs(params),
        callback);
}
}

function callGetProfile(commandsToExecute, callback) {

    var ssId = document.getElementById('gp_sessionId').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
```

```
        createGetProfileCmd());
    }
    else {
        InteractAPI.getProfile(ssId, callback);
    }
}

function callEndSession(commandsToExecute, callback) {

    var ssId = document.getElementById('es_sessionId').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createEndSessionCmd());
    }
    else {
        InteractAPI.endSession(ssId, callback);
    }
}

function callSetDebug(commandsToExecute, callback) {

    var ssId = document.getElementById('sd_sessionId').value;
    var isDebug = document.getElementById('isDebug').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }
}
```

```
    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createSetDebugCmd(isDebug));
    }
    else {
        InteractAPI.setDebug(ssId, isDebug, callback);
    }
}

function callGetVersion(commandsToExecute, callback) {

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createGetVersionCmd());
    }
    else {
        InteractAPI.getVersion(callback);
    }
}

function callExecuteBatch(commandsToExecute, callback) {

    if (!commandsToExecute)
        return ;

    InteractAPI.executeBatch(commandsToExecute.ssid,
        commandsToExecute.commands, callback);
}

function getNameValuePairs(parameters) {
```

```

if (parameters === '')
    return null ;

var parts = parameters.split(';');
var nvpArray = new Array(parts.length);

for(i = 0; i < parts.length; i += 1) {
    var nvp = parts[i].split(',') ;
    var value = null;
    if
(nvp[2]===InteractAPI.NameValuePair.prototype.TypeEnum.NUMERIC) {
        if (isNaN(nvp[1])) {
            value = nvp[1]; //a non number was provided as
number,
            pass it to API as it is
        }
        else {
            value = Number(nvp[1]);
        }
    }
    else {
        value = nvp[1];
    }
    //special handling NULL value
    if (value && typeof value === 'string') {
        if (value.toUpperCase() === 'NULL') {
            value = null;
        }
    }
    nvpArray[i] = InteractAPI.NameValuePair.create(nvp[0], value,
nvp[2]) ;
}

```

```

        return nvpArray;
    }

    function showResponse(textDisplay) {
        var newWin = open('', 'Response', 'height=300,width=300,titlebar=no,
            scrollbars=yes,toolbar=no,
            resizable=yes,menubar=no,location=no,status=no');

        if (newWin.locationbar !== 'undefined' && newWin.locationbar
            && newWin.locationbar.visible)
            newWin.locationbar.visible = false;

        var displayHTML = '<META HTTP-EQUIV="Content-Type"
            CONTENT="text/html; charset=UTF-8">
            <html><head><style>TD { border-width : thin; border-style :
solid }</style.'
            + "<script language='Javascript'>"
            + "var desiredDomain = 'unicacorp.com'; "
            + "if (location.href.indexOf(desiredDomain)>=0)
"
            + "{ document.domain = desiredDomain;} "
            + "</script></head><body> "
            + textDisplay
            + "</body></html>" ;

        newWin.document.body.innerHTML = displayHTML;
        newWin.focus() ;
    }

    function onSuccess(response) {
        showResponse("*****Response*****<br> " +
JSON.stringify(response)) ;
    }

```

```

}

function onError(response) {
    showResponse("*****Error*****<br> " + response) ;
}

function formatResoponse(response) {

}

function printBatchResponse(batResponse) {

}

function printResponse(response) {

}

```

## Exemple d'objet de réponse JavaScript onSuccess

Cet exemple montre les trois variables de l'objet de réponse JavaScript : offerLists, messages et profile.

`offerList` renvoie une liste non nulle si `getOffer` ou `getOffersForMultipleInteractionPoints` est appelé en tant qu'API ou sous la forme d'une commande de traitement par lots. Vous devez toujours tester le null avant d'effectuer des opérations sur cette variable.

Vous devez toujours vérifier le statut de la réponse JavaScript de `messages`.

`Profile` renvoie une valeur non nulle si `getProfile` est appelé en tant qu'API ou sous la forme d'une commande de traitement par lots. Si vous n'utilisez pas `getProfile`, vous pouvez ignorer cette variable. Vous devez toujours tester le null avant d'effectuer des opérations sur cette variable.

```

function onSuccess(response)
InteractAPI.ResponseTransUtil._buildResponse = function(response) {
    'use strict';

    if (!response) return null;

    var offerList = null;
    //transform offerLists to JS Objects
    if (response.offerLists) {
        offerList = [];
        for (var ofListCt=0;
ofListCt<response.offerLists.length;ofListCt++) {
            var ofListObj =
this._buildOfferList(response.offerLists[ofListCt]);
            if (ofListObj) offerList.push(ofListObj);
        }
    }

    var messages = null;
    //transform messages to JS Objects
    if (response.messages) {
        messages = [];
        for (var msgCt=0; msgCt<response.messages.length;msgCt++) {
            var msgObj =
this._buildAdvisoryMessage(response.messages[msgCt]);
            if (msgObj) messages.push(msgObj);
        }
    }

    var profile = null;
    //transform profile nvps to JS Objects
    if (response.profile) {

```





# Chapitre 10. A propos de l'API ExternalCallout

Unica Interact propose une macro extensible, `EXTERNALCALLOUT`, qui peut être utilisée avec vos diagrammes temps réel. 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 temps réel 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 propriétés de configuration Unica Platform sur le serveur d'exécution dans la catégorie `Interact | diagramme | ExternalCallouts`.

Outre les informations contenues dans cette section, le JavaDoc de l'API ExternalCallout est disponible sur n'importe quel serveur d'exécution Unica 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. Le serveur d'exécution ne comporte pas d'utilitaire de nettoyage automatique des connexions.

Vous devez compiler votre mise en œuvre par rapport à `interact_externalcallout.jar` qui se trouve dans le répertoire `lib` de votre installation d'environnement d'exécution 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`
- `getValue`
- `initialize`
- `shutdown`

## Ajout d'un service Web à utiliser avec la macro EXTERNALCALLOUT

Cette procédure permet d'ajouter un service Web à utiliser avec la macro `EXTERNALCALLOUT`. La macro `EXTERNALCALLOUT` reconnaît les appels externes uniquement si vous avez défini les propriétés de configuration appropriées.

Dans l'environnement d'exécution Unica Platform pour l'environnement de conception, éditez les propriétés de configuration suivantes dans la catégorie `Interact > flowchart > externalCallouts`.

Propriété de configuration	Paramètre
Catégorie <code>externalCallouts</code>	Créez une catégorie pour votre appel.
<code>class</code>	Noms de classe de votre appel

Propriété de configuration	Paramètre
classpath	Chemin d'accès aux fichiers de classe de vos appels externes
Catégorie Données de paramètre	Si votre appel requiert des paramètres, créez de nouvelles propriétés de configuration de paramètre et affectez à chacun une valeur.

## getNumberOfArguments

Interact permet de transmettre un nombre variable d'arguments à votre appel externe. La méthode `getNumberOfArguments` doit renvoyer -1 pour autoriser un nombre variable d'arguments. La méthode `getNumberOfArguments` renvoie le nombre d'arguments attendus par la classe Java™ avec laquelle vous effectuez l'intégration.

```
getNumberOfArguments()
```

### 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

La méthode `getValue` exécute la fonctionnalité de base de l'appel et renvoie les résultats.

```
getValue(audienceID, configData, arguments)
```

La méthode `getValue` nécessite les paramètres suivants :

- **audienceID** - Valeur identifiant l'ID audience.
- **configData** - Mappe avec des paires clé-valeur des données de configuration requises par l'appel.
- **arguments** - Arguments requis par l'appel. 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 mise en œuvre.

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;
}
```

## initialiser

La méthode `initialize` est appelée une fois lorsque le serveur d'exécution démarre. S'il existe des opérations 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.

```
initialize(configData)
```

La méthode `initialize` nécessite les paramètres suivants :

- **configData** - Mappe avec des paires clé-valeur des données de configuration requises par l'appel.

Unica 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

Aucun.

## 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

La méthode `shutdown` est appelée lorsque le serveur d'exécution démarre. Si des tâches de nettoyage sont requises par votre appel, elles doivent exécuter à ce moment.

```
shutdown(configData)
```

La méthode `shutdown` nécessite le paramètre suivant :

- **configData** - Mappe avec des paires clé-valeur des données de configuration requises par l'appel.

Si la méthode `shutdown` échoue pour une raison quelconque, elle renvoie `CalloutException`.

## Valeur de retour

Aucun.

## Exemple

```
public void shutdown(Map<String, String> configurationData) throws
    CalloutException
{
    // shutdown scoreQueryUtility here
}
```

## Exemple d'API ExternalCallout

Cet exemple crée un appel externe qui extrait un score de crédit.

Créez un appel externe qui extrait un score de crédit :

1. Créez un fichier appelé `GetCreditScore.java` ayant le contenu ci-après. 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.IAffiniumExtern
alCallout;
```

```
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`, dans la page **Gestion des configurations**.
6. Dans un diagramme temps réel, l'appel 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 Unica 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 Unica Interact ou du changement de l'ID d'audience Unica 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 les mappe à un objet `ISessionDataRootNode`, puis désigner ces données mappées avec la macro `EXTERNALCALLOUT` dans le processus de sélection d'un diagramme temps réel.

Vous devez compiler votre mise en œuvre par rapport à `interact_externalcallout.jar` qui se trouve dans le répertoire `lib` de votre installation d'environnement d'exécution 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 la mise en œuvre de l'exemple, voir `Interact_home/samples/externalcallout/XMLProfileDataService.java`.



**Remarque :** L'exemple d'implémentation est destiné à n'être utilisé qu'à titre d'exemple. Vous ne devez pas l'utiliser dans votre implémentation.

## Ajout d'une source de données à utiliser avec Profile Data Services

Cette procédure permet d'ajouter 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 Unica Platform pour l'environnement d'exécution, 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 niveau d'audience.
activé	Indique si cette source de données est activée pour le niveau d'audience auquel elle est affectée.
className	Nom qualifié complet de la classe de source de données qui implémente <code>IInteractProfileDataService</code>
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 ce niveau d'audience. Il doit s'agir d'une valeur unique dans toutes les sources de données de chaque niveau d'audience. En d'autres termes, si une priorité est définie sur 100 pour une source de données, aucune autre source de données dans le niveau d'audience ne peut avoir une priorité de 100.

## Interface IParameterizableCallout

L'API des appels paramétrables se trouve dans l'interface `IParameterizableCallout`.

Cette interface est l'interface de base des interfaces d'API exposées qui peuvent accepter les paramètres de la configuration via Unica Platform. Comme il s'agit d'une interface de base, elle ne doit pas être implémentée directement. Les paramètres sont extraits des nœuds enfant du nœud `Parameter Data` sous la catégorie qui fait référence à cette implémentation. Dans l'exemple ci-après, ESB est une implémentation personnalisée du service de données de profil, qui à son tour implémente l'interface `IParameterizableCallout`. Les paramètres `endPoint` et `login`, sont transmis avec leurs

valeurs dans cette classe d'implémentation lorsque le moteur Unica Interact tente de l'initialiser et d'y mettre fin.

```
Profile Data Services
...ESB
  ...Parameter Data
    ...endPoint
    ...login
```

Cette interface comprend deux méthodes :

- `initialize`
- `shutdown`

## initialiser

La méthode `initialize` initialise cette classe d'implémentation.

```
void initialize(java.util.Map<java.lang.String, java.lang.String>
configurationData)
    throws CalloutException
```

La méthode `initialize` nécessite le paramètre suivant :

- **configurationData** - Mappe avec des paires nom/valeur de paramètres configurés par les utilisateurs

## Exception générée

```
CalloutException
```

## shutdown

La méthode `shutdown` arrête cette classe d'implémentation.

```
void shutdown(java.util.Map<java.lang.String,java.lang.String>
configurationData)
throws CalloutException
```

La méthode `shutdown` nécessite le paramètre suivant :

- **configurationData** - Mappe avec des paires nom/valeur de paramètres configurés par les utilisateurs

## Exception générée

```
CalloutException
```

## Interface ITriggeredMessageAction

L'API d'action de message déclenché se trouve dans l'interface `ITriggeredMessageAction`. Cette interface vous permet d'obtenir et de définir le nom de cette instance.

L'interface `ITriggeredMessageAction` sert d'interface de base pour les autres interfaces et ne doit jamais être implémentée directement.

Cette interface comprend deux méthodes :

- `getName`
- `setName`

### getName

La méthode `getName` renvoie le nom de l'instance `ITriggeredMessageAction`.

```
java.lang.String getName()
```

### setName

La méthode `setName` définit le nom de l'instance `ITriggeredMessageAction`.

```
void setName(java.lang.String name)
```

Lorsque vous initialisez la classe d'implémentation de cette interface, Unica Interact définit le nom de l'interface avec le nom indiqué dans l'interface utilisateur de configuration.

Dans l'exemple ci-après, le nom de cette passerelle est `InteractLog`.

```
triggeredMessage
    ...gateways
        ...InteractLog
```

La méthode `setName` nécessite le paramètre suivant :

- `name` - Nom que vous souhaitez définir pour l'instance `ITriggeredMessageAction`.

## Interface IChannelSelector

L'API de sélecteur de canal se trouve dans l'interface `IChannelSelector`. Cette interface permet de sélectionner les canaux sortants en fonction de l'offre à envoyer et des attributs de session.

Pour obtenir un exemple d'implémentation de l'utilisation de l'action de message déclenché, avec des descriptions commentées de l'implémentation de l'exemple, voir [Interact\\_home/samples/triggeredmessage/SampleChannelSelector.java](#).



**Remarque :** L'exemple d'implémentation est destiné à n'être utilisé qu'à titre d'exemple. Vous ne devez pas l'utiliser dans votre implémentation.

Vous devez essayer d'utiliser cette implémentation au lieu d'écrire la vôtre.

Cette interface comprend une méthode :

- `selectChannels`

### selectChannels

La méthode `selectChannels` sélectionne les canaux sortants auxquels l'offre transmise doit être envoyée avec l'interface `IChannelSelector`.

```

java.util.List<java.lang.String> selectChannels

( java.util.Map<java.lang.String, java.util.Map<java.lang.String,
                                     java.lang.Object>> availableChannels,
  com.uniacorp.interact.api.Offer offer,
  com.uniacorp.interact.treatment.
  optimization.IInteractSessionData
  sessionData )

```

Unica Interact tente d'envoyer cette offre à tous les canaux renvoyés.

La méthode `selectChannels` nécessite les paramètres suivants :

- **availableChannels** : mappe des canaux sortants disponibles qui sont configurés dans l'interface utilisateur des messages déclenchés, dans les paramètres de la phase de conception d'Unica Interact. Dans chaque entrée de la mappe, la clé correspond au nom du canal et la valeur, aux paramètres configurés pour ce canal dans la phase de conception d'Unica Interact. L'ordre d'itération de cette mappe correspond à l'ordre défini sur cette interface utilisateur. Si le canal préféré du profil est utilisé sur l'interface utilisateur des messages déclenchés, il est remplacé par le véritable canal avant que cette méthode ne soit appelée. En outre, si le même canal intervient plusieurs fois sur l'interface utilisateur, seule l'occurrence de priorité la plus élevée est conservée et tous les doublons sont supprimés.
- **offer** : offre à distribuer
- **sessionData** : attributs actuellement stockés dans la session Unica Interact associée

## Interface IDispatcher

L'API de répartiteur se trouve dans l'interface `IDispatcher`. Cette interface envoie des offres aux passerelles ciblées.

Comme il n'existe qu'une instance de cette classe pour chaque répartiteur configuré, l'implémentation de cette interface doit être sans état dans le cadre d'Unica Interact.

Pour obtenir un exemple d'implémentation de l'utilisation de l'action de message déclenché, avec des descriptions commentées de l'implémentation de l'exemple, voir [Interact\\_home/samples/triggeredmessage/SampleDispatcher.java](#).



**Remarque :** L'exemple d'implémentation est destiné à n'être utilisé qu'à titre d'exemple. Vous ne devez pas l'utiliser dans votre implémentation.

Vous devez essayer d'utiliser cette implémentation au lieu d'écrire la vôtre.

Cette interface comprend une méthode :

- `dispatch`

## dispatch

La méthode `dispatch` envoie des offres aux passerelles cible dans l'interface `IDispatcher`.

```
boolean dispatch(java.lang.String channel,
                 java.lang.String gatewayName,
                 java.util.Collection<com.unicacorp.interact.api.Offer> offers,
                 com.unicacorp.interact.api.NameValuePair[] profileData)
    throws com.unicacorp.interact.exceptions.InteractException
```

Une fois que les canaux sortants ont été sélectionnés pour une offre candidate, Unica Interact tente d'envoyer les offres candidates aux gestionnaires associés au canal. Les gestionnaires sont essayés en fonction de leur priorité définie, de la plus élevée à la plus basse. Pour chaque gestionnaire, Unica Interact appelle cette méthode du répartiteur configuré. C'est à l'implémentation de cette instance de répartiteur de déterminer la manière d'acheminer l'offre à la passerelle cible, qui est configurée dans le même gestionnaire. Si plusieurs offres sont envoyées au même gestionnaire suite à la même évaluation de message déclenché, Unica Interact essaye d'envoyer toutes ces offres dans un lot.

La méthode `dispatch` nécessite les paramètres suivants :

- **channel** - Canal de sortie auquel ces offres sont envoyées
- **gatewayName** - Nom de la passerelle cible



- **offers** - Offres à envoyer à la passerelle dans un lot
- **profileData** - Attributs de profils alimentés par `IGateway.validate` et transmis à `IGateway.deliver`

## Valeur de retour

La méthode `dispatch` indique si la répartition a abouti ou échoué

## Exception générée

```
com.unicacorp.interact.exceptions.InteractException
```

## Interface IGateway

L'API de passerelle se trouve dans l'interface `IGateway`. Cette interface reçoit des offres d'Unica Interact et les envoie à leur destination.

Chaque implémentation de cette interface communique avec une destination particulière. La destination doit effectuer la conversion de données nécessaire, fournir les attributs requis et effectuer les tâches similaires liées à la destination.

Pour obtenir un exemple d'implémentation de l'utilisation de l'action de message déclenché, avec des descriptions commentées de l'implémentation de l'exemple, voir [Interact\\_home/samples/triggeredmessage/SampleOutboundGateway.java](#).



**Remarque :** L'exemple d'implémentation est destiné à n'être utilisé qu'à titre d'exemple. Vous ne devez pas l'utiliser dans votre implémentation. Par exemple : `SampleOutboundGateway` est inclus sous le répertoire d'exemple pour une référence de mise en œuvre.

Cette interface comprend deux méthodes :

- `deliver`
- `validate`

## deliver

La méthode `deliver` est appelée pour envoyer la ou les offres à une destination de l'interface `IGateway`.

```
void deliver(java.util.Collection<com.unicacorp.interact.api.Offer> offers,
            com.unicacorp.interact.api.NameValuePair[]
            profileData,
            java.lang.String channel)
```

La méthode `deliver` nécessite les paramètres suivants :

- **offers** - Offre à envoyer
- **profileData** - Attributs de profil spécifiés dans `parameterMap` par la méthode de validation
- **channel** : canal de sortie auquel ces offres seront envoyées

## validate

La méthode `validate` valide les offres candidates dans l'interface `IGateway`.

```
java.util.Collection<com.unicacorp.interact.api.Offer> validate
    (com.unicacorp.interact.treatment.optimization.
    IInteractSessionData sessionData,
    java.util.Collection<com.unicacorp.interact.api.Offer>
    candidateOffers,
    java.util.Map<java.lang.String, java.lang.Object> parameterMap,
    java.lang.String channel)
```

Le moteur Unica Interact appelle cette méthode pour valider les offres candidates. L'implémentation de cette méthode doit vérifier les offres, les attributs d'offre et les attributs de session par rapport aux exigences de la destination pour déterminer quelles offres peuvent être envoyées par l'intermédiaire de cette passerelle. En outre, elle peut ajouter les paramètres nécessaires dans la mappe transmise, qui est retransmise à la méthode de distribution.

La méthode `validate` nécessite les paramètres suivants :

- **sessionData** : attributs actuellement stockés dans la session Unica Interact associée
- **candidateOffers** : offres sélectionnées par Unica Interact en fonction de la méthode de sélection des offres, de ses paramètres et d'autres facteurs. Ces offres peuvent être distribuées à partir de la perspective d'Unica Interact, mais sont tout de même sujettes à la passerelle.
- **parameterMap** : mappe à utiliser par l'implémentation de cette méthode pour transmettre des paramètres à sa méthode de distribution
- **channel** : canal de sortie auquel ces offres seront envoyées

# Chapitre 11. Utilitaires Unica Interact

Cette section décrit les utilitaires d'administration fournis avec Unica 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 serveurs 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 fonctionnalité `OffersBySQL`. Par exemple, vous pouvez configurer un diagramme de traitement par lots Unica 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 Unica 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 serveurs canal interactif/exécution. Un échantillon de fichier est fourni à titre de référence.



**Remarque :** Avant d'utiliser l'utilitaire `runDeployment`, vous devez d'abord l'éditer avec un éditeur de texte pour indiquer l'emplacement de l'environnement d'exécution 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 .

## Utilisation de l'utilitaire runDeployment dans un environnement sécurisé (SSL)

Pour utiliser l'utilitaire runDeployment, lorsque la sécurité a été activée sur le serveur Unica 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
```

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 Unica Interact, et remplacez `deployment.properties` par le chemin et le nom réels du fichier de propriétés que vous avez personnalisé pour ce déploiement.

## Echantillon de fichier `deployment.properties`

L'échantillon de fichier `deployment.properties` contient la liste commentée de tous les paramètres que vous devez personnaliser pour le faire correspondre à votre environnement. Cet échantillon de fichier contient également des commentaires qui décrivent chaque paramètre, et explique pourquoi il peut être nécessaire de personnaliser une valeur particulière.

```
#####
####
#
# The following properties feed into the InvokeDeploymentClient program.
# The program will look for a deploymentURL setting. The program will post
a
# request against that url; all other settings are posted as parameters in
# that request. The program then checks the status of the deployment and
# returns back when the deployment is at a terminal state (or if the
# specified waitTime has been reached).
#
# the output of the program will be of this format:
# <STATE> : <Misc Detail>
#
# where state can be one of the following:
# ERROR
# RUNNING
# SUCCESS
#
# Misc Detail is data that would normally populate the status message area
```

```

# in the deployment gui of the IC summary page. NOTE: HTML tags may exist
# in the Misc Detail
#
#####
####

#####
####
# deploymentURL: url to the InvokeDeployment servlet that resides in
  Interact
# Design time.  should be in the following format:
# http://dt_host:port/Campaign/interact/InvokeDeploymentServlet
#####
####
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet
et

#####
####
# dtLogin:  this is the login that you would use to login to the Design
  Time if
# you had wanted to deploy the IC via the deployment gui inside the IC
  summary
# page.
#####
####
dtLogin=asm_admin

#####
####
# dtPW:  this is the PW that goes along with the dtLogin

```

```
#####  
####  
dtPW=  
  
#####  
####  
# icName:  this is the name of the Interactive Channel that you want to  
  deploy  
#####  
####  
icName=ic1  
  
#####  
####  
# partition:  this is the name of the partition  
#####  
####  
partition=partition1  
  
#####  
####  
# request:  this is the type of request that you want this tool to execute  
# currently, there two behaviors.  If the value is "deploy", then the  
  deployment  
# will be executed.  All other values would cause the tool to simply return  
  the  
# status of the last deployment of the specified IC.  
#####  
####  
request=deploy
```



```
#####  
####  
# serverGroup:  this is the name of the server group that you would like to  
# deploy the IC.  
#####  
####  
serverGroup=defaultServerGroup  
  
#####  
####  
# serverGroupType:  this will indicate whether or not this deployment is  
going  
# against production server group or a test server group. 1 denotes  
production  
# 2 denotes test.  
#####  
####  
serverGroupType=1  
  
#####  
####  
# rtLogin:  this is the account used to authenticate against the server  
group  
# that you are deploying to.  
#####  
####  
rtLogin=asm_admin  
  
#####  
####  
# rtPW:  this is the password associated to the rtLogin
```

```
#####  
####  
rtPW=  
  
#####  
####  
# waitTime: Once the tool submits the deployment request, the tool will  
# check  
# the status of the deployment. If the deployment has not completed (or  
# failed), then the tool will continue to poll the system for the status  
# until  
# a completed state has been reached, OR until the specified waitTime (in  
# seconds) has been reached.  
  
#####  
####  
waitTime=5  
  
#####  
####  
# pollTime: If the status of a deployment is still in running state, then  
# the  
# tool will continue to check the status. It will sleep in between status  
# checks a number of seconds based on the pollTime setting .  
  
#####  
####  
pollTime=3  
  
#####  
####  
# global: Setting to false will make the tool NOT deploy the global  
# settings.  
# Non-availability of the property will still deploy the global settings.
```

```
#####
####
global=true
```

## Nettoyage de l'utilitaire du jeton arrivé à expiration

Pour implémenter le code et la configuration dans Interact Runtime, les jetons arrivés à expiration doivent être supprimés de la table UACI\_RTToken. Cette opération doit être effectuée automatiquement à l'aide d'une unité d'exécution en arrière-plan avec un intervalle configurable. Cet arrière-plan doit être générique, afin qu'il puisse servir à nettoyer d'autres tables.

### Changements apportés à la configuration

Un nœud "Nettoyage" est ajouté sous le chemin d'accès à "Affinium|interact|services" afin d'effectuer les activités liées au nettoyage. Un nœud secondaire "expiredTokens" est ajouté sous le chemin d'accès à "Affinium|interact|services|Cleanup" afin de réaliser l'opération de nettoyage pour les jetons arrivés à expiration. Vous devez définir la zone "activer" en tant que `true` sous le chemin d'accès à "Affinium|interact|services|Cleanup|expiredTokens". La valeur par défaut de la zone "activer" de "expiredTokens" est définie en tant que `True`. Les valeurs admises sont `True` et `False`.



**Remarque :** Les utilisateurs peuvent insérer les jetons dans la table UACI\_RTToken en démarrant une session et en activant la zone "tokenAuthentication" sous le chemin d'accès à "Affinium|interact|general|API".

### Paramètres JVM facultatifs.

Les utilisateurs peuvent définir les paramètres JVM suivants sous les options JVM.

- -Dcom.unica.interact.cleanupThreadPoolCoreSize=1
- -Dcom.unica.interact.cleanupInitialDelay=300
- -Dcom.unica.interact.cleanupDelay=300
- -Dcom.unica.interact.cleanupBatchSize=5000

où :

- `cleanupThreadPoolCoreSize` est le nombre d'unités d'exécution devant être conservées dans le pool. S'il n'est pas défini, le système prend la valeur par défaut 300.
- `cleanupInitialDelay` est le délai de la première exécution, exprimé en secondes. S'il n'est pas défini, le système prend la valeur par défaut 5 000.
- `cleanupDelay` est le délai entre la fin d'une exécution et le début de la suivante, exprimé en secondes. S'il n'est pas défini, le système prend la valeur par défaut 5 000.
- `cleanupBatchSize` est le nombre de jetons arrivés à expiration que l'utilisateur souhaite supprimer en une seule tentative. S'il n'est pas défini, le système prend la valeur par défaut 5 000.

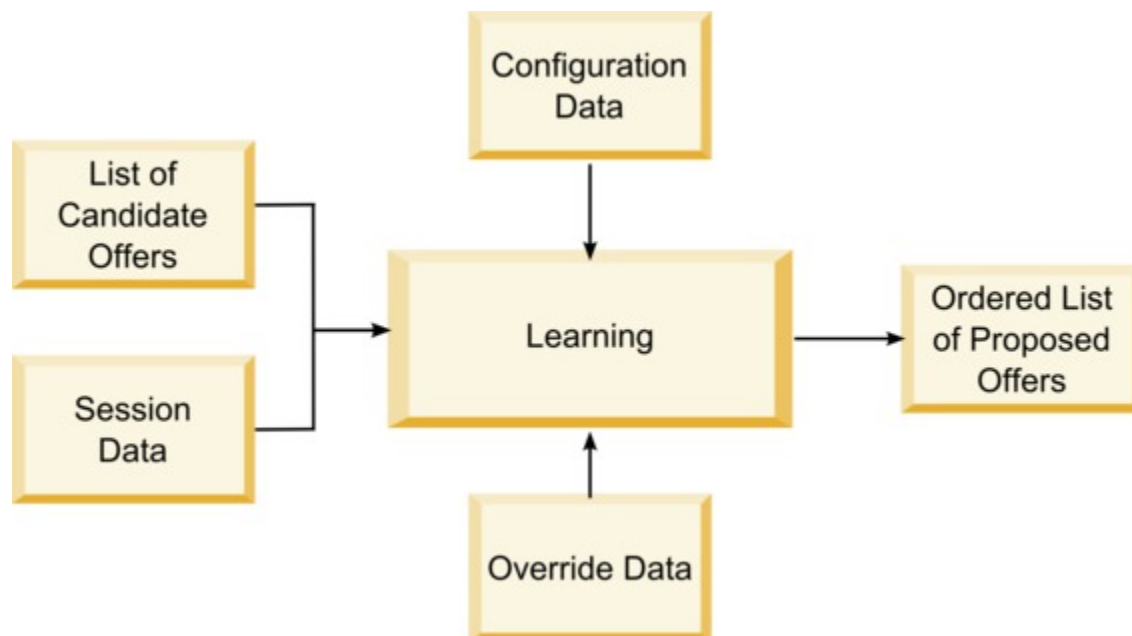
# Chapitre 12. A propos de l'API d'apprentissage

Unica Interact fournit un module d'apprentissage, qui utilise un algorithme bayésien 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 Unica Interact

- Données de contact de l'offre
- Données d'acceptation de l'offre
- Toutes les données de session
- Donnée d'offre spécifiques à Unica 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 algorithme 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 mise en œuvre 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 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 mise en œuvre par rapport à `interact_learning.jar` qui se trouve dans le répertoire `lib` de votre installation d'environnement d'exécution Unica Interact.

Lorsque vous écrivez votre mise en œuvre 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.

## Configuration de l'environnement d'exécution pour qu'il reconnaisse les modules d'apprentissage externes

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 Unica Platform.

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

1. Dans Unica 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
<code>optimizationType</code>	<b>ExternalLearning</b>
<code>externalLearningClass</code>	Nom de classe de l'apprentissage externe
<code>externalLearningClassPath</code>	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 Unica Platform, une copie de la classe ou des fichiers

Propriété de configuration	Paramètre
	JAR doit être présente au même emplacement sur chaque serveur.

2. Redémarrez le serveur d'exécution Unica Interact pour que les changements 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.

### initialiser

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.

```
initialize(ILearningConfig config, boolean debug)
```





- **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`.

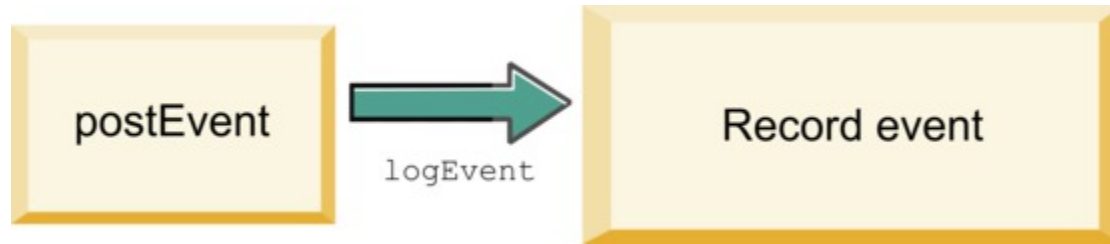
## Valeur de retour

Aucun.

## logEvent

La méthode `logEvent` est appelée par le serveur d'exécution lorsque l'API Unica 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.

```
logEvent(ILearningContext context,  
         IOffer offer,  
         IClientArgs clientArgs,  
         IInteractSession session,  
         boolean debug)
```



- **context** - Objet `ILearningContext` définissant le contexte d'apprentissage de l'événement, par exemple, contact, acceptation ou refus.
- **offer** - Objet `IOffer` définissant l'offre à propos de laquelle cet événement est consigné.
- **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 proximité 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

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 commandée 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.

```
optimizeRecommendList(list(ITreatment) recList,
    IClientArgs clientArg, IInteractSession session,
    boolean debug)
```



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 requises par l'environnement d'exécution.
- **session** - Objet `IInteractSession` contenant toutes les données de session.
- **debug** - Valeur booléenne. S'il s'agit de `true`, indique que la proximité 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

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.

```
reinitialize(ILearningConfig config,  
            boolean debug)
```



- **config** - Objet `ILearningConfig` qui contient toutes les propriétés de configuration.
- **debug** - Valeur booléenne. S'il s'agit de `true`, indique que la proximité 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`.

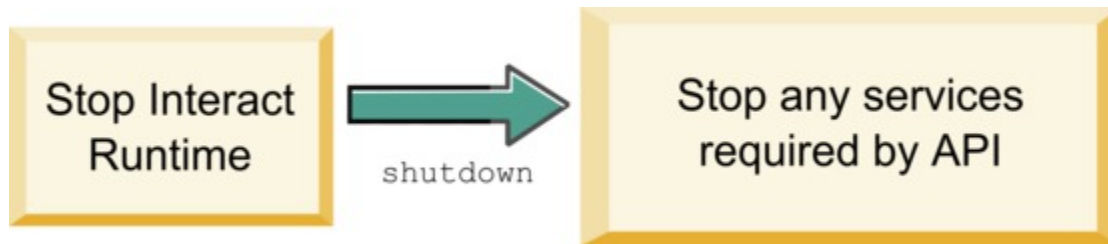
### Valeur de retour

Aucun.

## shutdown

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.

```
shutdown(ILearningConfig config, boolean debug)
```



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 proximité 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

Aucun.

## Interface `IAudienceID`

L'interface `IAudienceID` prend en charge l'interface `IInteractSession`. Il s'agit d'une interface pour l'ID d'audience. Etant donné que votre ID d'audience peut être composé de plusieurs parties, cette interface permet d'accéder à tous les éléments de l'ID d'audience ainsi qu'à son nom.

### `getAudienceLevel`

La méthode `getAudienceLevel` renvoie le niveau d'audience.

```
getAudienceLevel()
```

## Valeur de retour

La méthode `getAudienceLevel` renvoie une chaîne qui définit le niveau d'audience.

## getComponentNames

La méthode `getComponentNames` obtient l'ensemble de noms des composants qui constituent l'ID audience. Par exemple, si votre ID d'audience est constitué des valeurs `customerName` et `accountID`, `getComponentNames` renvoie un ensemble contenant les chaînes `customerName` et `accountID`.

```
getComponentNames()
```

## Valeur de retour

Un ensemble de chaînes contenant les noms des composants de l'ID d'audience.

## getComponentValue

La méthode `getComponentValue` renvoie la valeur du composant indiqué.

```
getComponentValue(String componentName)
```

- **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 passées 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 Unica Interact. Ces données sont stockées dans une mappe.

## getValue

La méthode `getValue` renvoie la valeur de l'élément de mappe demandée.

```
getValue(int clientArgKey)
```

Les éléments suivants sont requis dans la mappe.

- **1 - NUMBER\_OF\_OFFERS\_REQUESTED.** Nombre d'offres demandées par la méthode `getOffers` de l'API Unica 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

La méthode `getAudienceId` renvoie un objet `AudienceID`. Utilisez l'interface `IAudienceID` pour extraire les valeurs.

```
getAudienceId()
```

### Valeur de retour

La méthode `getAudienceId` renvoie un objet `AudienceID`.

## 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.

```
getSessionData()
```

## 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 changer la valeur des données dans la session d'exécution.

## `getDataType`

La méthode `getDataType` renvoie le type de données du nom de paramètre indiqué.

```
getType(string parameterName)
```

## Valeur de retour

La méthode `getType` renvoie un objet `InteractDataType`. `InteractDataType` est une énumération Java™ représentée par Inconnu, String, Double, Date ou Liste.

## `getParameterNames`

La méthode `getParameterNames` renvoie un ensemble de tous les noms des données dans la session en cours.

```
getParameterNames()
```



## 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

La méthode `getValue` renvoie la valeur de l'objet correspondant au `parameterName` indiqué. L'objet peut être Chaîne, Double ou Date.

```
getValue(parameterName)
```

La méthode `getValue` nécessite le paramètre suivant :

- **parameterName** - Chaîne définissant la paire nom-valeur des données de session.

## Valeur de retour

La méthode `getValue` renvoie un objet contenant la valeur du paramètre nommé.

## setValue

La méthode `setValue` permet de définir une valeur pour le `parameterName` indiqué. La valeur peut être Chaîne, Double ou Date.

```
setValue(string parameterName, object value)
```

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

Aucun.

# 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

La méthode `getName` renvoie le nom de l'attribut d'apprentissage.

```
getName ( )
```

### 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`
- **getAggregateStatsIntervallInMinutes** - Renvoie un entier
- **getConfidenceLevel** - Renvoie un entier
- **getDataSourceName** - Renvoie une chaîne
- **getDataSourceType** - Renvoie une chaîne
- **getInsertRawStatsIntervallInMinutes** - 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

La méthode `getLearningContext` renvoie la constante qui nous indique s'il s'agit d'un scénario de contact, d'acceptation ou de refus.

```
getLearningContext()
```

- **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

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 Unica Campaign.

```
getResponseCode()
```

### 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

La méthode `getCreateDate` renvoie la date de création de l'offre.

```
getCreateDate()
```

### Valeur de retour

La méthode `getCreateDate` renvoie la date de création de l'offre.

## getEffectiveDateFlag

La méthode `getEffectiveDateFlag` renvoie un nombre qui définit la date effective de l'offre.

```
getEffectiveDateFlag()
```

- **0** - La date effective est une date absolue, telle que le 15 mars 2010.
- **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

La méthode `getExpirationDateFlag` renvoie un entier qui définit la date d'expiration de l'offre.

```
getExpirationDateFlag()
```

- **0** - Une date absolue, par exemple le 15 mars 2010.
- **1** - Un certain nombre de jours après la recommandation, par exemple 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

La méthode `getOfferAttributes` renvoie les attributs de l'offre définis pour l'offre sous la forme d'un objet `IOfferAttributes`.

```
getOfferAttributes()
```

## Valeur de retour

La méthode `getOfferAttributes` renvoie un objet `IOfferAttributes`.

## getOfferCode

La méthode `getOfferCode` renvoie le code de l'offre comme défini dans Unica Campaign.

```
getOfferCode()
```

## Valeur de retour

La méthode `getOfferCode` renvoie un objet `IOfferCode`.

## getOfferDescription

La méthode `getOfferDescription` renvoie la description de l'offre définie dans Unica Campaign.

```
getOfferDescription()
```

### Valeur de retour

La méthode `getOfferDescription` renvoie une chaîne.

## getOfferID

La méthode `getOfferID` renvoie le ID de l'offre tel qu'il est défini dans Unica Campaign.

```
getOfferID()
```

### Valeur de retour

La méthode `getOfferID` renvoie une valeur longue qui définit le ID de l'offre.

## getOfferName

La méthode `getOfferName` renvoie le nom de l'offre tel qu'il est défini dans Unica Campaign.

```
getOfferName()
```

### Valeur de retour

La méthode `getOfferName` renvoie une chaîne.

## getUpdateDate

La méthode `getUpdateDate` renvoie la date de la dernière mise à jour de l'offre.

```
getUpdateDate()
```

### 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

La méthode `getParameterNames` renvoie une liste des noms de paramètres de l'offre.

```
getParameterNames()
```

#### Valeur de retour

La méthode `getParameterNames` renvoie un ensemble définissant la liste des noms de paramètres de l'offre.

### getValue

La méthode `getValue` renvoie un objet qui définit la valeur de l'attribut de l'offre.

```
getValue(String parameterName)
```

La méthode `getValue` renvoie la valeur d'un attribut d'offre donné.

#### Valeur de retour

## 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

La méthode `getPartCount` renvoie le nombre d'éléments qui constituent un code d'offre.

```
getPartCount()
```

### Valeur de retour

La méthode `getPartCount` renvoie un entier définissant le nombre d'éléments qui constituent le code de l'offre.

## getParts

La méthode `getParts` obtient une liste non modifiable des éléments du code de l'offre.

```
getParts()
```

### 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 mises en oeuvre 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

La méthode `getOfferCode` renvoie la valeur des colonnes de code de l'offre dans la table de substitution de score pour ce membre d'audience.

```
getOfferCode()
```

### 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

La méthode `getParameterNames` renvoie la liste des paramètres.

```
getParameterNames()
```

### 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

La méthode `getValue` renvoie la valeur de la colonne de zone dans la table de substitution de score pour ce membre d'audience.

```
getValue(String parameterName)
```

- **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

La méthode `getCellCode` renvoie le code de cible tel qu'il est défini dans Unica Campaign. La cible est la cible affectée au segment dynamique associé à cette offre.

```
getCellCode()
```

### Valeur de retour

La méthode `getCellCode` renvoie une chaîne qui définit le code de cible.

## getCellId

La méthode `getCellId` renvoie l'ID interne de la cible tel qu'il est défini dans Unica Campaign. La cible est la cible affectée au segment dynamique associé à cette offre.

```
getCellId()
```

### Valeur de retour

La méthode `getCellId` renvoie une valeur longue qui définit le ID de cible.

## getCellName

La méthode `getCellName` renvoie le nom de la cible tel qu'il est défini dans Unica Campaign. La cible est la cible affectée au segment dynamique associé à cette offre.

```
getCellName()
```

## Valeur de retour

La méthode `getCellName` renvoie une chaîne qui définit le nom de la cible.

## getLearningScore

La méthode `getLearningScore` renvoie le score de ce traitement.

## Valeur de retour

La méthode `getLearningScore` renvoie un entier qui définit le score déterminé par l'algorithme d'apprentissage. La priorité est la suivante.

1. Renvoie la valeur de substitution, si elle est présente dans la mappe des valeurs de substitution indexée par `IScoreoverride.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 `IScoreoverride.SCORE`.
4. Renvoie la valeur des vendeurs.

## getMarketerScore

La méthode `getMarketerScore` renvoie le score du vendeur défini par le curseur dans l'onglet Stratégie d'interaction de l'offre.

```
getMarketerScore()
```

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

La méthode `getOffer` renvoie l'offre de ce traitement.

```
getOffer()
```

### Valeur de retour

La méthode `getOffer` renvoie un objet `IOffer` qui définit l'offre de ce traitement.

## getOverrideValues

La méthode `getOverrideValues` renvoie les substitutions définies dans les offres par défaut ou la table de substitution des scores.

```
getOverrideValues()
```

### Valeur de retour

La méthode `getOverrideValues` renvoie un objet `IScoreOverride`.

## 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.

```
getPredicate()
```

### 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

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.

```
getPredicateScore()
```

### 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

La méthode `getScore` renvoie le score marketing défini par la stratégie d'interaction dans Unica Campaign ou par la table de substitution de score.

```
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 Unica Campaign si la propriété `enableScoreOverrideLookup` est définie sur `false`.
- Le score de l'offre tel que défini par le `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

La méthode `getTreatmentCode` renvoie le code de traitement.

```
getTreatmentCode()
```

## Valeur de retour

La méthode `getTreatmentCode` renvoie une chaîne qui définit le code de traitement.

## setActualValueUsed

Utilisez la méthode `setActualValueUsed` pour définir les valeurs qui sont utilisés à différentes étapes de l'exécution de l'algorithme d'apprentissage.

```
setActualValueUsed(string paramName, object value)
```

Par exemple, si vous utilisez cette méthode pour écrire dans les tables de l'historique des contacts et des réponses, et modifiez les exemples de rapport existants, vous pouvez inclure des données de votre algorithme d'apprentissage dans la génération de rapports.

- **paramName** - Chaîne définissant le nom du paramètre que vous définissez.
- **value** - Objet contenant la valeur du paramètre que vous définissez.

## Valeur de retour

Aucun.

## Exemple d'API d'apprentissage

Cette section contient un échantillon de mise en œuvre d'`ILearningInterface`. Notez que cette mise en œuvre 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 du nombre de 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;

/**
 * This is a sample implementation of the learning optimizer.
 * The interface ILearning may be found in the interact.jar library.
 *
 * To actually use this implementation, select ExternalLearning as the
 * optimizationType in the offerServing node
 * of the Unica Interact application within the Platform configuration.
 * Within the offerserving node there is also
```



```

* an External Learning config category - within there you must set the
name of the class to this:
* com.unicacorp.interact.samples.learning.v2.SampleLearning. Please note
however, this implementation is just a sample
* and was not designed to be used in a production environment.
*
*
* This example keeps track of accept and contact counts and uses the ratio
of accept to contacts
* for a particular offer as the acceptance probability rate for the offer.
*
*
* Offers not presented will get higher priority for recommending.
* Offers with at least one contact will be ordered based on descending
acceptance probability rate.
*
* Note: all counts are kept in memory. This is not a realistic scenario
since you would run out of memory sooner or
* later. In a real production scenario, the counts should be persisted
into a database.
*
*/

```

```

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.uniacorp.interact.treatment.optimization.v2.ILearning#initialize
 * (com.uniacorp.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.uniacorp.interact.treatment.optimization.v2.ILearning#reinitialize
 * (com.uniacorp.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#optimizeRecomme
ndList

```

```

    * (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);
    }
}
}
```

# Chapitre 13. Unica Interact WSDL

L'installation d'Unica Interact inclut deux fichiers XML WSDL (Web Services Description Language) qui décrivent les services Web disponibles et indiquent comment y accéder. Vous pouvez afficher ces fichiers dans votre répertoire de base d'Unica Interact. Un exemple est illustré ici.

Après l'installation d'Unica Interact, vous pouvez trouver les fichiers WSDL Unica Interact à l'emplacement suivant :

- `<Interact_home>/conf/InteractService.wsdl`
- `<Interact_home>/conf/InteractAdminService.wsdl`

A chaque nouvelle édition ou groupe de correctifs, Unica Interact WSDL peut présenter des modifications. Pour plus d'informations, voir les Unica InteractNotes sur l'édition ou les fichiers Readme fournis avec l'édition.

Une copie de `InteractService.wsdl` est présentée ici pour référence. Pour vous assurer que vous utilisez la documentation la plus récente, voir les fichiers WSDL installés avec Unica Interact.

```
<?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/"
  xmlns: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="nsl:CommandImpl"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="executeBatchResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" name="return" nillable="false"
type="nsl: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>
  </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="nsl: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="nsl:NameValuePairImpl"/>
      <xs:element minOccurs="1" name="audienceLevel" nillable="false"
type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="parameters"
nillable="true" type="nsl: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: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>
```

# Chapitre 14. Propriétés de configuration de l'environnement d'exécution d'Unica Interact

Cette section décrit toutes les propriétés de configuration de l'environnement d'exécution d'Unica Interact.

## Interact | général

Ces propriétés de configuration définissent les paramètres généraux de votre environnement d'exécution, notamment le niveau de journalisation par défaut et la propriété des paramètres régionaux.

### log4jConfig

#### Description

Emplacement du fichier qui contient les propriétés log4j. Ce chemin doit être absolu ou relatif à la variable d'environnement `INTERACT_HOME`. `INTERACT_HOME` est l'emplacement du répertoire d'installation d'Unica Interact.

#### Valeur par défaut

```
./conf/interact_log4j2.xml
```

### asmUserForDefaultLocale

#### Description

La propriété `asmUserForDefaultLocale` définit l'utilisateur Unica duquel Unica Interact dérive ses paramètres régionaux.

Les propriétés des paramètres régionaux définissent la langue d'affichage lors de la phase de conception et la phase d'exécution et celle des messages de conseils dans lesquels se trouvent les API Unica Interact. Si les paramètres régionaux ne correspondent pas à ceux du système d'exploitation de vos machines, Unica Interact continuera de s'exécuter, mais il se peut que la langue d'affichage de la phase de conception et des messages de conseils soit différente.

## Valeur par défaut

`asm_admin`

## configurationRefreshInMins

### Description

L'utilisateur Admin disposant du rôle Admin (Platform) peut activer ou désactiver la fonctionnalité pour mettre à jour dynamiquement les données de configuration en définissant l'actualisation de la configuration en minutes, via les valeurs  $\leq 0$  (off) ou  $> 0$  (on). Par exemple, la valeur actuelle de l'actualisation de la configuration en minutes est de 5 minutes. Les changements apportés à la configuration peuvent avoir besoin d'au maximum 5 minutes pour être effectifs. La nouvelle valeur définie de l'actualisation de la configuration est effective en quelques minutes. Le système rafraîchira certaines des données de configuration conformément aux changements apportés plus tard à la valeur.

Remarque : l'actualisation automatique n'est pas prise en charge pour les paramètres de configuration suivants :

1. Général : toutes les sources de données ne configurent pas d'assistance, seul le niveau supérieur configure une assistance
2. Contrôle
3. profil
4. cacheManagement
5. triggeredMessage
6. activityOrchestrator
7. simulator
8. ETL

## Interact | général | API

Cette propriété de configuration définit le paramètre pour l'authentification des API Unica Interact.



## **tokenAuthentication**

### **La description**

Utilisez cette propriété tokenAuthentication pour activer ou désactiver l'authentification pour les API Unica Interact.

### **Valeur par défaut**

FALSE

### **Valeurs valides**

TRUE | FALSE

## **enabledLogging**

### **La description**

Utilisez cette propriété pour déterminer s'il faut activer la journalisation de l'API globalement.

### **Valeur par défaut**

FALSE

### **Valeurs valides**

TRUE | FALSE

## **Interact | général | centralizedLogger**

Cette propriété de configuration définit la journalisation centralisée pour Unica Interact.

## **enabled**

### **Description**

Définit si le consignateur centralisé est activé ou désactivé dans Unica Interact.

### **Valeur par défaut**

FALSE

### **Valeurs valides**

TRUE | FALSE

## **maxBatchSize**

### **Description**

Le nombre maximal d'entrées de journal qui seront conservées dans la base de données dans un seul lot.

### **Valeur par défaut**

1000

### **Valeurs valides**

Entiers non négatifs et positifs

## **maxDelayInSec**

### **Description**

Les durées d'attente maximales d'une entrée de journal avant d'être conservée dans la base de données.

### **Valeur par défaut**

15

### **Valeurs valides**

Entiers non négatifs et positifs

Même si la fonctionnalité centralizedLogger est activée pour l'ensemble du groupe de serveur, elle peut être désactivée sur un seul serveur en indiquant la ligne suivante dans le fichier interact\_log4j2.xml de ce serveur :

```
<AppenderRef ref="db"/>
```

## **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'auto-apprentissage d'Unica Interact.

Si vous créez votre propre mise en oeuvre d'apprentissage via l'API d'apprentissage, vous pouvez configurer votre mise en oeuvre d'apprentissage personnalisée pour lire ces valeurs à l'aide de l'interface `ILearningConfig`.

## **jndiName**

### **Description**

Cette propriété `jndiName` sert à identifier la source de données JNDI (Java™ Naming and Directory Interface) qui est définie sur le serveur d'applications (Websphere ou WebLogic) pour les tables d'apprentissage auxquelles accèdent les serveurs d'exécution Unica Interact.

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'Unica 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 | MARIADB

## connectionRetryPeriod

### Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Unica Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables d'apprentissage. Unica 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, Unica 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 Unica 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 destinées au module d'apprentissage intégré. Unica 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, Unica 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'Unica 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'Unica Interact.

#### **Valeur par défaut**

SQLServer

### Valeurs valides

SQLServer | DB2® | ORACLE | MARIADB

## aliasPrefix

### Description

La propriété `AliasPrefix` spécifie la manière dont Unica Interact génère le nom d'alias qu'Unica Interact crée automatiquement en cas d'utilisation d'une table des dimensions et d'écriture dans une nouvelle table accessible par les serveurs d'exécution d'Unica 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

H

## connectionRetryPeriod

### Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Unica Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables client de l'environnement d'exécution. Unica 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, Unica 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 Unica Interact attend avant toute tentative de reconnexion à la base de données pour les tables client de l'environnement d'exécution d'Unica 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 vos tables de données de profil. Unica 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, Unica 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.

Lorsque vous utilisez une base de données DB2, le nom du schéma doit être en majuscules.

### Valeur par défaut

Aucune valeur par défaut définie.

## Interact | general | API | requestThreadPool

Cette propriété de configuration configure le pool de threads qui récupère les traitements supplémentaires à partir d'autres ID d'audience spécifiés dans `UACISupplementAudience`.

### corePoolSize

### La description

Le nombre minimal de threads dans ce pool.

**Valeur par défaut**

Aucun

**Valeurs valides**

Numérique

**maxPoolSize**

Le nombre maximal de threads dans ce pool.

**Valeur par défaut**

Aucun

**Valeurs valides**

Numérique

**keepAliveTimeSecs**

La durée maximale en secondes pendant laquelle le système attend avant de supprimer un thread inactif de ce pool.

**Valeur par défaut**

Aucun

**Valeurs valides**

Numérique

**termWaitSecs**

La durée maximale en secondes pendant laquelle le système attend avant d'arrêter un thread dans ce pool lorsque le système s'arrête.

**Valeur par défaut**

Aucun

**Valeurs valides**

Numérique



## 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**

Cette propriété `jndiName` sert à identifier la source de données JNDI (Java™ Naming and Directory Interface) qui est définie sur le serveur d'applications (Websphere ou WebLogic) pour les tables de l'environnement d'exécution.

La base de données de l'environnement d'exécution est remplie 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 remplie 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 | MARIADB

### **connectionRetryPeriod**

#### **Description**

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Unica Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables système de l'environnement d'exécution. Unica 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, Unica 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 remplie 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 Unica Interact attend avant toute tentative de reconnexion à la base de données pour les tables système de l'environnement d'exécution d'Unica 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 remplie 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

## **schéma**

#### **Description**

Nom du schéma qui contient les tables destinées à l'environnement d'exécution. Unica 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, Unica 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 | général | 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 Unica 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 d'accès 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 Unica Interact conformément au modèle spécifié dans le paramètre <code>LoaderControlFileTemplate</code> .
<DATABASE>	Ce jeton est remplacé par le nom de la source de données dans laquelle Unica Interact charge des données. Le nom de la source de données est le même que celui appliqué au nom de 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 Unica Interact pendant le proces-

Jeton	Description
	sus de chargement. Ce fichier se trouve dans UNICA_ACTMPDIR, le répertoire temporaire de Unica Interact.
<DBCOLUMN-NUMBER>	Ce jeton est remplacé par l'ordinal de colonne de la base de données.
<FIELDLENGTH>	Ce jeton est remplacé par la longueur de la zone en cours de chargement dans la base de données.
<FIELDNAME>	Ce jeton est remplacé par le nom de la zone en cours de chargement dans la base de données.
<FIELDNUMBER>	Ce jeton est remplacé par le numéro de la zone en cours de chargement dans la base de données.
<FIELDTYPE>	Ce jeton est remplacé par le littéral "CHAR()". La longueur de cette zone est spécifiée entre les parenthèses (). Si votre base de données ne comprend pas le type de la zone, 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 cette zone a été chargée.
<NUMFIELDS>	Ce jeton est remplacé par le nombre de zones contenues 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.

Jeton	Description
<TABLENAME>	Ce jeton est remplacé par le nom de la table de la base de données dans laquelle Unica Interact charge des données.
<USER>	Ce jeton est remplacé par le nom d'utilisateur de la base de données utilisé lors de la connexion de diagramme actuel à la source de données.

### Valeur par défaut

Aucune valeur par défaut définie.

## LoaderControlFileTemplateForAppend

### Description

La propriété `LoaderControlFileTemplateForAppend` indique le chemin d'accès complet et le nom du modèle de fichier de contrôle préalablement configuré dans Unica Interact. Si ce paramètre est configuré, Unica 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 Unica 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 Unica 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 jetons associés à votre modèle de fichier de contrôle sont les mêmes que ceux associés 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 Unica 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, Unica Interact crée le fichier de données temporaire sous la forme d'un fichier à plat à largeur fixe.

Si vous spécifiez une valeur, celle-ci est utilisée lorsque l'utilitaire de chargement est appelé pour remplir une table potentiellement renseignée. Unica Interact crée le fichier de données temporaire sous la forme d'un fichier à 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 remplir une table potentiellement renseignée, Unica Interact applique des délimiteurs à la fin de chaque ligne.

### Valeur par défaut

FAUX

### Valeurs valides

TRUE | FALSE

## LoaderUseLocaleDP

### Description

La propriété `LoaderUseLocaleDP` indique, lorsque Unica 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 séparateur décimal.

Définissez cette valeur sur `FALSE` pour indiquer que le point (.) est utilisé en tant que séparateur décimal.

Définissez-la sur `TRUE` pour indiquer que le symbole de séparateur décimal propre à votre région est utilisé.

### Valeur par défaut

FAUX

### 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 Unica 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



Cette propriété `jndiName` sert à identifier la source de données JNDI (Java™ Naming and Directory Interface) qui est définie sur le serveur d'applications (Websphere ou WebLogic) pour les tables client auxquelles accède l'environnement de conception lors de l'exécution de tests de diagrammes temps réel.

### **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 | MARIADB

## **aliasPrefix**

### **Description**

La propriété `AliasPrefix` spécifie la manière dont Unica Interact génère le nom d'alias que Unica Interact crée automatiquement en cas d'utilisation d'une table des dimensions et d'écriture dans une nouvelle table pour les tables client accessibles par l'environnement de conception quand il exécute des tests de diagrammes temps réel.

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

H

## connectionRetryPeriod

### Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Unica Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour les tables d'exécution de test. Unica 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, Unica 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 Unica 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 aux exécutions de diagramme temps réel. Unica 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, Unica 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 de réponse intersession d'Unica Interact. Ces paramètres ne concernent pas le module d'historique des réponses et des contacts.

### **jndiName**

#### **Description**

Cette propriété `jndiName` sert à identifier la source de données Java™ Naming and Directory Interface (JNDI) qui est définie sur le serveur d'applications (WebSphere® ou WebLogic) comme source de données de contacts et d'historique des réponses pour le suivi de réponse intersession d'Unica Interact.

#### **Valeur par défaut**

### **type**

#### **Description**

Type de la base de données associé à la source utilisée par la source de suivi de réponse intersession Unica Interact.

#### **Valeur par défaut**

SQLServer

#### **Valeurs valides**

SQLServer | DB2® | ORACLE | MARIADB

## connectionRetryPeriod

### Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Unica Interact retente automatiquement la demande de connexion à la base de données en cas d'échec pour le suivi de réponse intersession Unica Interact. Unica 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, Unica 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 Unica Interact attend avant toute tentative de reconnexion à la base de données pour le suivi de réponse intersession Unica 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 destinées au suivi de réponse intersession d'Unica Interact. Unica 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, Unica 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 | général | 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 | diagramme

Cette section définit les paramètres de configuration des diagrammes temps réel.

Vous ne pouvez pas utiliser de formats DT\_DELIM\_XXX avec des diagrammes de session temps réel.

### **defaultDateFormat**

#### **Description**

Format de date par défaut utilisé par Unica Interact pour convertir la date en chaîne et inversement.

#### **Valeur par défaut**

dd/MM/yy

### **idleFlowchartThreadTimeoutInMinutes**

#### **Description**

Nombre de minutes pendant lesquelles Unica 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 Unica 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'Unica Interact maintient dans la file d'attente. Si ce nombre de requêtes est atteint, Unica 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 d'unités d'exécution dédiées 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 d'unités d'exécution dédiées aux processus de diagrammes temps réel.

**Valeur par défaut**

20

**sessionVarPrefix**

**Description**

Préfixe des variables de session.

**Valeur par défaut**

SessionVar

**Interact | diagramme | ExternalCallouts | [NomAppelExterne]**

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™ accessible par la macro `EXTERNALCALLOUT`.

**Valeur par défaut**

Aucune valeur par défaut définie.



## classpath

### Description

Le classpath de la classe Java™ représentée par cet appel externe. Le classpath 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 Unica Platform, une copie du fichier jar doit être présente au même emplacement de chaque serveur. Le classpath 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, sur 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 à la classe

Il s'agit de la classe Java™ accessible par la macro `EXTERNALCALLOUT`.

Si plusieurs JAR sont fournis dans ce paramètre, ils doivent être séparés à l'aide de leur caractère de séparation de chemin d'accès à la plateforme, comme le point-virgule ";" sur Windows, et les deux-points ":" sur Linux).

### Valeur par défaut

Aucune valeur par défaut définie.

## Remarque sur l'appel d'API de service Web depuis des appels externes Interact

- Interact recommande que les modules d'appels externes n'effectuent aucun appel SOAP supplémentaire pour mener à bien leur traitement. Bien que cela puisse fonctionner correctement en cas de configuration adéquate, cela peut également entraîner un chargement de classe et des erreurs internes de configuration SOAP qui sont difficiles à corriger. Il est recommandé d'utiliser REST ou d'autres cadres d'API

de service Web si vous souhaitez effectuer des appels vers d'autres systèmes depuis leurs appels externes.

- Si vous devez utiliser des API SOAP depuis des appels externes Interact, ils doivent être compilés par rapport à la version de SOAP/Axis2 incluse avec Interact. Au moment de cette publication, la version est Axis2 version 1.52. Il est possible de déterminer la version exacte utilisée par Interact en recherchant les fichiers `*axis2*.jar` dans la bibliothèque `InteractRT.war/web-inf/lib` et en notant la version répertoriée à la fin de leur nom de fichier (par exemple, `axis2-kernel-1.5.2.jar`). La version d'Axis2 utilisée pour compiler vos stubs SOAP est généralement indiquée en commentaire en haut du fichier `*Stub.java` SOAP généré ou qui vous a été fourni.
- Si votre appel externe rencontre des erreurs SOAP, vous devez activer la journalisation de chargement de classe sur votre serveur d'application afin de vérifier que seules les bibliothèques Axis2 et Axiom fournies par Interact sont chargées et utilisées par Interact et votre code d'appel externe.
- Les erreurs de traitement d'appels externes se trouvent dans le fichier `interact.log`.

## Interact | diagramme | ExternalCallouts | [NomAppelExterne] | Données de paramètre | [NomParamètre]

Cette section définit le réglage des paramètres d'un appel personnalisé écrit à l'aide de l'API des appels externes.

### valeur

#### Description

Valeur d'un paramètre requis par la classe de l'appel.

#### Valeur par défaut

Aucune valeur par défaut définie.

#### Exemple

Si le nom d'hôte d'un serveur externe est requis par l'appel, créez une catégorie de paramètre nommée `host` et définissez la propriété `value` en tant que nom du serveur.

## Interact | surveillance

Ces propriétés de configuration vous permettent de définir les paramètres de suivi JMX. Vous ne devez configurer ces propriétés que si vous utilisez la surveillance 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'Unica Interact.

### protocol

#### Description

Définissez le protocole du service de messagerie d'Unica 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

9 998

### enableSecurity

#### Description

Opérateur booléen qui active ou désactive la sécurité du serveur d'exécution Unica 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'Unica Interact. Ces données d'identification sont authentifiées par Unica 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 est sans effet sur JMX pour Unica Campaign (la phase de conception d'Unica Interact).

#### **Valeur par défaut**

True

#### **Valeurs valides**

True | False

## Interact | surveillance | activitySubscribers

Cet ensemble de propriétés de configuration active le noeud racine des paramètres associés aux abonnés éloignés qui peuvent recevoir une mise à jour périodique sur les données de performances de base dans l'environnement d'exécution d'Unica Interact.

### **heartbeatPeriodInSecs**

#### **Description**

Intervalle, en secondes, pendant lequel chaque instance d'exécution envoie une mise à jour aux abonnés.

#### **Valeur par défaut**

60

## Interact | surveillance | activitySubscribers | (cible)

### **(target)**

#### **Description**

Noeud racine des paramètres d'un abonné.

## URL

### Description

URL de cet abonné. Ce nœud final doit pouvoir accepter des messages JSON transportés via HTTP.

## continuousErrorsForAbort

### Description

Nombre de mises à jour ayant échoué en continu avant que l'instance d'exécution arrête d'envoyer d'autres mises à jour à cet abonné.

### Valeur par défaut

5

## timeoutInMillis

### Description

Délai, en millisecondes, pendant lequel le processus d'envoi attend lors de l'envoi de mises à jour à cet abonné.

### Valeur par défaut

1 000

### Valeurs valides

## Activé

### Description

Indique si cet abonné est activé ou désactivé.

### Valeur par défaut

True

### Valeurs valides

True OU False

## type

### Description

Type de ce magasin de données. Lorsque cette option est sélectionnée, le paramètre **className** doit être ajouté à la valeur correspondant au nom qualifié complet de cette classe d'implémentation. **classPath** doit être ajouté à l'identificateur URI du fichier JAR s'il ne se trouve pas dans le chemin d'accès aux classes de l'environnement d'exécution Interact.

### Valeur par défaut

`InteractLog`

### Valeurs valides

`InteractLog`, `RelationalDB` et `Custom`

## jmxInclusionCycles

### Description

Intervalle dans le multiplicateur de **heartbeatPeriodInSecs** auquel des statistiques JMX détaillées sont envoyées à cet abonné.

### Valeur par défaut

5

### Valeurs valides

## Interact | profil

Ces propriétés de configuration contrôlent plusieurs fonctionnalités de service d'offres facultatives, notamment la suppression des offres et le remplacement de score.

## enableScoreOverrideLookup

### Description

Si la propriété est définie sur `True`, Unica 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`, Unica 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 `Interact | profil | Niveaux d'audience | (Niveau d'audience) | scoreOverrideTable`. Vous devez définir la propriété `scoreOverrideTable` uniquement pour les niveaux d'audiences dont vous avez besoin. Si la propriété `scoreOverrideTable` est en blanc pour un niveau d'audience, la table de remplacement de score correspondante est désactivée.

### Valeur par défaut

Faux

### Valeurs valides

True | False

## enableOfferSuppressionLookup

### Description

Si la propriété est définie sur `True`, Unica 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`, Unica 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 `Interact | profil | Niveaux d'audience | (Niveau d'audience) | offerSuppressionTable`. Vous devez définir la propriété `enableOfferSuppressionLookup` uniquement pour les niveaux d'audience dont vous avez besoin.

### Valeur par défaut

Faux

### Valeurs valides

True | False

## enableProfileLookup

### Description

En cas de nouvelle installation d'Unica Interact, cette propriété n'est pas autorisée. En cas de mise à jour de l'installation d'Unica 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`, Unica 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 `Interact | profil | Niveaux d'audience | (Niveau d'audience) | profileTable`.

Le paramètre **Charger ces données dans la mémoire au début d'une session de visite** de l'assistant de mappage des tables de canal interactif remplace cette propriété de configuration.

### Valeur par défaut

Faux

### 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 | profil | Niveaux d'audience | [Nom du niveau d'audience]

Ces propriétés de configuration vous permettent de définir les noms de table requis pour des fonctions Unica Interact supplémentaires. Vous ne devez définir le nom de la table que lorsque vous utilisez la fonctionnalité correspondante.

### Nouveau nom de la catégorie

#### Description

Nom de votre niveau d'audience.

### scoreOverrideTable

#### Description

Nom de la table qui contient les informations de remplacement de score associées à ce niveau d'audience. Cette propriété s'applique si vous avez défini la propriété `enableScoreOverrideLookup` sur `true`. Vous devez définir cette propriété pour les niveaux d'audience 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 niveau d'audience, vous pouvez laisser cette propriété non définie, même si `enableScoreOverrideLookup` est défini sur `true`.

Unica Interact recherche cette table dans les tables client accessibles aux serveurs d'exécution Unica Interact, définis par la propriété `prodUserDataSource`.

Si vous avez défini la propriété `schema` pour cette source de données, Unica Interact ajoutera ce nom de table au schéma, par exemple, `schema.UACI_ScoreOverride`. Si vous saisissez un nom qualifié complet, tel que `mySchema.UACI_ScoreOverride`, Unica 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 niveau d'audience. Vous devez définir cette propriété pour les niveaux d'audience 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 niveau d'audience, vous pouvez laisser cette propriété non définie. Si `enableOfferSuppressionLookup` a pour valeur `true`, cette propriété doit avoir pour valeur une table valide.

Unica Interact recherche cette table dans les tables client accessibles aux serveurs d'exécution, définis par la propriété `prodUserDataSource`.

### Valeur par défaut

UACI\_BlackList

## contactHistoryTable

### Description

Nom de la table de transfert qui contient les données d'historique des contacts associées à ce niveau d'audience.

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, Unica Interact ajoute ce nom de table au schéma, par exemple, `schema.UACI_CHStaging`. Si vous saisissez un nom qualifié complet, tel que `mySchema.UACI_CHStaging`, Unica Interact n'ajoute pas le nom du schéma.

Si la journalisation de l'historique des contacts est désactivée, il n'est pas nécessaire de définir cette propriété.

### 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 niveau d'audience.

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, Unica Interact ajoute ce nom de table au schéma, par exemple, `schema.UACI_CHOfferAttrib`. Si vous saisissez un nom qualifié complet, tel que `mySchema.UACI_CHOfferAttrib`, Unica Interact n'ajoute pas le nom du schéma.

Si la journalisation de l'historique des contacts est désactivée, il n'est pas nécessaire de définir cette propriété.

### Valeur par défaut

UACI\_CHOfferAttrib

## responseHistoryTable

### Description

Nom de la table intermédiaire qui contient la table de transfert de l'historique des réponses associée à ce niveau d'audience.

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, Unica Interact ajoute ce nom de table au schéma, par exemple, `schema.UACI_RHStaging`. Si vous saisissez un nom complet, tel que `mySchema.UACI_RHStaging`, Unica Interact n'ajoute pas le nom du schéma.

Si la journalisation de l'historique des réponses est désactivée, il n'est pas nécessaire de définir cette propriété.

### Valeur par défaut

UACI\_RHStaging

## crossSessionResponseTable

### Description

Nom de la table associée à ce niveau d'audience requis pour le suivi de réponse intersession dans les tables d'historique des contacts et des réponses accessibles à la fonctionnalité de suivi de réponse.

Si vous avez défini la propriété `schema` pour cette source de données, Unica Interact ajoute ce nom de table au schéma, par exemple, `schema.UACI_XSessResponse`. Si vous saisissez un nom complet, tel que `mySchema.UACI_XSessResponse`, Unica Interact n'ajoute pas le nom du schéma.

Si la journalisation des réponses de session croisée est désactivée, il n'est pas nécessaire de définir cette propriété.

### Valeur par défaut

UACI\_XSessResponse

## userEventLoggingTable

### Description

Ce paramètre désigne la table de base de données qui est utilisée pour consigner les activités liées aux événements définis par l'utilisateur. Les événements définis par l'utilisateur apparaissent dans l'onglet Événements des pages de synthèse Canal interactif, dans l'interface d'Unica Interact. La table de base de données que vous spécifiez ici contient des informations telles que l'ID de l'événement, son nom, sa fréquence pour ce niveau d'audience depuis le dernier vidage de la mémoire cache des activités d'événement, etc.

Si vous avez défini la propriété `schema` pour cette source de données, Unica Interact ajoute ce nom de table au schéma, par exemple, `schema.UACI_UserEventActivity`. Si vous saisissez un nom qualifié complet, tel que `mySchema.UACI_UserEventActivity`, Unica Interact n'ajoute pas le nom du schéma.

### Valeur par défaut

`UACI_UserEventActivity`

## patternStateTable

### Description

Ce paramètre désigne la table de base de données qui est utilisée pour consigner les états des modèles d'événement, par exemple si la condition du modèle a été satisfaite ou non, si le modèle est arrivé à expiration, a été désactivé, etc.

Si vous avez défini la propriété `schema` pour cette source de données, Unica Interact ajoute ce nom de table au schéma, par exemple, `schema.UACI_EventPatternState`. Si vous saisissez un nom qualifié complet, tel que `mySchema.UACI_EventPatternState`, Unica Interact n'ajoute pas le nom du schéma.

Un paramètre `patternStateTable` est requis pour chaque niveau d'audience même si vous n'utilisez pas de modèles d'événement. Le paramètre `patternStateTable` est basé sur la DDL de l'élément `UACI_EventPatternState` inclus. Voici un exemple dans lequel l'ID d'audience a deux composants : `ComponentNum` et `ComponentStr`.

```
CREATE TABLE UACI_EventPatternState_Composite
(
    UpdateTime bigint NOT NULL,
    State varbinary(4000),
    ComponentNum bigint NOT NULL,
    ComponentStr nvarchar(50) NOT NULL,
```

```
CONSTRAINT PK_CustomerPatternState_Composite PRIMARY KEY
( ComponentNum, ComponentStr, UpdateTime )
)
```

### Valeur par défaut

UACI\_EventPatternState

## Interact | profil | Niveaux d'audience | [Nom du niveau d'audience] | Offres par SQL brut

Ces propriétés de configuration vous permettent de définir les noms de table requis pour des fonctions Unica Interact. Vous ne devez définir le nom de la table que lorsque vous utilisez la fonctionnalité correspondante.

### enableOffersByRawSQL

#### Description

S'il est défini sur `True`, Unica Interact active la fonctionnalité `offersBySQL` pour ce niveau d'audience. 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`, Unica Interact n'utilise pas la fonctionnalité `offersBySQL`.

Si vous associez cette propriété à la valeur `true`, vous pouvez aussi configurer la propriété `Interact | profil | Niveaux d'audience | (Niveau d'audience) | Offres par SQL brut | Modèle SQL` afin de définir un ou plusieurs modèles SQL.

#### Valeur par défaut

Faux

#### 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**

Aucun

#### **Valeurs valides**

Nom du modèle SQL

### **nom**

#### **Catégorie de configuration**

```
Interact | profile | Audience Levels |  
[AudienceLevelName] | Offers by Raw SQL | (SQL Templates)
```

### 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

Aucun

## SQL

### Catégorie de configuration

```
Interact | profile | Audience Levels |  
[AudienceLevelName] | Offers by Raw SQL | (SQL Templates)
```

### 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}` s'appuie sur la session contenant une variable nommée `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 fonctionnalité. Les procédures mémorisées ne sont pas prises en charge ici.

### Valeur par défaut

Aucun



## Interact | profil | Niveaux d'audience | [NomNiveauAudience] | Modèle SQL

Ces propriétés de configuration vous permettent de définir un ou plusieurs modèles de requête SQL utilisés par la fonction `offersBySQL` de Unica Interact.

### nom

#### 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

Aucun

### 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}` s'appuie sur la session contenant une variable nommée `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 fonctionnalité. Les procédures mémorisées ne sont pas prises en charge ici.

#### Valeur par défaut

Aucun

## Interact | profil | Niveaux d'audience | [NomNiveauAudience | Services de données de profil | [Source de données]

Ces propriétés de configuration vous permettent de définir les noms de table requis pour des fonctions Unica Interact. Vous ne devez définir le nom de la table que lorsque vous utilisez la fonctionnalité 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 niveaux d'audience, et qui est préconfigurée 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

(Non disponible pour l'entrée 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 niveau d'audience.

#### Valeur par défaut

Aucun

#### Valeurs valides

Toute chaîne de texte.

### enabled

#### Description

Si vous définissez la valeur `True`, cette source de données est activée pour le niveau d'audience auquel elle est associée. Si vous définissez la valeur `False`, Unica Interact n'utilise pas cette source de données pour ce niveau d'audience.

#### Valeur par défaut

`True`

### Valeurs valides

True | False

## className

### Description

(Non disponible pour l'entrée 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 spécifiant un nom de classe qualifié complet.

## classPath

### Description

(Non disponible pour l'entrée Base de données par défaut.) 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 affichée, mais le chemin d'accès aux classes du serveur d'applications qui le contient est utilisé par défaut.

### Valeurs valides

Chaîne spécifiant le chemin d'accès aux classes.

## priorité

### Description

Priorité de cette source de données dans ce niveau d'audience. Il doit s'agir d'une valeur unique dans toutes les sources de données pour chaque niveau d'audience. (En d'autres termes, si une priorité est définie sur 100 pour une source de données, aucune autre source de données dans le niveau d'audience ne peut avoir une 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.

## Affinium|interact|profile|Audience Levels|[Audience Levels]requestLogTable

Cette propriété de configuration vous permet de déterminer le nom de la table d'enregistrement des demandes d'API ciblées sur ce niveau d'audience particulier. Si ce champ n'est pas renseigné, aucune journalisation d'API n'a lieu à ce niveau d'audience. Cette fonctionnalité doit être activée globalement pour consigner les demandes d'API pour ce niveau d'audience.

#### **Valeur par défaut**

chaîne vide

## Interact | offerserving

Ces propriétés de configuration définissent les propriétés de configuration d'apprentissage génériques. Pour régler votre mise en oeuvre d'apprentissage lorsque vous utilisez l'auto-apprentissage, utilisez les propriétés de configuration associées à l'environnement de conception.

### **offerTieBreakMethod**

#### **La description**

La propriété `offerTieBreakMethod` définit le comportement de la diffusion d'offres lorsque deux offres ont des scores équivalents (à égalité). Si vous associez cette propriété à la valeur par défaut `Random`, Unica Interact présente un choix aléatoire d'offres possédant un score équivalent. Si vous associez cette configuration à la valeur `Newer Offer`, Unica Interact propose l'offre la plus récente (celle qui possède l'ID d'offre le plus élevé) avant l'offre plus ancienne (ID d'offre moins élevé) si les scores des offres sont identiques.



#### Note:

Unica Interact a une fonctionnalité facultative qui permet à l'administrateur de configurer le système pour renvoyer les offres dans un ordre aléatoire indépendant du score, en définissant l'option `percentRandomSelection` (`Campaign | partitions | [partition_number] | Interact | learning | percentRandomSelection`). La propriété `offerTieBreakMethod` décrite ici est utilisée uniquement lorsque `percentRandomSelection` est défini sur zéro (désactivé).

#### Valeur par défaut

Aléatoire

#### Valeurs valides

Aléatoire | Offre plus récente

## optimizationType

#### La description

La propriété `optimizationType` définit si Unica Interact utilise un moteur d'apprentissage pour faciliter les affectations d'offres. Si défini sur `NoLearning`, Unica Interact n'utilise pas l'apprentissage. Si défini sur `BuiltInLearning`, Unica Interact utilise le moteur d'apprentissage Bayésien construit avec Unica Interact. Si défini sur `ExternalLearning`, Unica Interact utilise un moteur d'apprentissage que vous fournissez. Si vous sélectionnez

`ExternalLearning`, vous devez définir les propriétés `externalLearningClass` et `externalLearningClassPath`.

### **Valeur par défaut**

NoLearning

### **Valeurs valides**

NoLearning | BuiltInLearning | ExternalLearning

## **segmentationMaxWaitTimeInMS**

### **La 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**

### **La description**

Préfixe ajouté aux codes de traitement.

### **Valeur par défaut**

Aucune valeur par défaut définie.

## **effectiveDateBehavior**

### **La description**

Détermine si Unica Interact doit utiliser la date d'entrée en vigueur d'une offre lors du filtrage des offres qui sont présentées à un visiteur. Les valeurs sont les suivantes :

- -1 indique à Unica Interact d'ignorer la date d'entrée en vigueur sur l'offre.

0 indique à Unica Interact d'utiliser la date d'entrée en vigueur pour filtrer l'offre, de sorte que si la date d'entrée en vigueur de l'offre est antérieure ou égale à la date en cours, l'offre est présentée aux visiteurs.

Si une valeur **effectiveDateGracePeriod** est définie, la période de grâce est également appliquée pour déterminer si l'offre doit être diffusée.

- Tout entier positif indique à Unica Interact d'utiliser la date en cours plus la valeur de cette propriété pour déterminer s'il convient de présenter l'offre aux visiteurs, de sorte que si la date d'entrée en vigueur de l'offre est antérieure à la date en cours plus la valeur de cette propriété, l'offre est présentée aux visiteurs.

Si une valeur **effectiveDateGracePeriod** est définie, la période de grâce est également appliquée pour déterminer si l'offre doit être diffusée.

### Valeur par défaut

-1

## **effectiveDateGracePeriodOfferAttr**

### La description

Indique le nom de l'attribut personnalisé dans une définition d'offre qui indique le délai de grâce de la date d'entrée en vigueur. Par exemple, vous pouvez configurer cette propriété avec une valeur de `AltGracePeriod`. Vous définiriez ensuite des offres avec un attribut personnalisé appelé `AltGracePeriod` qui est utilisé pour spécifier le nombre de jours à utiliser comme période de grâce avec la propriété **effectiveDateBehavior**.

Supposons que vous créez un nouveau modèle d'offre avec une date d'effet de 10 jours à compter de la date actuelle et que vous incluez un attribut personnalisé appelé `AltGracePeriod`. Lorsque vous créez une offre à l'aide du modèle, si vous définissez la valeur de `AltGracePeriod` sur 14 jours, l'offre

sera proposée aux visiteurs, car la date actuelle se situe dans la période de grâce de la date d'effet de l'offre.

### **Valeur par défaut**

Page blanche

## **alwaysLogLearningAttributes**

### **La description**

Indique si Unica Interact doit écrire des informations sur les attributs de visiteur utilisés par le module d'apprentissage dans les fichiers journaux.

Notez que la définition de cette valeur sur true peut affecter les performances d'apprentissage et la taille des fichiers journaux.

### **Valeur par défaut**

**False**

Tenez compte des remarques suivantes :

- Si Learning est activé, pour la version 2, à l'aide de SampleMethod1, pour la version 1, à l'aide de SampleMethod1
- Lors du processus de configuration d'Interact, le système vérifie si les paramètres correspondent.
- Lors du processus d'optimiseur de traitement d'offre Interact dans Learning version 2, l'offre interactive est gérée en fonction des paramètres SampleMethod1 et SampleMethod2. Le RWA de l'offre est également calculé en fonction des paramètres SampleMethod1 et SampleMethod2.

## **includeArbitrationInfo**

Cette propriété de configuration détermine s'il faut inclure le résumé de l'arbitrage des offres dans la réponse des requêtes getOffers et getOffersForMultipleInteractionPoints.

### **Valeur par défaut**



False

### **Valeurs valides**

True | False

## **Interact | offerserving | Configuration de l'apprentissage intégré**

Ces propriétés de configuration définissent les paramètres d'écriture de la base de données pour les tables d'auto-apprentissage. Pour régler votre mise en œuvre d'apprentissage, utilisez les propriétés de configuration associées à l'environnement de conception.

### **version**

#### **Description**

Vous pouvez sélectionner 1 ou 2. La version 1 est la version de configuration de base qui n'utilise pas de paramètres pour définir des limites d'unité d'exécution et d'enregistrement. La version 2 est la version de configuration étendue qui permet de définir des paramètres d'unité d'exécution et d'enregistrement afin d'améliorer les performances. Ces paramètres effectuent une agrégation et une suppression lorsque les limites paramétrées sont atteintes.

#### **Valeur par défaut**

1

## **insertRawStatsIntervallnMinutes**

#### **Description**

Nombre de minutes d'attente du module d'apprentissage Unica Interact avant d'insérer des lignes supplémentaires dans les tables de transfert 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

### **Valeurs valides**

Entier positif

## **aggregateStatsIntervallInMinutes**

### **Description**

Nombre de minutes d'attente du module d'apprentissage Unica Interact entre l'agrégation des données dans les tables de statistiques 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.

## **autoAdjustPercentage**

### **Description**

Valeur qui détermine le pourcentage de données que l'exécution de l'agrégation tente de traiter en fonction des mesures de l'exécution précédente. Par défaut, cette valeur est égale à zéro, ce qui signifie que l'agrégateur traite tous les enregistrements de transfert et que cette fonctionnalité d'ajustement automatique est désactivée.

### **Valeur par défaut**

0

### **Valeurs valides**

Nombre compris entre 0 et 100.

## **excludeAbnormalAttribute**

### **Description**

Paramètre qui détermine s'il faut marquer ces attributs comme non valides. S'il a pour valeur `IncludeAttribute`, les attributs anormaux sont inclus sans être marqués comme non valides. S'il a pour valeur `ExcludeAttribute`, les attributs anormaux sont exclus et marqués comme non valides.

### **Valeur par défaut**

`IncludeAttribute`

### **Valeurs valides**

**`IncludeAttribute` | `ExcludeAttribute`**

## **saveOriginalValues**

### **Description**

Vous pouvez définir les valeurs suivantes: "Toutes les valeurs", "Valeurs regroupées" ou "Aucune". Cela permet de contrôler les valeurs consignées dans la table `UACI_LearningAttributeHist`.

Si le paramètre "Toutes les valeurs" est sélectionné, tous les attributs d'apprentissage seront consignés dans la table. Si ce paramètre est défini sur "Valeurs regroupées", seuls ces attributs seront consignés dans la table pour laquelle des casiers sont créés sous "Interact-> Apprentissage global".

Si le paramètre "Aucune" est défini, aucune valeur ne sera consignée dans `UACI_LearningAttributeHist`.

La valeur par défaut est "Aucune".

### **Valeur par défaut**

Aucun

### **Valeurs valides**

**Toutes les valeurs | Valeurs regroupées | Aucune**

## Interact | offerserving | Configuration de l'apprentissage intégré | Données de paramètre | [Nom du paramètre]

Ces propriétés de configuration définissent les paramètres associés à votre module d'apprentissage externe.

### **numberOfThreads**

#### **Description**

Nombre maximal d'unités d'exécution utilisées par l'agrégateur d'apprentissage pour traiter les données. Une valeur valide correspond à un entier positif et ne doit pas dépasser le nombre maximal de connexions configurées dans la source de données d'apprentissage. Ce paramètre n'est utilisé que par la version 2 de l'agrégateur.

#### **Valeur par défaut**

10

### **maxLogTimeSpanInMin**

#### **Description**

Si la version 1 de l'agrégateur est sélectionnée, vous pouvez traiter les enregistrements de transfert dans des itérations pour éviter les lots de base de données trop volumineux. Dans ce cas, ces enregistrements de transfert sont traités par blocs, itération par itération, dans un même cycle d'agrégation. La valeur de ce paramètre spécifie l'intervalle maximal entre les enregistrements de transfert que l'agrégateur tente de traiter dans chaque itération. Cet intervalle est basé sur la zone LogTime associée à chaque enregistrement de transfert et seuls les enregistrements dont la zone LogTime est comprise dans la première fenêtre de temps sont traités. Une valeur valide correspond à un entier non négatif. Si la valeur est égale à 0, il n'y a pas de limite, ce qui signifie que tous les enregistrements de transfert sont traités dans une même itération.

#### **Valeur par défaut**

0

## **maxRecords**

### **Description**

Si la version 2 de l'agrégateur est sélectionnée, vous pouvez traiter les enregistrements de transfert dans des itérations pour éviter les lots de base de données trop volumineux. Dans ce cas, ces enregistrements de transfert sont traités par blocs, itération par itération, dans un même cycle d'agrégation. La valeur de ce paramètre spécifie le nombre maximal d'enregistrements de transfert que l'agrégateur tente de traiter dans chaque itération. Une valeur valide correspond à un entier non négatif. Si la valeur est égale à 0, il n'y a pas de limite, ce qui signifie que tous les enregistrements de transfert sont traités dans une même itération.

### **Valeur par défaut**

0

## **valeur**

### **Description**

Valeur d'un paramètre requis par la classe d'un module d'apprentissage intégré.

### **Valeur par défaut**

Aucune valeur par défaut définie.

## **Interact | offerserving | Configuration de l'apprentissage externe**

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 la propriété `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 la propriété `optimizationType` est définie sur `ExternalLearning`, définissez `externalLearningClass` sur le classpath associé au moteur d'apprentissage externe.

Le classpath 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 Unica Platform, une copie du fichier jar doit être présente au même emplacement de chaque serveur. Le classpath 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, sur 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 à la classe

### Valeur par défaut

Aucune valeur par défaut définie.

## Disponibilité

Cette propriété s'applique uniquement si la propriété `optimizationType` est définie sur `ExternalLearning`.

## Interact | offerserving | Configuration de l'apprentissage externe | Données de paramètre | [Nom du paramètre]

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 | offerserving | Contraintes

Ces propriétés de configuration définissent les contraintes appliquées au processus de service d'offres.

### **maxOfferAllocationInMemoryPerInstance**

#### Description

Taille d'un bloc d'offres. Unica Interact conserve un pool d'offres en mémoire de sorte que le système n'ait pas à interroger la base de données à chaque fois qu'une offre est renvoyée. Chaque fois qu'une offre est renvoyée, le pool

est ajusté. Lorsque le pool est épuisé, Unica Interact obtient un autre bloc d'offres pour remplir le pool.

**Valeur par défaut**

1000

**Valeurs valides**

Un nombre entier supérieur à 0.

**maxDistributionPerIntervalPerInstanceFactor**

**Description**

Pourcentage de contrainte pour une allocation d'offre donnée nécessaire pour qu'un serveur d'exécution prenne en charge la distribution sur les serveurs d'exécution.

**Valeur par défaut**

100

**Valeurs valides**

Un nombre entier entre 0 et 100.

**constraintCleanupIntervalInDays**

**Description**

Fréquence à laquelle les nombres désactivés de la table UACI\_OfferCount sont nettoyés. Une valeur inférieure à 1 désactive cette fonction.

**Valeur par défaut**

7

**Valeurs valides**

Un nombre entier supérieur à 0.



## 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 génération 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'Unica 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`.

### **Affinium|interact|services|contactHist|treatmentStoreReference**

Ce paramètre de configuration est le nœud racine pour les paramètres liés au magasin de données des traitements récemment servis.

### **daysBackForXSessContact**

Le nombre de jours au cours desquels un traitement servi est conservé dans le magasin de données pour une recherche de session croisée. Si la valeur n'est pas positive ou égale à zéro, la fonction de suivi de contact de session croisée est désactivée.

#### **Valeur par défaut**

0

## Valeur valide

N'importe quel nombre positif

## Interact | services | contactHist

Les propriétés de configuration de cette catégorie définissent les paramètres associés au service de collecte des données pour les tables de transfert d'historique des contacts.

### enableLog

#### Description

Si la valeur est définie sur `true`, 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 `false`, 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 (`Mémoire cache`) ou dans un fichier (`External Loader file`). Vous pouvez uniquement utiliser `External Loader File` si vous avez configuré Unica 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

Cache mémoire | External Loader File

## Interact | services | contactHist | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de cache associés au service de collecte des données pour la table de transfert d'historique des contacts. **Remarque** : Lorsque contactHist ou responseHist est configuré pour utiliser memoryCache, vous pouvez créer une source de données systemTablesDataSource et configurer les paramètres sous Affinium|interact|général|systemTablesDataSource|loaderProperties. Ensuite, les enregistrements de transfert de l'historique de contact/réponse seront conservés dans les fichiers dans le répertoire défini par Affinium|interact|services|externalLoaderStagingDirectory si la conservation dans la base de données échoue. Autrement, des entrées INFO seront consignées lors de l'initialisation indiquant que la reprise n'est pas activée.

## threshold

### 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 | contactStatusCodes

Les propriétés de configuration de cette catégorie définissent les paramètres du type de statut de contact personnalisé qui peuvent être transmis dans Interact avec des événements de contact.

### Nouveau nom de la catégorie

#### Description

Cette propriété définit le nom de la catégorie de code du statut de contact.

### Code

#### Description

Cette propriété définit le code personnalisé pour votre type de contact.  
Ce code définit doit exister dans la table système HCL Campaign  
UA\_ContactStatus.

### action

#### Description

Action correspondant au code de type de contact personnalisé. L'action définie ici va remplacer l'action définie dans la table système HCL Campaign  
UA\_ContactStatus.

#### Valeur par défaut

Aucun

#### Valeur valide

LogContact | Aucun

## Interact | services | contactHist | fileCache

Les propriétés de configuration de cette catégorie définissent les paramètres de cache associés au service de collecte des données d'historique des contacts en cas d'utilisation d'un utilitaire de chargement. **Prérequis** : pour Configuration Affinium|interact|services|externalLoaderStagingDirectory set loaderStagingData.

## threshold

### 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 `true`, 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 `false`, aucune statistique n'est collectée.

La collecte de données n'étant pas requise, vous pouvez définir cette propriété sur `false` si vous n'utilisez pas la génération 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 cache associés au service qui collecte les statistiques relatives au nombre d'utilisations de la chaîne par défaut du point d'interaction.

### threshold

#### 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 | statsOpsElig

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 `true`, le service qui collecte les statistiques relatives aux offres éligibles est activé. Si la valeur est définie sur `false`, aucune statistique n'est collectée.

La collecte de données n'étant pas requise, vous pouvez définir cette propriété sur `false` si vous n'utilisez pas la génération de rapports d'IBM.

### **Valeur par défaut**

True

### **Valeurs valides**

True | False

## Interact | services | tatsOpsElig | cache

Les propriétés de configuration de cette catégorie définissent les paramètres de cache associés au service qui collecte les statistiques relatives aux offres éligibles.

### **threshold**

#### **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 `true`, le service qui collecte les statistiques relatives à l'activité des événements est activé. Si la valeur est définie sur `false`, aucune statistique n'est collectée.

La collecte de données n'étant pas requise, vous pouvez définir cette propriété sur `false` si vous n'utilisez pas la génération 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 cache associés au service qui collecte les statistiques relatives à l'activité des événements.

### threshold

#### 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 | eventPattern

Les propriétés de configuration de la catégorie `eventPattern` définissent les paramètres du service qui collecte les statistiques relatives à l'activité des modèles d'événement.

## persistUnknownUserStates

### Description

Détermine si les états de modèle d'événement d'un ID d'audience inconnu (visiteur) sont conservés dans la base de données. Par défaut, lorsqu'une session se termine, les états de tous les modèles d'événement mis à jour qui sont associés à l'ID d'audience du visiteur sont stockés dans la base de données, à condition que l'ID d'audience soit connu (c'est-à-dire que le profil du visiteur puisse être trouvé dans la source de données des profils).

La propriété `persistUnknownUserStates` détermine le comportement si l'ID d'audience est inconnu. Par défaut, elle a pour valeur `False`, et pour les ID d'audience inconnus, les états de modèle d'événement sont effacés à la fin de la session.

Si vous associez cette propriété à la valeur `True`, les états de modèle d'événement des utilisateurs inconnus (dont le profil est introuvable dans le service de données des profils configuré) sont conservés.

### Valeur par défaut

False

### Valeurs valides

True | False

## **mergeUnknowUserInSessionStates**

### **Description**

Détermine la façon dont les états de modèle d'événement des ID d'audience (visiteurs) inconnus sont conservés. Si l'ID d'audience change au cours d'une session, Unica Interact tente de charger les états de modèle d'événement sauvegardés pour le nouvel ID d'audience depuis la table de base de données. Si l'ID d'audience précédent était inconnu et que vous avez associé la propriété `mergeUnknowUserInSessionStates` à la valeur `True`, les activités des événements utilisateur appartenant à l'ID d'audience précédent de la même session sont fusionnées dans le nouvel ID d'audience.

### **Valeur par défaut**

False

### **Valeurs valides**

True | False

## **enableUserEventLog**

### **Description**

Détermine si les activités des événements utilisateur sont journalisées dans la base de données.

### **Valeur par défaut**

False

### **Valeurs valides**

True | False

## **Interact | services | eventPattern | userEventCache**

Les propriétés de configuration de la catégorie `userEventCache` définissent les paramètres qui déterminent à quel moment une activité des événements est déplacée depuis le cache pour être conservée dans la base de données.

## **threshold**

### **Description**

Détermine le nombre maximal d'états de modèle d'événement pouvant être stocké dans le cache des états de modèle d'événement. Une fois la limite atteinte, les états utilisés le moins récemment sont effacés du cache.

### **Valeur par défaut**

100

### **Valeurs valides**

Nombre souhaité d'états de modèle d'événement à conserver dans le cache.

## **insertPeriodInSecs**

### **Description**

Détermine la durée maximale, en secondes, pendant laquelle les activités des événements utilisateur sont mises en file d'attente dans la mémoire. Une fois que la durée maximale définie par cette propriété est atteinte, ces activités sont conservées dans la base de données.

### **Valeur par défaut**

3 600 (60 minutes)

### **Valeurs valides**

Nombre de secondes de votre choix.

## **Interact | services | eventPattern | advancedPatterns**

Les propriétés de configuration de cette catégorie contrôlent si l'intégration à Unica Interact Advanced Patterns est activée et elles définissent les intervalles de délai d'expiration pour les connexions avec Unica Interact Advanced Patterns.

## **enableAdvancedPatterns**

### **Description**

Si la valeur est `true`, l'intégration à Unica Interact Advanced Patterns est activée. Si la valeur est `false`, l'intégration n'est pas activée. Si l'intégration a été préalablement activée, Unica Interact utilise les derniers états de modèle reçus de Unica Interact Advanced Patterns.

#### **Valeur par défaut**

True

#### **Valeurs valides**

True | False

### **connectionTimeoutInMilliseconds**

#### **Description**

Temps maximal possible pour établir une connexion HTTP à partir de l'environnement en temps réel Unica Interact avec Unica Interact Advanced Patterns. En cas d'expiration du délai de la requête, Unica Interact utilise les dernières données sauvegardées des modèles.

#### **Valeur par défaut**

30

### **readTimeoutInMilliseconds**

#### **Description**

Une fois qu'une connexion HTTP est établie entre l'environnement en temps réel Unica Interact et Unica Interact Advanced Patterns, et qu'une requête est envoyée à Unica Interact Advanced Patterns pour obtenir le statut d'un modèle d'événement, temps maximal nécessaire pour recevoir des données. En cas d'expiration du délai de la requête, Unica Interact utilise les dernières données sauvegardées des modèles.

#### **Valeur par défaut**

100

## connectionPoolSize

### Description

Taille du pool de connexions HTTP pour la communication entre l'environnement en temps réel Unica Interact et Unica Interact Advanced Patterns.

### Valeur par défaut

10

Interact | services | eventPattern | advancedPatterns | autoReconnect

Les propriétés de configuration dans cette catégorie spécifient des paramètres pour la fonction de reconnexion automatique dans l'intégration avec Unica Interact Advanced Patterns.

## activer

### Description

Détermine si le système se reconnecte automatiquement si des problèmes de connexion se produisent entre l'environnement en temps réel Unica Interact et Unica Interact Advanced Patterns. La valeur par défaut **True** active cette fonction.

### Valeur par défaut

True

### Valeurs valides

True | False

## durationInMinutes

### Description

Cette propriété spécifie l'intervalle de temps durant lequel le système évalue les problèmes de connexion répétés entre l'environnement en temps réel Unica Interact et Unica Interact Advanced Patterns.

**Valeur par défaut**

10

**numberOfFailuresBeforeDisconnect**

**Description**

Cette propriété indique le nombre d'échecs de connexion autorisés pendant la période spécifiée avant que le système se déconnecte automatiquement de Unica Interact Advanced Patterns.

**Valeur par défaut**

3

**consecutiveFailuresBeforeDisconnect**

**Description**

Détermine si la fonction de reconnexion automatique évalue uniquement des échecs consécutifs de la connexion entre l'environnement en temps réel Unica Interact avec Unica Interact Advanced Patterns. Si vous définissez cette valeur sur **False**, tous les échecs dans l'intervalle de temps spécifié sont évalués.

**Valeur par défaut**

True

**sleepBeforeReconnectDurationInMinutes**

**Description**

Le système attend pendant le nombre de minutes spécifié dans cette propriété avant de se reconnecter après la déconnexion du système en raison d'échecs répétés comme indiqué dans les autres propriétés de cette catégorie.

**Valeur par défaut**

5

## sendNotificationAfterDisconnect

### Description

Cette propriété détermine si le système envoie une notification électronique lorsqu'un incident de connexion se produit. Le message de notification inclut le nom d'instance en temps réel Unica Interact pour laquelle un échec s'est produit et la durée après laquelle la reconnexion a lieu, comme indiqué dans la propriété **sleepBeforeReconnectDurationInMinutes**. La valeur par défaut **True** signifie que des notifications sont envoyées.

### Valeur par défaut

True

## 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 `true`, la fonctionnalité de conversion du journal personnalisé en table est activée. Si la valeur est définie sur `false`, 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 cache associés au service qui collecte des données personnalisées en vue de les convertir en table (événement qui utilise le paramètre d'événement `UACICustomLoggerTableName`).

### threshold

#### 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 de transfert d'historique des réponses.

### enableLog

#### Description

Si la valeur est définie sur `true`, le service qui écrit dans les tables de transfert d'historique des réponses est activé. Si la valeur est définie sur `false`, aucune donnée n'est écrite.



La table de transfert de l'historique des réponses est définie par la propriété de niveau d'audience `responseHistoryTable`. La valeur par défaut est `UACI_RHStaging`.

### Valeur par défaut

True

### Valeurs valides

True | False

## cacheType

### Description

Définit si les données du cache sont gardées en mémoire ou dans un fichier. Vous ne pouvez utiliser `External Loader File` que si vous avez configuré Unica 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

Cache mémoire | External Loader File

## actionOnOrphan

### Description

Ce paramètre détermine le traitement qui doit être apporté aux événements de réponse qui n'ont pas d'événements de contact associés actuellement publiés. Le paramètre s'applique aux événements de réponse en session. Si ce paramètre a pour valeur `NoAction`, l'événement de réponse est traité comme si l'événement de contact correspondant avait été publié. Si ce paramètre a pour valeur `Warning`, l'événement de réponse est traité comme si l'événement de

contact correspondant avait été publié, mais qu'un message d'avertissement avait été copié dans `interact.log`. Si ce paramètre a pour valeur `skip`, l'événement de réponse n'est pas traité et un message d'erreur est copié dans `interact.log`. Le paramétrage que vous choisissez ici est effectif que la journalisation de l'historique des réponses soit activée ou non.

### Valeur par défaut

NoAction

### Valeurs valides

Aucune action | Avertissement | Ignorer

## suppressionActionOnResponse

### Description

Ce paramètre gère la suppression d'une offre à laquelle a répondu un événement de réponse dans une session. Les quatre options suivantes sont disponibles :

- NoSuppression Ne pas supprimer cette offre
- SuppressionTillAudienceChange Cette offre est supprimée jusqu'à ce que l'ID d'audience active dans cette session change.
- SuppressionForAudience Cette offre est supprimée tant que l'ID d'audience active dans cette session est identique à celui utilisé au moment où l'offre a été retournée.
- SuppressionInSession Cette offre est supprimée tout au long de cette session, même si l'ID d'audience change.

Voici un exemple avec une séquence d'API.

1. startSession (audience 1)
2. getOffers -> retourner l'offre A
3. postEvent (contact de l'offre A)
4. postEvent (accepter ou rejeter l'offre A)
5. Obtenir des offres

- 6. setAudience (audience 2)
- 7. Obtenir des offres
- 8. setAudience (audience 1)
- 9. Obtenir des offres

**Valeur par défaut**

SuppressionTillAudienceChange

**Valeurs valides**

NoSuppression | SuppressionTillAudienceChange| SuppressionForAudience|  
SuppressionInSession

Le tableau suivant indique si l'offre A est supprimée dans les étapes 5, 7 et 9.

Paramètre	Etape 5	Etape 7	Etape 9
NoSuppression	N	N	N
SuppressionTillAudienceChange	Y	N	N
SuppressionForAudience	Y	N	Y
SuppressionInSession	Y	Y	Y

**Interact | services | responseHist | cache**

Les propriétés de configuration de cette catégorie définissent les paramètres de cache associés au service qui collecte les données d'historique des réponses. **Remarque :** Lorsque contactHist ou responseHist est configuré pour utiliser memoryCache, vous pouvez créer une source de données systemTablesDataSource et configurer les paramètres sous Affinium|interact|général|systemTablesDataSource|loaderProperties. Ensuite, les enregistrements de transfert de l'historique de contact/réponse seront conservés dans les fichiers dans le répertoire défini par Affinium|interact|services|externalLoaderStagingDirectory si la conservation dans la base de données échoue.

Autrement, des entrées INFO seront consignées lors de l'initialisation indiquant que la reprise n'est pas activée.

## **threshold**

### **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 | response Hist | responseTypeCodes**

Les propriétés de configuration de cette catégorie définissent les paramètres du service d'historique des réponses.

## **Nouveau nom de la catégorie**

### **Description**

Nom de votre code de type de réponse.

## **code**

### **Description**

Code personnalisé de votre type de réponse.

### **Valeur par défaut**

Code personnalisé ajouté à la table UA\_UsrResponseType.

## **action**

### **Description**

Action correspondant au code de type de réponse personnalisé.

L'action définie pour l'événement publié remplace l'action définie ici. Par conséquent, si un événement logAccept est publié sans code de type de réponse, cet événement est traité en tant qu'événement d'acceptation. Si un événement logAccept est publié avec un code de type de réponse existant dans cette configuration, l'action configurée est utilisée pour déterminer s'il s'agit d'un événement d'acceptation. Si un événement logAccept est publié avec un code de type de réponse qui n'existe pas dans cette configuration, cet événement n'est pas traité en tant qu'événement d'acceptation. Lorsqu'un événement est traité en tant qu'événement d'acceptation, les statistiques d'apprentissage sont mises à jour en conséquence si l'apprentissage est activé. Les règles d'expression de l'offre sont évaluées si l'une d'entre elles est basée sur l'acceptation de cette offre.

### **Valeur par défaut**

Aucun

### **Valeurs valides**

LogAccept | LogReject | None

## **Interact | services | responseHist | fileCache**

Les propriétés de configuration de cette catégorie définissent les paramètres de cache associés au service qui collecte les données d'historique des réponses en cas d'emploi d'un utilitaire de chargement.

## **threshold**

### **Description**

Nombre d'enregistrements accumulés avant que le Unica Interact les écrive dans la base de données.

`responseHist` - La table définie par la propriété de niveau d'audience `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 ne devez configurer ces paramètres que si vous utilisez le suivi de réponse intersession d'Unica Interact.

### **enableLog**

#### **Description**

Si la valeur est définie sur `true`, le service `crossSessionResponse` est activé et Unica Interact écrit des données dans les tables de transfert de suivi de réponse intersession. Si la valeur est définie sur `false`, le service `crossSessionResponse` est désactivé.

#### **Valeur par défaut**

False

### **xsessionProcessIntervallInSecs**

#### **Description**

Nombre de secondes entre chaque exécution du processus intersession. Ce processus déplace les données des tables de transfert de l'historique des réponses intersession dans la table de transfert de l'historique des réponses et dans le module d'auto-apprentissage.

#### **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, Unica Interact associe la valeur -1 à la réponse dans la colonne `Mark` de la table de transfert `xSessResponse`. Vous pouvez alors faire correspondre ou supprimer ces réponses manuellement.

#### **Valeur par défaut**

180

### **xsessionResponseBatchsize**

#### **Description**

Le nombre d'enregistrements de réponse de session croisée à traiter en même temps. Au lieu de traiter tous les nouveaux enregistrements ou les nouvelles tentatives d'enregistrement à la fois, le système génère une boucle à travers les enregistrements `xsessionResponseBatchsize` en une fois. Il s'agit d'un changement de performance, étant donné que le traitement simultané d'un grand nombre d'enregistrements peut entraîner une certaine lenteur.

### **Valeur par défaut**

10000

### **Valeurs valides**

Tout entier supérieur à zéro.

## **generateOnlyOneResponseRecord**

### **Description**

Lorsqu'une réponse de session croisée est traitée, Interact doit la lier aux enregistrements de l'historique de contact disponible. Il arrive parfois que plusieurs enregistrements d'historique de contact correspondants soient détectés en fonction des critères données (code de traitement ou d'ID d'offre). Dans ce cas, Interact utilise le paramètre de configuration "generateOnlyOneResponseRecord" afin de décider du résultat

### **Valeurs**

- True : Seul un enregistrement de l'historique de réponse est généré à l'aide de l'enregistrement de l'historique de contact le plus récent.
- False : Un enregistrement de l'historique de réponse est généré pour chaque enregistrement de l'historique de contact correspondant.

### **Valeur par défaut**

False

## **Interact | services | crossSessionResponse | cache**

Les propriétés de configuration de cette catégorie définissent les paramètres de cache associés au service de collecte des données de réponses intersessions.

## **threshold**

### **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 | [Niveau d'audience] | TrackingCodes | byTreatmentCode**

Les propriétés de cette section définissent la méthode appliquée par le suivi de réponse intersession pour faire correspondre des codes de traitement à l'historique des contacts et des réponses.

**SQL**

**Description**

Cette propriété définit si Unica Interact doit utiliser l'instruction Utiliser SQL généré par le système ou l'instruction SQL personnalisée définie dans la propriété `OverrideSQL`.

**Valeur par défaut**

Utiliser SQL généré par le système

**Valeurs valides**

Utiliser SQL généré par le système | 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 `Utiliser SQL généré par le système`.

### 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 | [Niveau d'audience] | TrackingCodes | byOfferCode

Dans cette section, les propriétés définissent la méthode appliquée par le suivi de réponse intersession pour faire correspondre des codes d'offre à l'historique des contacts et des réponses.

## SQL

### Description

Cette propriété définit si Unica Interact doit utiliser l'instruction Utiliser SQL généré par le système ou l'instruction SQL personnalisée définie dans la propriété `OverrideSQL`.

### Valeur par défaut

Utiliser SQL généré par le système

### Valeurs valides

Utiliser SQL généré par le système | 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 `Utiliser SQL généré par le système`.

### Valeur par défaut

## useStoredProcedure

### Description

Si la valeur est définie sur true, 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`.

## nteract | services | crossSessionResponse | OverridePerAudience | [Niveau d'audience] | TrackingCodes | byAlternateCode

Dans cette section, les propriétés définissent la méthode appliquée par le suivi de réponse intersession pour faire correspondre un code alternatif défini par l'utilisateur à l'historique des contacts et des réponses.

## Nom

### 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 unités d'exécution associés aux services qui collectent les données pour les tables de transfert d'historique des contacts et des réponses.

### **corePoolSize**

#### **Description**

Nombre d'unités d'exécution 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 d'unités d'exécution est supérieur au nombre principal, il s'agit du laps de temps maximum pendant lequel les unités d'exécution inactives 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 d'unités d'exécution pour collecter les données d'historique des contacts et des réponses.

**Valeur par défaut**

1000

## **termWaitSecs**

**Description**

Lors de l'arrêt 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 d'unités d'exécution 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 d'unités d'exécution est supérieur au nombre principal, il s'agit du laps de temps maximum pendant lequel les unités d'exécution inactives en trop associées 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 d'unités d'exécution 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 l'arrêt du serveur d'exécution, laps de temps à attendre, en secondes, pour que les unités d'exécution de service associées 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 unités d'exécution qui écrivent les données de cache collectées dans les tables des bases de données de l'environnement d'exécution.

**corePoolSize**

**Description**

Nombre d'unités d'exécution 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 d'unités d'exécution à conserver dans le pool pour les unités d'exécution planifiées 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 unités d'exécution inactives en trop destinés aux unités d'exécution planifiées 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 unités d'exécution planifiées qui écrivent les données de cache dans le magasin de données.

### **Valeur par défaut**

1000

## **termWaitSecs**

### **Description**

Lors de l'arrêt du serveur d'exécution, laps de temps à attendre, en secondes, pour que les unités d'exécution de service associés aux unités d'exécution planifiés qui écrivent les données de cache dans le magasin de données prennent fin.

**Valeur par défaut**

5

## Interact | services | threadManagement | eventHandling

Les propriétés de configuration de cette catégorie définissent les paramètres de gestion des unités d'exécution associés aux services qui collectent des données pour la gestion des événements.

### **corePoolSize**

**Description**

Nombre d'unités d'exécution à conserver dans le pool, qu'ils soient actifs ou non, pour la collecte des données de gestion des événements.

**Valeur par défaut**

1

### **maxPoolSize**

**Description**

Nombre maximal d'unités d'exécution à conserver dans le pool pour les services qui collectent les données de gestion des événements.

**Valeur par défaut**

5

### **keepAliveTimeSecs**

**Description**

Lorsque le nombre d'unités d'exécution est supérieur au nombre principal, il s'agit du laps de temps maximum pendant lequel les unités d'exécution

inactives en trop attendent que les nouvelles tâches prennent fin en vue de collecter les données de gestion des événements.

#### **Valeur par défaut**

5

### **queueCapacity**

#### **Description**

Taille de la file d'attente utilisée par le pool d'unités d'exécution pour collecter les données de gestion des événements.

#### **Valeur par défaut**

1000

### **termWaitSecs**

#### **Description**

Lors de l'arrêt du serveur d'exécution, laps de temps à attendre, en secondes, pour que les unités d'exécution de service terminent de collecter les données de gestion des événements.

#### **Valeur par défaut**

5

## **Interact | services | configurationMonitor**

Les propriétés de configuration de cette catégorie vous permettent d'activer et de désactiver l'intégration à Unica Interact Advanced Patterns sans avoir à redémarrer Unica Interact en temps réel et elles permettent de définir l'intervalle d'interrogation de la valeur de propriété qui active l'intégration.

### **activer**

#### **Description**

Si la valeur est réglée sur `true`, active le service qui actualise la valeur de la propriété **Interact | services | eventPattern | advancedPatterns**

**enableAdvancedPatterns.** Si la valeur est réglée sur `false`, vous devez redémarrer Unica Interact en temps réel lorsque vous modifiez la valeur de la propriété **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns.**

#### Valeur par défaut

False

#### Valeurs valides

True | False

### refreshIntervallInMinutes

#### Description

Définit l'intervalle de temps pour l'interrogation de la valeur de la propriété **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns.**

#### Valeur par défaut

5

### Interact | services | CampaignSegments

#### isEnabled

##### Description

Cette fonction est activée si elle est définie sur `True`.

Les méthodes (`startSession` et `setAudience`) qui déclenchent la segmentation effectuent un appel d'API Campaign afin d'obtenir les segments de Campaign pour l'ID audience.

##### Valeur par défaut

`False`

##### Valeurs valides

True | False

## ServiceURL

### Description

L'URL de service Campaign, comme l'un des suivants.

### Valeur par défaut

http://localhost:7001/Campaign

## corePoolSize

### Description

Nombre d'unités d'exécution à conserver dans le pool, qu'ils soient actifs ou non, afin d'obtenir les segments de Campaign.

### Valeur par défaut

5

## maxPoolSize

### Description

Nombre maximum d'unités d'exécution à conserver dans le pool pour les unités d'exécution planifiées afin d'obtenir les segments de Campaign.

### Valeur par défaut

5

## keepAliveTimeSecs

### Description

Lorsque le nombre d'unités d'exécution est supérieur au nombre principal, il s'agit du laps de temps maximum pendant lequel les unités d'exécution inactives en trop attendent que les nouvelles tâches prennent fin en vue d'obtenir les segments de Campaign.

### Valeur par défaut

5

## queueCapacity

### Description

Taille de la file d'attente utilisée par le pool d'unités d'exécution en vue d'obtenir les segments de Campaign.

### Valeur par défaut

1000

## readTimeoutInMilliseconds

### Description

Par défaut, l'appel de service de Campaign est asynchrone. Si un paramètre `UACIWaitForSegmentation` avec une valeur `true` est transmis via un appel API, cet appel API est appelé de manière synchrone et patiente pendant la durée spécifiée par ce paramètre.

### Valeur par défaut

50

## Interact | cacheManagement

Cet ensemble de propriétés de configuration vous permet de définir les paramètres de sélection et de configuration de chacun des gestionnaires de cache pris en charge que vous pouvez utiliser pour améliorer les performances d'Unica Interact, tels que le mécanisme EHCACHE, qui est intégré dans l'installation d'Unica Interact. Utilisez les propriétés de configuration **Unica Interact | cacheManagement | Gestionnaires de cache** pour configurer le gestionnaire de cache que vous souhaitez utiliser. Utilisez les propriétés de configuration **Unica Interact | cacheManagement | caches** pour spécifier le gestionnaire de cache qui doit être utilisé par Unica Interact pour améliorer les performances.

## Interact | cacheManagement | Gestionnaires de cache

La catégorie Cache Managers vous permet de spécifier les paramètres relatifs aux solutions de gestion de cache que vous prévoyez d'utiliser avec Unica Interact.

## Interact | cacheManagement | Gestionnaires de cache | EHCACHE

La catégorie EHCACHE vous permet de spécifier les paramètres relatifs à la solution de gestion de cache EHCACHE ; vous pouvez ainsi la personnaliser afin d'améliorer les performances d'Unica Interact.

## Interact | Gestionnaires de cache | EHCACHE | Données de paramètre

Les propriétés de configuration de cette catégorie permettent de spécifier de quelle façon le système de gestion de cache EHCACHE fonctionne pour améliorer les performances d'Unica Interact.

### cacheType

#### Description

Vous pouvez configurer les serveurs d'exécution Unica Interact dans un groupe de serveurs pour qu'ils utilisent une adresse de multidiffusion afin de partager les données du cache. Cette méthode est également appelée cache distribué. Le paramètre cacheType vous permet de spécifier si vous utilisez le mécanisme de mise en cache EHCACHE intégré en mode **local** (autonome) ou **distribué** (comme avec un groupe de serveurs d'exécution).



#### Remarque :

Si vous sélectionnez la valeur **Distributed** pour le paramètre cacheType, tous les serveurs qui partagent le cache doivent faire partie d'un seul et même groupe de serveurs. Vous devez également activer la multidiffusion entre les membres d'un groupe de serveurs.

#### Valeur par défaut



Local

### Valeurs valides

Local | Distributed

## multicastIPAddress

### Description

Si vous spécifiez la valeur Distributed au paramètre **cacheType**, vous configurez le cache pour qu'il fonctionne via la multidiffusion entre tous les membres d'un groupe de serveurs d'exécution Unica Interact. La valeur **multicastIPAddress** correspond à l'adresse IP qui est utilisée par tous les serveurs Unica Interact du groupe de serveurs en mode écoute.

Cette adresse IP doit être unique parmi les groupes de serveurs.

### Valeur par défaut

230.0.0.1

## multicastPort

### Description

Si vous spécifiez la valeur Distributed au paramètre **cacheType**, le paramètre **multicastPort** indique le port qui est utilisé par tous les serveurs Unica Interact du groupe de serveurs en mode écoute.

### Valeur par défaut

6363

## overflowToDisk

### Description

Le gestionnaire de cache EHCACHE gère les informations de session à l'aide de la mémoire disponible. Pour les environnements dans lesquels la taille de la session est importante en raison d'un profil volumineux, le nombre de sessions qui doit être pris en charge dans la mémoire n'est peut-être pas suffisant pour le scénario client. Pour pallier à cette situation, le gestionnaire

de cache EHCache dispose d'une fonction facultative qui permet d'écrire temporairement sur le disque dur les informations de cache dont la taille est supérieure à la quantité d'informations qui peut être conservée en mémoire.

Si vous affectez la valeur "yes" à la propriété **overflowToDisk**, chaque machine virtuelle Java™ peut gérer un nombre de sessions simultanées supérieur au nombre autorisé par la mémoire.

#### **Valeur par défaut**

Non

#### **Valeurs valides**

No | Yes

## **diskStore**

#### **Description**

Lorsque la propriété de configuration **overflowToDisk** a pour valeur `yes`, cette propriété de configuration spécifie le répertoire de disque qui contiendra les entrées de cache dépassant de la mémoire. Si cette propriété de configuration n'existe pas ou si sa valeur n'est pas valide, le répertoire de disque est créé automatiquement dans le répertoire temporaire par défaut du système d'exploitation.

#### **Valeur par défaut**

Aucun

#### **Valeurs valides**

Répertoire auquel l'application Web hébergeant le module d'exécution Unica Interact peut accéder en écriture.

## **(Parameter)**

#### **Description**

Modèle que vous pouvez utiliser pour créer un paramètre personnalisé à utiliser avec le gestionnaire de cache. Vous pouvez définir n'importe quel nom de paramètre et associer à celui-ci n'importe quelle valeur.

Pour créer un paramètre personnalisé, cliquez sur **(Parameter)** et indiquez le nom et la valeur à affecter au paramètre. Lorsque vous cliquez sur **Enregistrer les modifications**, le paramètre que vous avez créé est ajouté à la liste dans la catégorie Parameter Data.

### Valeur par défaut

Aucun

## Interact | caches

Utilisez cet ensemble de propriétés de configuration pour spécifier le gestionnaire de cache pris en charge que vous souhaitez utiliser pour améliorer les performances d'Unica Interact, par exemple la mise en cache Ehcache ou Ignite, et pour configurer les propriétés de cache spécifiques du serveur d'exécution que vous configurez.

Ces propriétés concernent notamment les caches utilisés pour le stockage des données de session, des états de modèle d'événement et des résultats de segmentation. Lorsque vous définissez ces paramètres, vous pouvez spécifier la solution de cache à utiliser pour chaque type de mise en cache, ainsi que des paramètres individuels relatifs au fonctionnement du cache.

## Interact | cacheManagement | caches | InteractCache

La catégorie InteractCache permet de configurer la mise en cache de tous les objets de session, y compris les données de profil, les résultats de segmentation, les derniers traitements distribués, les paramètres transmis via des méthodes API et les autres objets utilisés par le module d'exécution Unica Interact.

La catégorie InteractCache est nécessaire au fonctionnement correct d'Interact.

Cette catégorie peut également être configurée via une configuration externe EHCACHE pour les paramètres qui ne sont pas pris en charge dans **Interact | cacheManagement | Caches**. Si vous utilisez EHCACHE, vous devez vérifier qu'InteractCache est correctement configuré.

## CacheManagerName

### Description

Nom du gestionnaire de cache qui gère le cache Unica Interact. La valeur que vous entrez dans cette zone doit correspondre à l'un des gestionnaires de cache définis dans les propriétés de configuration **Interact | cacheManagement | Gestionnaires de cache**, tels que `EHCache` ou `Ignite`.

### Valeur par défaut

`EHCache`

### Valeurs valides

N'importe quel gestionnaire de cache défini dans la propriété de configuration **Interact | cacheManagement | Gestionnaires de cache**.

## maxEntriesInCache

### Description

Nombre maximal d'objets de données de session à stocker dans ce cache. Lorsque le nombre maximal d'objets de données de session est atteint et que des données relatives à une session supplémentaire doivent être stockées, l'objet le moins récemment utilisé est supprimé.

### Valeur par défaut

100 000

### Valeurs valides

Un nombre entier supérieur à 0.

## timeoutInSecs

### Description

Durée, exprimée en secondes, qui s'est écoulée depuis l'utilisation ou la mise à jour d'un objet de données de session et qui est utilisée pour déterminer à quel moment l'objet doit être supprimé du cache.



**Remarque :** Si vous avez procédé à une mise à niveau à partir d'une version antérieure à 9.1, vous devrez reconfigurer la propriété `timeoutInSecs` car elle aura changé.

### Valeur par défaut

300

### Valeurs valides

Un nombre entier supérieur à 0.

## Interact | Caches | Cache Interact | Ignite

Un gestionnaire de cache "Ignite" est ajouté sous le nœud Gestionnaire de cache. Le cache Unica Interact Cache et le cache PatternStateCache peuvent utiliser EHCACHE ou Ignite indépendamment. Les paramètres suivants peuvent être configurés :

### cacheType

#### Description

Lorsque la valeur "Local" est sélectionnée, chaque nœud s'exécute indépendamment. Lorsque la valeur "Distribué" est sélectionnée, tous les nœuds forment une grille et les données sont distribuées sur la grille. Il s'agit de la valeur par défaut.

#### Valeur par défaut

Lorsque la valeur "Distribué" est sélectionnée, tous les nœuds forment une grille et les données sont distribuées sur la grille. Il s'agit de la valeur par défaut.

### discoveryIPAddresses

#### Description

La liste des adresses des nœuds séparée par des virgules sous le format `<IP>:<port>` pour que les nœuds communiquent les uns avec les autres. Si l'une de ces adresses est une adresse de multidiffusion, la reconnaissance

de multidiffusion est utilisée. Autrement, la reconnaissance d'IP statique est utilisée et, dans ce cas, au moins l'une d'elles doit être active à n'importe quel moment. C'est obligatoire lorsque la valeur "Distribué" est sélectionnée en tant que type de cache. La valeur par défaut 230.0.0.1:6363 est une multidiffusion.

### **Valeur par défaut**

230.0.0.1:6363

### **localPort**

Le port que chaque nœud utilisera pour communiquer avec les autres nœuds. Si ce port n'est pas spécifié, un port ouvert qui se trouve entre 47500 et 47509 sera utilisé. Il est suggéré de configurer ce paramètre si la reconnaissance d'IP statique est utilisée. Cette valeur peut être remplacée en utilisant la propriété JVM `-Dinteract.ignitePort=<valid port>`.

### **numberOfBackups**

Il s'agit des copies de sauvegarde de données enregistrées dans la grille. Une valeur supérieure entraînera un support de reprise et des performances de lecture améliorées avec le même coût que des performances d'écriture inférieures. Lorsque le type du cache est Distribué, définissez la valeur `numberOfBackups` sur 1.

### **overflowToDisk**

Détermine si les données sont conservées dans un fichier ou un disque temporaire.

**Remarque :** les requêtes portant sur la même session sur différentes instances peuvent échouer si une instance s'arrête et qu'aucune sauvegarde n'est configurée. Cela signifie que l'appel API échoue sur deux RT différents lorsque le type de cache est Ignite, dans un dossier.

## Interact | Caches | Cache Interact | Données de paramètre

Les propriétés de configuration de cette catégorie permettent de gérer le cache Interact qui est utilisé automatiquement par l'installation d'Unica Interact. Ces paramètres doivent être configurés individuellement pour chaque serveur d'exécution Unica Interact.

## asyncIntervalMillis

### Description

Délai d'attente, exprimé en millisecondes, observé par le gestionnaire de cache EHCACHE avant de répliquer des modifications sur d'autres instances d'exécution Unica Interact. Si la valeur n'est pas positive, ces modifications seront répliquées de façon synchrone.

Cette propriété de configuration n'est pas créée par défaut. Si vous créez cette propriété, elle est utilisée uniquement lorsque EHCACHE est le gestionnaire de cache et lorsque la propriété **cacheType** définie pour ce dernier a pour valeur `distributed`.

### Valeur par défaut

Aucun.

## (Parameter)

### Description

Modèle que vous pouvez utiliser pour créer un paramètre personnalisé à utiliser avec le cache Interact. Vous pouvez définir n'importe quel nom de paramètre et associer à celui-ci n'importe quelle valeur.

Pour créer un paramètre personnalisé, cliquez sur **(Parameter)** et indiquez le nom et la valeur à affecter au paramètre. Lorsque vous cliquez sur **Enregistrer les modifications**, le paramètre que vous avez créé est ajouté à la liste dans la catégorie Parameter Data.

### Valeur par défaut

Aucun

## Interact | cacheManagement | caches | PatternStateCache

La catégorie PatternStateCache permet d'héberger les états des modèles d'événement et les règles de suppression d'offre en temps réel. Par défaut, ce cache est configuré comme un cache à lecture immédiate et à écriture immédiate ; par conséquent, Unica Interact tente d'utiliser le premier modèle d'événement et les premières données de suppression

d'offre du cache. Si l'entrée demandée n'existe pas dans le cache, elle est chargée par l'implémentation du cache depuis la source de données, via la configuration JNDI ou directement via une connexion JDBC.

Pour utiliser une connexion JNDI, Unica Interact se connecte à un fournisseur de source de données existant qui a été défini via le serveur spécifié à l'aide du nom JNDI, de l'URL, etc. Pour une connexion JDBC, vous devez indiquer un ensemble de paramètres JDBC, notamment le nom de classe du pilote JDBC, l'URL de base de données et des informations d'authentification.

Notez que si vous définissez plusieurs sources JNDI et JDBC, la première source JNDI activée est utilisée, et si aucune source JNDI n'est activée, la première source JDBC activée est utilisée.

La catégorie `PatternStateCache` est nécessaire au fonctionnement correct d'Interact.

Cette catégorie peut également être configurée via une configuration externe `EHCache` pour les paramètres qui ne sont pas pris en charge dans **Interact | cacheManagement | Caches**.

Si vous utilisez `EHCache`, vous devez vérifier que `PatternStateCache` est correctement configuré.

## CacheManagerName

### Description

Nom du gestionnaire de cache qui gère le cache d'état de modèle Unica Interact. La valeur que vous entrez dans cette zone doit correspondre à l'un des gestionnaires de cache définis dans les propriétés de configuration **Interact | cacheManagement | Gestionnaires de cache**, tels que `EHCache` ou `Ignite`.

### Valeur par défaut

`EHCache`

### Valeurs valides

N'importe quel gestionnaire de cache défini dans la propriété de configuration **Interact | cacheManagement | Gestionnaires de cache**.



## maxEntriesInCache

### Description

Nombre maximal d'états de modèle d'événement à stocker dans ce cache. Lorsque le nombre maximal d'états de modèle d'événement est atteint et que des données relatives à un état de modèle d'événement supplémentaire doivent être stockées, l'objet le moins récemment utilisé est supprimé.

### Valeur par défaut

100 000

### Valeurs valides

Un nombre entier supérieur à 0.

## timeoutInSecs

### Description

Spécifie la durée, en secondes, qui s'écoule avant qu'un objet état de modèle d'événement expire dans le cache des états de modèle d'événement. Lorsqu'un objet état de ce type est inactif dans le cache depuis le nombre de secondes défini par `timeoutInSecs`, il peut être éliminé du cache en fonction de la règle du moins récemment utilisé. La valeur de cette propriété doit être supérieure à celle définie par la propriété `sessionTimeoutInSecs`.



**Remarque :** Si vous avez procédé à une mise à niveau à partir d'une version antérieure à 9.1, vous devrez reconfigurer la propriété `timeoutInSecs` car elle aura changé.

### Valeur par défaut

300

### Valeurs valides

Un nombre entier supérieur à 0.

## Interact | Caches | PatternStateCache | Données de paramètre

Les propriétés de configuration de cette catégorie permettent de gérer le cache des états de modèle utilisé pour héberger les états des modèles d'événement et les règles de suppression d'offre en temps réel.

### (Parameter)

#### Description

Modèle que vous pouvez utiliser pour créer un paramètre personnalisé à utiliser avec le cache des états de modèle. Vous pouvez définir n'importe quel nom de paramètre et associer à celui-ci n'importe quelle valeur.

Pour créer un paramètre personnalisé, cliquez sur **(Parameter)** et indiquez le nom et la valeur à affecter au paramètre. Lorsque vous cliquez sur **Enregistrer les modifications**, le paramètre que vous avez créé est ajouté à la liste dans la catégorie Parameter Data.

#### Valeur par défaut

Aucun

## Interact | cacheManagement | caches | PatternStateCache | loaderWriter

La catégorie **loaderWriter** contient la configuration du chargeur qui interagit avec des référentiels externes pour l'extraction et la persistance de modèles d'événement.

### className

#### Description

Nom de classe qualifié complet pour ce chargeur. Cette classe doit être conforme aux exigences du gestionnaire de cache choisi.

#### Valeur par défaut

```
com.unicacorp.interact.cache.ehcache.loaderwriter.
```

```
PatternStateEHCacheLoaderWriter
```

### **Valeurs valides**

Nom de classe qualifié complet.

## **classPath**

### **Description**

Chemin du fichier classe du chargeur. Si vous laissez cette valeur vide ou si l'entrée n'est pas valide, le chemin d'accès aux classes utilisé pour exécuter Unica Interact est utilisé.

### **Valeur par défaut**

Aucun

### **Valeurs valides**

Chemin d'accès aux classes valide.

## **writeMode**

### **Description**

Indique le mode utilisé par le programme d'écriture pour conserver les états de modèle d'événement nouveaux ou mis à jour dans le cache. Les options valides sont les suivantes :

- **WRITE\_THROUGH**. Chaque fois qu'une nouvelle entrée est ajoutée ou qu'une entrée existante est mise à jour, elle est immédiatement écrite dans les référentiels.
- **WRITE\_BEHIND**. Le gestionnaire de cache attend un certain temps avant de collecter un nombre de modifications donné, puis il les conserve dans les référentiels dans un lot.

### **Valeur par défaut**

WRITE\_THROUGH

### **Valeurs valides**

WRITE\_THROUGH ou WRITE\_BEHIND.

## batchSize

### Description

Nombre maximal d'objets d'état de modèle d'événement que le programme d'écriture va conserver dans un lot. Cette propriété est utilisée uniquement lorsque le paramètre **writeMode** a pour valeur `WRITE_BEHIND`.

### Valeur par défaut

100

### Valeurs valides

Valeur de type entier.

## maxDelayInSecs

### Description

Délai maximal, exprimé en secondes, observé par le gestionnaire de cache avant de conserver un objet d'état de modèle d'événement. Cette propriété est utilisée uniquement lorsque le paramètre **writeMode** a pour valeur `WRITE_BEHIND`.

### Valeur par défaut

5

### Valeurs valides

Valeur de type entier.

## Interact | Caches | PatternStateCache | loaderWriter | Données de paramètre

Les propriétés de configuration de cette catégorie permettent de gérer le chargeur de cache des états de modèle d'événement.

### (Parameter)

#### Description

Modèle que vous pouvez utiliser pour créer un paramètre personnalisé à utiliser avec le chargeur de cache des états de modèle d'événement. Vous pouvez définir n'importe quel nom de paramètre et associer à celui-ci n'importe quelle valeur.

Pour créer un paramètre personnalisé, cliquez sur **(Parameter)** et indiquez le nom et la valeur à affecter au paramètre. Lorsque vous cliquez sur **Enregistrer les modifications**, le paramètre que vous avez créé est ajouté à la liste dans la catégorie Parameter Data.

### Valeur par défaut

Aucun

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jndiSettings

La catégorie **jndiSettings** contient la configuration de la source de données JNDI qui sera utilisée par le chargeur pour communiquer avec la base de données de sauvegarde. Pour créer un nouvel ensemble de paramètres JNDI, développez la catégorie **jndiSettings** et cliquez sur la propriété **(jndiSetting)**.

### *(jndiSettings)*



**Remarque :** Lorsque WebSphere Application Server est utilisé, le programme d'écriture du chargeur n'est pas connecté à **jndiSettings**.

### Description

Lorsque vous cliquez sur cette catégorie, un formulaire apparaît. Pour définir une source de données JNDI, indiquez les valeurs suivantes :

- **Nom de la nouvelle catégorie** correspond au nom que vous souhaitez utiliser pour identifier cette connexion JNDI.
- Le paramètre **Activé** vous permet d'indiquer si vous souhaitez que cette connexion JNDI soit disponible ou non pour utilisation. Affectez la valeur `True` à ce paramètre pour les nouvelles connexions.
- **jniName** correspond au nom JNDI qui a été défini dans la source de données lors de sa configuration.
- **providerUrl** correspond à l'URL permettant de rechercher la source de données JNDI. Si vous laissez cette zone vide, l'URL de l'application Web hébergeant le serveur d'exécution Unica Interact est utilisée.
- **Fabrique de contexte initial** correspond au nom de classe qualifié complet de la classe de fabrique de contexte initial utilisée pour la connexion au fournisseur JNDI. Si l'application Web hébergeant le serveur d'exécution Unica Interact est utilisée pour **providerUrl**, laissez cette zone vide.

#### Valeur par défaut

Aucun.

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jdbcSettings

La catégorie **jdbcSettings** contient la configuration des connexions JDBC qui seront utilisées par le chargeur pour communiquer avec la base de données de sauvegarde. Pour créer un nouvel ensemble de paramètres JDBC, développez la catégorie **jdbcSettings** et cliquez sur la propriété (**jdbcSetting**).

#### *(jdbcSettings)*

##### Description

Lorsque vous cliquez sur cette catégorie, un formulaire apparaît. Pour définir une source de données JDBC, indiquez les valeurs suivantes :

- **Nom de la nouvelle catégorie** correspond au nom que vous souhaitez utiliser pour identifier cette connexion JDBC.
- Le paramètre **Activé** vous permet d'indiquer si vous souhaitez que cette connexion JDBC soit disponible ou non pour utilisation. Affectez la valeur `True` à ce paramètre pour les nouvelles connexions.
- **driverClassName** correspond au nom de classe qualifié complet du pilote JDBC. Cette classe doit exister dans le chemin d'accès aux classes configuré pour le démarrage du serveur cache d'hébergement.
- **databaseUrl** correspond à l'URL permettant de rechercher cette source de données JDBC.
- **asmUser** correspond au nom de l'utilisateur Unica qui a été configuré avec les données d'identification pour la connexion à la base de données dans cette connexion JDBC.
- **asmDataSource** correspond au nom de la source de données Unica qui a été configurée avec les données d'identification pour la connexion à la base de données dans cette connexion JDBC.
- **maxConnection** correspond au nombre maximal de connexions simultanées qui peuvent être établies avec la base de données dans cette connexion JDBC.

### Valeur par défaut

Aucun.

## Interact | triggeredMessage

Les propriétés de configuration de cette catégorie définissent les paramètres de tous les messages déclenchés et de la distribution des offres par les canaux.

### backendProcessIntervalMin

#### Description

Cette propriété définit la durée en minutes pendant laquelle l'unité d'exécution en arrière-plan charge et traite les distributions d'offre retardées. Cette valeur

doit être un nombre entier. Si elle est négative ou nulle, le processus d'arrière-plan est désactivé.

### **Valeurs valides**

Entier positif

## **autoLogContactAfterDelivery**

### **Description**

Si cette propriété est définie sur true, un événement de contact est automatiquement envoyé dès que cette offre est attribuée ou qu'elle est mise en file d'attente pour être distribuée ultérieurement. Si cette propriété est définie sur false, aucun événement de contact n'est envoyé automatiquement pour les offres sortantes. Il s'agit du comportement par défaut.

### **Valeurs valides**

True | False

## **waitForFlowchart**

### **Description**

Cette propriété détermine si le diagramme doit attendre la fin de la segmentation en cours d'exécution et indique le comportement à adopter une fois le délai d'expiration dépassé.

**DoNotWait** : le traitement d'un message déclenché démarre que la segmentation soit en cours d'exécution ou non. Toutefois, si des segments sont utilisés dans la règle d'éligibilité et/ou NextBestOffer est sélectionné comme méthode de sélection des offres, l'exécution des messages déclenchés reste en attente.

**OptionalWait** : le traitement d'un message déclenché attend la fin ou l'arrivée à expiration de la segmentation en cours d'exécution. Si le délai d'expiration est dépassé, un avertissement est consigné et le traitement de ce message déclenché se poursuit. Il s'agit de l'option par défaut.



**MandatoryWait** : le traitement d'un message déclenché attend la fin ou l'arrivée à expiration de la segmentation en cours d'exécution. Si le délai d'expiration est dépassé, une erreur est consignée et le traitement de ce message déclenché est abandonné.

### **Valeurs valides**

DoNotWait | OptionalWait | MandatoryWait

## **Interact | triggeredMessage | offerSelection**

Les propriétés de configuration de cette catégorie définissent les paramètres de la sélection d'offres dans les messages déclenchés.

### **maxCandidateOffers**

#### **Description**

Cette propriété définit le nombre maximal d'offres éligibles renvoyées par le moteur afin d'obtenir la meilleure offre pour la distribution. Il est possible qu'aucune de ces offres éligibles renvoyées ne puisse être envoyée à partir du canal sélectionné. Plus le nombre d'offres candidates est élevé, moins cela se produit. Toutefois, un nombre plus important d'offres candidates peut augmenter le temps de traitement.

#### **Valeurs valides**

Entier positif

### **defaultCellCode**

#### **Description**

Si l'offre distribuée est le résultat de l'évaluation d'une règle stratégique ou d'un enregistrement géré par une table, une cellule cible y est associée et les informations de cette cellule sont utilisées dans toutes les consignations associées. Toutefois, si une liste d'offres spécifiques est utilisée comme entrée pour la sélection d'offres, aucune cellule cible n'est disponible. Dans ce cas, la valeur de ce paramètre de configuration est utilisée. Vous devez

vous assurer que cette cellule cible et sa campagne sont incluses dans le déploiement. Le moyen le plus simple d'y parvenir consiste à ajouter la cellule dans une stratégie déployée.

## Interact | triggeredMessage | répartiteurs

Les propriétés de configuration de cette catégorie définissent les paramètres de tous les répartiteurs dans les messages déclenchés.

### **dispatchingThreads**

#### **Description**

Cette propriété définit le nombre d'unités d'exécution utilisées par le moteur pour appeler les répartiteurs de manière asynchrone. Si la valeur est négative ou nulle, l'appel des répartiteurs est synchrone. La valeur par défaut est 0.

#### **Valeurs valides**

Un entier

## Interact | triggeredMessage | répartiteurs | <nomRépartiteur>

Les propriétés de configuration de cette catégorie définissent les paramètres d'un répartiteur spécifique dans les messages déclenchés.

### **category name**

#### **Description**

Cette propriété définit le nom de ce répartiteur. Ce nom doit être unique parmi tous les répartiteurs.

### **type**

#### **Description**

Cette propriété définit le type de répartiteur.

#### **Valeurs valides**

InMemoryQueue | JMSQueue | Personnalisé | Kafka



**Remarque :** Pour WebSphere et WebLogic, il est recommandé d'utiliser la dernière version de groupe de correctifs fournie pour le JVM. Si vous avez utilisé Kafka dans la version précédente, vous pouvez alors définir le type comme étant Kafka dans la version mise à niveau.

JMSQueue ne prend en charge que WebLogic. Vous ne pouvez pas utiliser JMSQueue si vous utilisez WebSphere Application Server.

## className

### Description

Cette propriété définit le nom de classe complet de cette implémentation de répartiteur. Si le type est InMemoryQueue, la valeur doit être vide. Si le type est Custom, ce paramètre doit avoir la valeur `"com.unicacorp.interact.eventhandler.triggeredmessage.dispatchers.KafkaDispatcher"`. Si le type est Kafka, la valeur doit alors être vide.

## classPath

### Description

Cette propriété définit l'URL du fichier JAR qui inclut l'implémentation de ce répartiteur. Si le type est Kafka, la valeur doit alors être vide.

Interact | triggeredMessage | répartiteurs | <nomRépartiteur> |  
Données de paramètre

Les propriétés de configuration de cette catégorie définissent les paramètres d'un répartiteur spécifique dans les messages déclenchés.

Vous pouvez choisir entre trois types de répartiteurs. InMemoryQueue est le répartiteur interne d'Unica Interact. Custom est utilisé pour Kafka. JMSQueue est utilisé pour se connecter à un fournisseur JMS via JNDI. Kafka est distribué en tant que plateforme de diffusion, qui est utilisée pour publier et s'abonner aux flux d'enregistrements.

## category name

### La description

Cette propriété définit le nom de ce paramètre. Ce nom doit être unique parmi tous les paramètres de ce répartiteur.

## valeur

### La description

Cette propriété définit les paramètres, au format de paires nom/valeur, requis par ce répartiteur.



**Note:** Tous les paramètres sont sensibles à la casse et doivent être entrés comme indiqué ici.

Si le type est InMemoryQueue, le paramètre ci-après est pris en charge.

- queueCapacity: Facultatif. Nombre maximal d'offres en attente de répartition dans la file d'attente. Si cette propriété est spécifiée, il doit s'agir d'un entier positif. Si elle ne l'est pas ou qu'elle n'est pas valide, la valeur par défaut (1 000) est utilisée.

Si le type est Custom, les paramètres ci-après sont pris en charge.

- providerUrl : <hostname>:port (sensible à la casse)
- queueManager: nom du gestionnaire de files d'attente créé sur le serveur Kafka.
- messageQueueName: nom de la file d'attente de messages qui a été créée sur le serveur Kafka.
- enableConsumer: cette propriété doit être définie sur true.

- `asmUserforMQAuth`: nom d'utilisateur pour se connecter au serveur. Ce nom est requis si le serveur applique l'authentification. Sinon, il ne doit pas être spécifié.
- `authDS` : Le mot de passe associé au nom d'utilisateur pour se connecter au serveur. Ce nom est requis si le serveur applique l'authentification. Sinon, il ne doit pas être spécifié.

Si le type est `JMSQueue`, le paramètre ci-après est pris en charge.

- `providerUrl` : l'URL du fournisseur JNDI (sensible à la casse).
- `connectionFactoryJNDI` : nom JNDI de la fabrique de connexions JMS.
- `messageQueueJNDI` : nom JNDI de la file d'attente JMS à laquelle les messages déclenchés sont envoyés et extraits.
- `enableConsumer` : cette propriété spécifie si un consommateur de ces messages déclenchés doit être démarré dans Unica Interact. cette propriété doit être définie sur `true`. Si elle ne l'est pas, la valeur par défaut (`false`) est utilisée.
- `initialContextFactory` : nom complet de la classe de fabrique de contexte initial JNDI. Si vous utilisez `WebLogic`, la valeur de ce paramètre doit être `weblogic.jndi.WLInitialContextFactory`.

Si le type est `Kafka`, les paramètres ci-après sont pris en charge.

- `providerUrl` : une liste de paires hôte/port à utiliser pour établir la connexion initiale au cluster Kafka. Cette liste doit être sous la forme `host1:port1,host2:port2,...`.
- `rubrique` : une rubrique est une catégorie ou un nom de flux dans lequel les messages sont stockés et publiés. Tous les messages Kafka sont organisés en rubriques. Si vous devez envoyer un message, vous pouvez l'envoyer vers une rubrique spécifique et si vous souhaitez lire un message, vous pouvez le faire depuis une rubrique spécifique. Les applications producteurs écrivent des données à des rubriques et les applications clients lisent à partir des rubriques. Le nom de la

rubrique doit contenir un caractère alphanumérique ASCII, '.', '\_' et des caractères '-'. En raison des limitations au niveau des noms de rubrique, vous pouvez utiliser des rubriques avec un point ('.') ou un trait de soulignement ('\_'). La longueur maximale du nom d'une rubrique peut aller jusqu'à 255 caractères. Par exemple, si vous créez ou fournissez un nom de rubrique "InteractTM\_1", et si vous créez une rubrique telle que "InteractTM.1", l'erreur suivante est générée. "Le sujet InteractTM.1 entre en collision avec des sujets existants : InteractTM\_1."

- `group.id` : spécifie le nom du groupe de consommateurs auquel appartient un consommateur Kafka.
- `zookeeper.connect` : spécifie la chaîne de connexion zookeeper sous la forme `hostname:port`, où `hostname` et `port` sont l'hôte et le port d'un serveur ZooKeeper.
- `authentication` : les utilisateurs peuvent utiliser Kafka en activant différents mécanismes d'authentification.

### Paramètres obligatoires pour publier et s'abonner à des messages

Par défaut, le serveur Kafka ne prend en charge aucun mécanisme d'authentification. Les utilisateurs peuvent démarrer le serveur Kafka en supposant que le mécanisme d'authentification est désactivé. Dans ce cas, les utilisateurs peuvent définir le paramètre "authentication" avec la valeur "Aucune".

**Table 28. Paramètres obligatoires pour publier des messages**

Paramètres	Valeurs de paramètre autorisées/d'exemple
<code>providerUrl</code>	<hôte> : <port> (exemple : local-host : 9092)
<code>rubrique</code>	N'importe quelle chaîne (exemple : InteractTM)

**Table 28. Paramètres obligatoires pour publier des messages (continued)**

Paramètres	Valeurs de paramètre autorisées/d'exemple
authentication	none   Plain   SSL   SASL_SSL
zookeeper.connect	<hôte> : <port> (exemple : localhost : 2181)

**Table 29. Paramètres obligatoires pour s'abonner à des messages**

Paramètres	Valeur de paramètre autorisée/d'exemple
providerUrl	<hôte> : <port> (exemple : localhost : 9092)
group.id	N'importe quelle chaîne (exemple : InteractTMGateway)
rubrique	N'importe quelle chaîne (exemple : InteractTM)
authentication	none   Plain   SSL   SASL_SSL
zookeeper.connect	<hôte> : <port> (exemple : localhost : 2181)

### Mécanisme d'authentification

Vous pouvez utiliser Kafka en activant différents mécanismes d'authentification.

#### Authentification par mécanisme SASL\_PLAIN

Si vous avez besoin d'utiliser le mécanisme d'authentification SASL\_PLAIN, vous devez définir le paramètre "authentification" avec la valeur "Simple" en plus des paramètres pris en charge.

Les paramètres suivants sont requis si le mécanisme SASL\_PLAIN est pris en charge.

- `asmUserforMQAuth`: nom d'utilisateur pour se connecter au serveur. Ce nom est requis si le serveur applique l'authentification.
- `authDS` : Le mot de passe associé au nom d'utilisateur pour se connecter au serveur.
- nom d'utilisateur/mot de passe : nom d'utilisateur ou mot de passe du serveur Kafka configuré dans le fichier de configuration JASS.

Le tableau suivant présente les paramètres requis pour le mécanisme SASL\_PLAIN.

<b>Paramètres</b>	<b>Valeurs de paramètre autorisées/d'exemple</b>
<code>authentification</code>	Normal
<code>asmUserforMQAuth</code>	N'importe quelle chaîne (exemple : <code>test_user</code> )
<code>authDS</code>	N'importe quelle chaîne (exemple : <code>authDS</code> )
<code>nom utilisateur</code>	N'importe quelle chaîne (exemple : <code>test_user</code> )
<code>mot_de_passe</code>	N'importe quelle chaîne (exemple : <code>test-secret</code> )

Si le paramètre "authentification" est "Simple", vous devez utiliser des paramètres `asmUserforMQAuth/authDS` ou nom d'utilisateur/mot de passe pour l'authentification.



Créez les sources de données (authDS) dans la section Utilisateur de la configuration de plateforme. Reportez-vous à l'exemple suivant pour obtenir les détails des sources de données.

Source de données	Nom d'utilisateur	Mot de passe
authDS	test_user	test-secret

### Authentification par mécanisme SSL

Pour utiliser le mécanisme d'authentification SSL, vous devez définir le paramètre "authentification" avec la valeur "SSL" en plus des paramètres pris en charge.

Les paramètres suivants sont requis pour prendre en charge le mécanisme SSL.

- `ssl.keystore.location` : emplacement du fichier de magasin de clés. Vous pouvez l'utiliser pour une authentification bidirectionnelle pour le client.
- `ssl.truststore.location` : emplacement du fichier de magasin de confiance.
- `SSLKeystoreDS` : nom de la source de données du magasin de clés, qui stocke le mot de passe du magasin de clés SSL.
- `SSLKeyDS` : le nom de la source de données de la clé, qui stocke le mot de passe de la clé ssl.
- `SSLTruststoreDS` : le nom de la source de données du truststore, qui stocke le mot de passe du truststore ssl.

Le tableau suivant contient les paramètres pris en charge pour le mécanisme SSL.

Paramètres	Valeurs de paramètre autorisées/d'exemple
authentification	SSL

Paramètres	Valeurs de paramètre autorisées/d'exemple
ssl.keystore.location	Emplacement du magasin de clés SSL (exemple : <code>C:/SSL/kaf-ka.client.keystore.jks</code> )
ssl.truststore.location	Emplacement du magasin de clés SSL (exemple : <code>C:/SSL/kaf-ka.client.truststore.jks</code> )
asmUserforMQAuth	N'importe quelle chaîne (exemple : <code>test_user</code> )
SSLKeystoreDS	N'importe quelle chaîne (exemple : <code>SSLKeystoreDS</code> )
SSLKeyDS	N'importe quelle chaîne (exemple : <code>SSLKeyDS</code> )
SSLTruststoreDS	N'importe quelle chaîne (exemple : <code>SSLTruststoreDS</code> )

Créez les sources de données (SSLKeystoreDS, SSLKeyDS et SSLTruststoreDS) dans la section Utilisateur de la configuration de plateforme. Reportez-vous à l'exemple suivant pour obtenir les détails des sources de données.

Source de données	Nom d'utilisateur	Mot de passe
SSLKeystoreDS	fichier de clés	keystore-secret
SSLKeyDS	key	key-secret
SSLTruststoreDS	magasin de clés de confiance	truststore -secret



**Note:** Le magasin de clés client ou le magasin de confiance est requis du côté producteur ou consommateur dans l'application Unica Interact (où l'application Interact est installée). `C:/SSL/kafka.client.keystore.jks` et `C:/SSL/kafka.client.truststore.jks` sont les emplacements locaux où l'application Interact est installée.

## Authentification par Kerbrose

Kerbrose est utilisé comme méthode d'authentification dans le récepteur Kafka et la passerelle sortante Kafka.

Pour utiliser Kerbrose, les paramètres suivants avec leurs valeurs doivent être définis sur le récepteur de l'orchestrateur d'activité ou sur la passerelle sortante du message déclencheur, en plus des paramètres définis pour "Authentification par mécanisme SSL".

- `authentication = SASL_SSL`
- `sasl.mechanism = GSSAPI`

En outre, les paramètres JVM suivants doivent être ajoutés au serveur d'applications hébergeant l'environnement d'exécution Interact.

- `-Djava.security.auth.login.config=/path/to/jaas.conf`
- `-Djava.security.krb5.conf=/path/to/krb5.conf`

## Authentication by SASL\_SSL mechanism

If you require to use the SASL\_SSL authentication mechanism, then you must set the parameter "authentication" with value "SASL\_SSL" along with its supported parameters. SASL\_SSL mechanism is the combination of SASL\_PLAIN and SSL mechanisms. The following table includes the supported parameters for SASL\_SSL mechanism.

Paramètres	Valeurs de paramètre autorisées/d'exemple
authentication	SASL_SSL
asmUserforMQAuth	N'importe quelle chaîne (exemple : test_user)
authDS	N'importe quelle chaîne (exemple : authDS)
nom utilisateur	N'importe quelle chaîne (exemple : test_user)
mot_de_passe	N'importe quelle chaîne (exemple : test-secret)
ssl.keystore.location	Emplacement du magasin de clés SSL (exemple : C:/SSL/kaf-ka.client.keystore.jks )
ssl.truststore.location	Emplacement du magasin de clés SSL (exemple : C:/SSL/kaf-ka.client.truststore.jks )
SSLKeystoreDS	N'importe quelle chaîne (exemple : SSLKeystoreDS)
SSLKeyDS	N'importe quelle chaîne (exemple : SSLKeyDS)
SSLTruststoreDS	N'importe quelle chaîne (exemple : SSLTruststoreDS)

If the "authentication" parameter is "SASL\_SSL", you must either use asmUserforMQAuth/authDS or username/password.

Créez les sources de données (authDS, SSLKeystoreDS, SSLKeyDS et SSLTruststoreDS) dans la section Utilisateur de la configuration de plateforme.

Reportez-vous à l'exemple suivant pour obtenir les détails des sources de données.

Source de données	Nom d'utilisateur	Mot de passe
authDS	admin	admin-secret
SSLKeystoreDS	fichier de clés	test1234
SSLKeyDS	key	test1234
SSLTruststoreDS	magasin de clés de confiance	test1234



**Note:** Si vous fournissez des sources de données telles qu'authDS, SSLKeystoreDS, SSLKeyDS ou SSLTruststoreDS dans le paramètre de configuration de plateforme, vous devez également fournir le paramètre `asmUserforMQAuth`.

Le keystore/truststore client est requis du côté producteur/consommateur dans l'application Interact (où l'application Unica Interact est installée). `C:/SSL/kafka.client.keystore.jks` et `C:/SSL/kafka.client.truststore.jks` sont les emplacements locaux où l'application Interact est installée.

### Paramètres facultatifs pour publier des messages

La table suivante inclut les paramètres facultatifs requis pour publier des messages.

- `acks` : la configuration `acks` contrôle les critères selon lesquels les requêtes sont considérées comme terminées. Le paramètre "tous" entraîne le blocage de la validation complète de l'enregistrement.
- `retries` : si la demande échoue, le producteur peut réessayer. Etant donné que le nombre de nouvelles tentatives spécifié est défini sur 0, aucune

nouvelle tentative n'est autorisée. L'activation des nouvelles tentatives peut donner lieu à des doublons.

- `batch.size` : la taille de lot par défaut est en octets, lorsque plusieurs enregistrements sont regroupés et envoyés à une partition.
- `linger.ms` : le producteur attend le délai indiqué pour permettre l'envoi d'autres enregistrements afin que les enregistrements envoyés puissent être regroupés.
- `buffer.memory` : le nombre total d'octets de mémoire que le producteur peut utiliser pour mettre en mémoire tampon les enregistrements qui attendent d'être envoyés au serveur.

La table suivante inclut les paramètres facultatifs requis pour publier des messages.

Paramètres	Valeur par défaut	Valeurs de paramètre autorisées/d'exemple
<code>acks</code>	1	0, 1, all
<code>retries</code>	3	entier non négatif
<code>batch.size</code>	16384	Nombre entier positif
<code>linger.ms</code>	0	entier non négatif
<code>buffer.memory</code>	33554432	Nombre entier positif

### Paramètres facultatifs pour publier des messages

`enable.auto.commit` signifie que des décalages sont réalisés automatiquement avec une fréquence contrôlée par la configuration "`auto.commit.interval.ms`". La valeur d'`auto.commit.interval.ms` ne doit pas dépasser 1 000, étant donné que l'intervalle d'interrogation est défini sur 1 000. La valeur d'`auto.commit.interval.ms` ne doit pas dépasser la valeur de l'intervalle d'interrogation.

La table suivante inclut les paramètres facultatifs pour s'abonner à des messages.

Paramètres	Valeur par défaut	Valeurs de paramètre autorisées/d'exemple
enable.auto.commit	true	True, False
auto.commit.interval.ms	200	Nombre entier positif

### Paramètres de gestion de threads facultatifs

Les paramètres facultatifs suivants peuvent être utilisés pour la gestion de threads.

- `corePoolSize` : nombre de threads à conserver dans le pool pour surveiller le service Kafka.
- `maxPoolSize` : nombre maximal de threads à conserver dans le pool pour surveiller le service Kafka.
- `keepAliveTimeSecs` : durée maximale pendant laquelle les threads inactifs en excès attendent de nouvelles tâches avant de se terminer pour surveiller le service Kafka, lorsque le nombre de threads est supérieur au cœur.
- `queueCapacity` : taille de la file d'attente utilisée par le pool de threads pour surveiller le service Kafka.

Le tableau suivant contient les paramètres facultatifs pour la gestion de threads.

Paramètres	Valeur par défaut	Valeurs de paramètre autorisées/d'exemple
<code>corePoolSize</code>	1	Nombre entier positif
<code>maxPoolSize</code>	5	Nombre entier positif
<code>keepAliveTimeSecs</code>	5	Nombre entier positif
<code>queueCapacity</code>	100	Nombre entier positif

## Paramètres Zookeeper optionnels

Les paramètres facultatifs suivants peuvent être utilisés pour des activités Zookeeper.

`zookeeper.connection.timeout.ms` : Le temps maximum que le client attend pour établir une connexion avec zookeeper. S'ils ne sont pas définis, la valeur de "`zookeeper.session.timeout.ms`" est utilisée.

Le tableau suivant contient les paramètres facultatifs pour les activités Zookeeper.

Paramètres	Valeur par défaut	Valeur de paramètre autorisée/d'exemple
<code>zookeeper.connection.timeout.ms</code>	6000	Nombre entier positif

## Paramètres facultatifs pour créer une rubrique

Les paramètres facultatifs suivants peuvent être utilisés pour créer une rubrique.

- `num.partitions` : nombre de partitions pour le sujet de validation de décalage.
- `replication.factor` : le facteur de réplication pour modifier les rubriques de journal et les rubriques de répartition créées par l'application de traitement de flux.

Le tableau suivant contient les paramètres facultatifs pour la création de rubrique.



Paramètres	Valeur par défaut	Valeurs de paramètre autorisées/d'exemple
num.partitions	1	Nombre entier positif
replication.factor :	1	Nombre entier positif

## Interact | triggeredMessage | passerelles | <nomPasserelle>

Les propriétés de configuration de cette catégorie définissent les paramètres d'une passerelle spécifique dans les messages déclenchés.

Unica Interact ne prend pas en charge plusieurs instances d'une même passerelle. Tous les fichiers de configuration de passerelle doivent être accessibles à partir de chaque noeud d'exécution Unica Interact. Dans le cas d'une configuration distribuée, assurez-vous que les fichiers de passerelle sont conservés à un emplacement partagé.



**Remarque :** Les passerelles prêtes à l'emploi, dont le nom est "EMail", "MobilePush" et "UBX", sont disponibles sous ce nœud, avec tous les paramètres requis et leur valeur respective. Il n'est pas nécessaire de modifier des valeurs de la configuration Platform. Les seules modifications nécessaires sont celles des fichiers de propriétés référencés dans ces configurations. En cas de mise à niveau depuis une version précédente d'Interact, toute configuration existante utilisant ces passerelles continuera à fonctionner comme d'habitude.

### category name

#### Description

Cette propriété définit le nom de cette passerelle. Elle doit être unique parmi toutes les passerelles.

### className

#### Description

Cette propriété définit le nom de classe complet de cette implémentation de passerelle.

## classPath

### Description

Cette propriété définit l'identificateur URI du fichier JAR qui inclut l'implémentation de cette passerelle. Si elle reste vide, le chemin d'accès aux classes de l'application Interact hôte est utilisé.

Par exemple, dans un système Windows, si le fichier JAR de passerelle est disponible dans le répertoire `C:\HCL\Unica\EmailGateway\IBM_Interact_OMO_OutboundGateway_Silverpop_1.0\lib\OMO_OutboundGateway_Silverpop.jar`, le chemin d'accès aux classes doit être `file:///C:/HCL/Unica/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`. Par exemple, dans un système Windows, si le fichier JAR de passerelle est disponible dans le répertoire `/opt/HCL/Unica/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`, le chemin d'accès aux classes doit être `file:///opt/HCL/Unica/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`.

Interact | triggeredMessage | passerelles | <nomPasserelle> | Données de paramètre

Les propriétés de configuration de cette catégorie définissent les paramètres d'une passerelle spécifique dans les messages déclenchés.

## category name

### Description

Cette propriété définit le nom de ce paramètre. Ce nom doit être unique parmi tous les paramètres de cette passerelle.

## valeur

### Description

Cette propriété définit les paramètres, au format de paires nom/valeur, requis par cette passerelle. Pour toutes les passerelles, les paramètres ci-après sont pris en charge.



#### Remarque :

- Tous les paramètres sont sensibles à la casse et doivent être entrés comme indiqué ici.
  - `validationTimeoutMillis` : délai en millisecondes après lequel la validation d'une offre par l'intermédiaire de cette passerelle arrive à expiration. La valeur par défaut est 500.
  - `deliveryTimeoutMillis` : délai en millisecondes après lequel la distribution d'une offre à l'aide de cette passerelle arrive à expiration. La valeur par défaut est 1000.
- Afin d'intégrer les journaux associés à la passerelle dans le fichier `Interact.log`, placez l'ancien fichier "`interact_log4j2.xml`" des versions précédentes à 11.1 dans "`InteractRT.war/WEB-INF/classes`", ainsi que dans un emplacement situé n'importe où en dehors du fichier war. Vous devez spécifier le paramètre JVM suivant dans le serveur d'applications : `-Dlog4j.configuration=file:/opt/any_location/interact_log4j.properties`

## Interact | triggeredMessage | canaux

Les propriétés de configuration de cette catégorie définissent les paramètres de tous les canaux dans les messages déclenchés.

## type

### Description

Cette propriété définit le nœud racine des paramètres liés à une passerelle spécifique. L'option Default utilise le sélecteur de canaux intégré, qui est basé sur la liste des canaux définis dans l'interface utilisateur des messages déclenchés. Si l'option Default est sélectionnée, les valeurs className et classPath doivent rester vides. L'option Custom utilise l'implémentation client d'IChannelSelector.

### Valeurs valides

Défaut | Personnalisé

## className

### Description

Cette propriété définit le nom de classe complet de l'implémentation client du sélecteur de canaux. Ce paramètre est obligatoire si le type est Custom.

## classPath

### Description

Cette propriété définit l'URL du fichier JAR qui inclut l'implémentation de l'implémentation client du sélecteur de canaux. Si elle reste vide, le chemin d'accès aux classes de l'application Interact hôte est utilisé.

## Interact | triggeredMessage | canaux | Données de paramètre

Les propriétés de configuration de cette catégorie définissent les paramètres d'un canal spécifique dans les messages déclenchés.

## category name

### Description

Cette propriété définit le nom de ce paramètre. Ce nom doit être unique parmi tous les paramètres de ce canal.

## valeur

### Description

Cette propriété définit les paramètres, au format de paires nom/valeur, requis par le sélecteur de canaux.

Si vous utilisez **Canal préféré des clients** pour votre canal, vous devez créer

Interact | triggeredMessage | canaux | <nomCanal>

Les propriétés de configuration de cette catégorie définissent les paramètres d'un canal spécifique dans les messages déclenchés.

## category name

### Description

Cette propriété définit le nom du canal par l'intermédiaire duquel les offres sont envoyées. Elle doit correspondre à celles définies dans la phase de conception sous **Campaign | partitions | <partition[N]> | Interact | outboundChannels**.

Interact | triggeredMessage | canaux | <nomCanal> | <nomGestionnaire>

Les propriétés de configuration de cette catégorie définissent les paramètres d'un gestionnaire spécifique dans les messages déclenchés utilisé pour envoyer des offres.

## category name

### Description

Cette propriété définit le nom du gestionnaire que le canal utilisera pour envoyer des offres.

## dispatcher

### Description

Cette propriété définit le nom du répartiteur par l'intermédiaire duquel ce gestionnaire envoie des offres à la passerelle. Il doit s'agir de l'un de ceux définis sous **interact | triggeredMessage | répartiteurs**.

## **passerelle**

### **Description**

Cette propriété définit le nom de la passerelle à laquelle ce gestionnaire envoie des offres. Il doit s'agir de l'une de celles définies sous **interact | triggeredMessage | passerelles**.

## **mode**

### **Description**

Cette propriété définit le mode d'utilisation de ce gestionnaire. Si Basculement est sélectionné, ce gestionnaire n'est utilisé que si aucun des gestionnaires de priorité supérieure définis dans ce canal n'est parvenu à envoyer des offres. Si Module complémentaire est sélectionné, ce gestionnaire est utilisé que d'autres gestionnaires aient réussi ou non à envoyer des offres.

## **priorité**

### **Description**

Cette propriété définit la priorité de ce gestionnaire. Le moteur commence par essayer d'utiliser le gestionnaire de priorité la plus élevée pour envoyer des offres.

### **Valeurs valides**

Tout entier

### **Par défaut**

100

## Interact | activityOrchestrator

La catégorie Activity Orchestrator spécifie les récepteurs et les passerelles pour votre activité de passerelle entrante Unica Interact.

Utilisez les propriétés de configuration **Interact | activityOrchestrator | récepteurs** pour configurer vos récepteurs Unica Interact. Utilisez les propriétés de configuration **Interact | activityOrchestrator | passerelles** pour configurer les passerelles à utiliser dans Unica Interact.

### Interact | activityOrchestrator | récepteurs

La catégorie Activity Orchestrator Receivers spécifie les récepteurs d'événements pour votre activité de passerelle entrante Unica Interact.

#### Nom de catégorie

##### Description

Nom de votre récepteur.

#### Type

##### Description

Type de récepteur. Vous pouvez choisir entre Kafka et Custom. Custom nécessite que vous utilisiez une implémentation de `iReceiver`.



**Remarque :** Si vous avez utilisé Kafka dans la version précédente, vous pouvez alors définir le type comme étant Kafka dans la version mise à niveau.

#### Activé

##### Description

Sélectionnez `True` pour activer le récepteur ou `false` pour le désactiver.

## className

### Description

Cette propriété définit le nom de classe complet de cette implémentation de récepteur. Elle n'est utilisée que lorsque le type est `Custom`. Si le type est `Kafka`, la valeur doit alors être vide.

## classPath

### Description

Cette propriété définit l'identificateur URI du fichier JAR qui inclut l'implémentation de ce récepteur. Si elle reste vide, le chemin d'accès aux classes de l'application Unica Interact hôte est utilisé. Elle n'est utilisée que lorsque le type est `Custom`. Si le type est `Kafka`, la valeur doit alors être vide.

### Interact | activityOrchestrator | récepteur | Données de paramètre

Vous pouvez ajouter des paramètres de récepteur tels que `queueManager` et `messageQueueName` pour définir votre file d'attente de récepteur.

Si le type est `Kafka`, les paramètres ci-après sont pris en charge.

- `providerUrl` : une liste de paires hôte/port à utiliser pour établir la connexion initiale avec le cluster Kafka. Cette liste doit avoir la forme de `host1:port1,host2:port2,...`
- `topic`: Une rubrique est un nom de catégorie ou de flux dans laquelle les messages sont stockés et publiés. Tous les messages Kafka sont organisés en rubriques. Si vous devez envoyer un message, vous pouvez l'envoyer vers une rubrique spécifique et si vous souhaitez lire un message, vous pouvez le faire depuis une rubrique spécifique. Les applications producteurs écrivent des données à des rubriques et les applications clients lisent à partir des rubriques. Le nom de la rubrique doit contenir un caractère alphanumérique ASCII, '.', '\_' et des caractères '-'. En raison des limitations au niveau des noms de rubrique, vous pouvez utiliser des rubriques avec un point ('.') ou un trait de soulignement ('\_'). La longueur maximale du nom d'une rubrique peut



aller jusqu'à 255 caractères. Par exemple, si vous créez ou fournissez un nom de rubrique "InteractTM\_1", et si vous créez une rubrique telle que "InteractTM.1", l'erreur suivante est générée. "Topic InteractTM.1 collides with existing topics: InteractTM \_1."

- **group.id** : Indique le nom du groupe de consommateurs auquel appartient un consommateur Kafka.
- **zookeeper.connect** : Spécifie la chaîne de connexion ZooKeeper sous forme de `hostname:port`, où le nom d'hôte et le port sont l'hôte et le port d'un serveur Zookeeper.
- **Authentication** : Les utilisateurs peuvent utiliser Kafka en activant différents mécanismes d'authentification.

### Paramètres obligatoires pour s'abonner à des messages

Par défaut, le serveur Kafka ne prend en charge aucun mécanisme d'authentification. Vous pouvez démarrer le serveur Kafka en supposant que le mécanisme d'authentification est désactivé. Dans ce cas, vous pouvez définir le paramètre "authentication" avec la valeur "Aucune". La table suivante inclut les paramètres obligatoires requis pour s'abonner à des messages.

Paramètres	Valeur de paramètre autorisée/d'exemple
providerUrl	<host>:<port> (exemple : localhost:9092)
group.id	N'importe quelle chaîne (exemple : InteractTMGateway)
topic	N'importe quelle chaîne (exemple : InteractTM)
authentication	Toute chaîne de caractères
zookeeper.connect	<host>:<port> (exemple : localhost:2181)

## Mécanisme d'authentification

Vous pouvez utiliser Kafka en activant différents mécanismes d'authentification.

### Authentification par mécanisme SASL\_PLAIN

Si vous avez besoin d'utiliser le mécanisme d'authentification SASL\_PLAIN, vous devez définir le paramètre "authentification" avec la valeur "Simple" en plus des paramètres pris en charge.

Les paramètres suivants sont requis si le mécanisme SASL\_PLAIN est pris en charge.

- `asmUserforMQAuth` : nom d'utilisateur permettant de se connecter au serveur. Ce nom est requis si le serveur applique l'authentification.
- `authDS` : mot de passe associé au nom d'utilisateur permettant de se connecter au serveur.
- Nom d'utilisateur/Mot de passe Le nom d'utilisateur ou le mot de passe du serveur Kafka configuré dans le fichier de configuration JASS.

Le tableau suivant présente les paramètres requis pour le mécanisme SASL\_PLAIN.

Paramètres	Valeurs de paramètre autorisées/d'exemple
<code>authentification</code>	Normal
<code>asmUserforMQAuth</code>	N'importe quelle chaîne (exemple : <code>test_user</code> )
<code>authDS</code>	N'importe quelle chaîne (exemple : <code>authDS</code> )
<code>nom utilisateur</code>	N'importe quelle chaîne (exemple : <code>test_user</code> )

Paramètres	Valeurs de paramètre autorisées/d'exemple
mot de passe	N'importe quelle chaîne (exemple : test-secret)

Si le paramètre "authentification" est "Simple", vous devez utiliser des paramètres asmUserforMQAuth/authDS ou nom d'utilisateur/mot de passe pour l'authentification.

Créez les sources de données (authDS) dans la section Utilisateur de la configuration de plateforme. Reportez-vous à l'exemple suivant pour obtenir les détails des sources de données.

Source de données	Nom d'utilisateur	Mot de passe
authDS	test_user	test-secret

### Authentification par mécanisme SSL

Pour utiliser le mécanisme d'authentification SSL, vous devez définir le paramètre "authentification" avec la valeur "SSL" en plus des paramètres pris en charge.

Les paramètres suivants sont requis pour prendre en charge le mécanisme SSL.

- `ssl.keystore.location` : L'emplacement du fichier du magasin de clés. Vous pouvez l'utiliser pour une authentification bidirectionnelle pour le client.
- `ssl.truststore.location` : L'emplacement du fichier du magasin de certification.
- `SSLKeystoreDS` : Le nom de la source de données du magasin de clés qui stocke le mot de passe du magasin de clés SSL.

- **SSLKeyDS** : Le nom de la source de données de clés qui stocke le mot de passe de la clé SSL.
- **SSLTruststoreDS** : Le nom de la source de données du magasin de certification qui stocke le mot de passe du magasin de certification SSL.

Le tableau suivant contient les paramètres pris en charge pour le mécanisme SSL.

Paramètres	Valeurs de paramètre autorisées/d'exemple
authentification	SSL
ssl.keystore.location	SSL Keystore location (exemple: <code>C:/SSL/kafka.client.keystore.jks</code> )
ssl.truststore.location	SSL Keystore location (exemple: <code>C:/SSL/kafka.client.truststore.jks</code> )
asmUserforMQAuth	N'importe quelle chaîne (exemple : <code>test_user</code> )
SSLKeystoreDS	N'importe quelle chaîne (exemple : <code>SSLKeystoreDS</code> )
SSLKeyDS	N'importe quelle chaîne (exemple : <code>SSLKeyDS</code> )
SSLTruststoreDS	N'importe quelle chaîne (exemple : <code>SSLTruststoreDS</code> )

Créez les sources de données (SSLKeystoreDS, SSLKeyDS et SSLTruststoreDS) dans la section Utilisateur de la configuration de plateforme. Reportez-vous à l'exemple suivant pour obtenir les détails des sources de données.

Source de données	Nom d'utilisateur	Mot de passe
SSLKeystoreDS	fichier de clés	keystore-secret
SSLKeyDS	key	key-secret
SSLTruststoreDS	magasin de clés de confiance	truststore -secret



**Remarque :** Le magasin de clés ou le magasin de certification client est requis côté Producteur ou Consommateur dans l'application Interact (où l'application Interact est installée).

`C:/SSL/kafka.client.keystore.jks` et `C:/SSL/kafka.client.truststore.jks` sont les emplacements locaux, où l'application Interact est installée.

### Authentication by SASL\_SSL mechanism

If you require to use the SASL\_SSL authentication mechanism, then you must set the parameter "authentication" with value "SASL\_SSL" along with its supported parameters. SASL\_SSL mechanism is the combination of SASL\_PLAIN and SSL mechanisms. The following table includes the supported parameters for SASL\_SSL mechanism.

Paramètres	Valeurs de paramètre autorisées/d'exemple
authentication	SASL_SSL
asmUserforMQAuth	N'importe quelle chaîne (exemple : test_user)
authDS	N'importe quelle chaîne (exemple : authDS)
nom utilisateur	N'importe quelle chaîne (exemple : test_user)

Paramètres	Valeurs de paramètre autorisées/d'exemple
mot de passe	N'importe quelle chaîne (exemple : test-secret)
ssl.keystore.location	SSL Keystore location (exemple: <code>C:/SSL/kafka.client.keystore.jks</code> )
ssl.truststore.location	SSL Keystore location (exemple: <code>C:/SSL/kafka.client.truststore.jks</code> )
SSLKeystoreDS	N'importe quelle chaîne (exemple : SSLKeystoreDS)
SSLKeyDS	N'importe quelle chaîne (exemple : SSLKeyDS)
SSLTruststoreDS	N'importe quelle chaîne (exemple : SSLTruststoreDS)

If the "authentication" parameter is "SASL\_SSL", you must either use `asmUserforMQAuth/authDS` or `username/password`.

Créez les sources de données (`authDS`, `SSLKeystoreDS`, `SSLKeyDS` et `SSLTruststoreDS`) dans la section Utilisateur de la configuration de plateforme. Reportez-vous à l'exemple suivant pour obtenir les détails des sources de données.

Source de données	Nom d'utilisateur	Mot de passe
<code>authDS</code>	admin	admin-secret
<code>SSLKeystoreDS</code>	fichier de clés	test1234
<code>SSLKeyDS</code>	key	test1234

Source de données	Nom d'utilisateur	Mot de passe
SSLTruststoreDS	magasin de clés de confiance	test1234



**Remarque :** Si vous fournissez des sources de données telles qu'authDS, SSLKeystoreDS, SSLKeyDS ou SSLTruststoreDS dans le paramètre de configuration de plateforme, vous devez également fournir le paramètre asmUserforMQAuth.

Le magasin de clés/magasin de certification client est requis côté Producteur/Consommateur dans l'application Interact (où l'application Interact est installée). `C:/SSL/kafka.client.keystore.jks` et `C:/SSL/kafka.client.truststore.jks` sont les emplacements locaux, où l'application Interact est installée.

### Paramètres facultatifs pour publier des messages

enable.auto.commit signifie que des décalages sont réalisés automatiquement avec une fréquence contrôlée par la configuration "auto.commit.interval.ms". La valeur d'auto.commit.interval.ms ne doit pas dépasser 1 000, étant donné que l'intervalle d'interrogation est défini sur 1 000. La valeur d'auto.commit.interval.ms ne doit pas dépasser la valeur de l'intervalle d'interrogation.

La table suivante inclut les paramètres facultatifs pour s'abonner à des messages.

Paramètres	Valeur par défaut	Valeurs de paramètre autorisées/d'exemple
enable.auto.commit	true	True, False
auto.commit.interval- .ms	200	Nombre entier positif

## Paramètres de gestion de threads facultatifs

Les paramètres facultatifs suivants peuvent être utilisés pour la gestion de threads.

- `corePoolSize` : Le nombre de threads à conserver dans le pool pour surveiller le service Kafka.
- `maxPoolSize` : Le nombre maximum de threads à conserver dans le pool pour surveiller le service Kafka.
- `keepAliveTimeSecs` : Le laps de temps maximum pendant lequel les unités d'exécution inactives en trop attendent que les nouvelles tâches prennent fin en vue de surveiller le service Kafka, lorsque le nombre de threads est supérieur au `core`.
- `queueCapacity` : La taille de la file d'attente utilisée par le pool de threads pour surveiller le service Kafka.

Le tableau suivant contient les paramètres facultatifs pour la gestion de threads.

Paramètres	Valeur par défaut	Valeurs de paramètre autorisées/d'exemple
<code>corePoolSize</code>	1	Nombre entier positif
<code>maxPoolSize</code>	5	Nombre entier positif
<code>keepAliveTimeSecs</code>	5	Nombre entier positif
<code>queueCapacity</code>	100	Nombre entier positif

## Paramètres Zookeeper optionnels

Les paramètres facultatifs suivants peuvent être utilisés pour des activités Zookeeper.

`zookeeper.connection.timeout.ms` : Le laps de temps maximum pendant lequel le client attend pour établir une connexion avec Zookeeper. S'ils ne sont pas définis, la valeur de "`zookeeper.session.timeout.ms`" est utilisée.



Le tableau suivant contient les paramètres facultatifs pour les activités Zookeeper.

Paramètre	Valeur par défaut	Valeur de paramètre autorisée/d'exemple
zookeeper.connection.timeout.ms	6000	Nombre entier positif

### Paramètres facultatifs pour créer une rubrique

Les paramètres facultatifs suivants peuvent être utilisés pour créer une rubrique.

- num.partitions : Le nombre de partitions pour la rubrique de validation de décalage.
- replication.factor : Le facteur de réplication pour modifier les rubriques de journal et les rubriques de répartition créé par l'application de traitement des flots.

Le tableau suivant contient les paramètres facultatifs pour la création de rubrique.

Paramètres	Valeur par défaut	Valeurs de paramètre autorisées/d'exemple
num.partitions	1	Nombre entier positif
replication.factor :	1	Nombre entier positif

## Interact | activityOrchestrator | passerelles

La catégorie Activity Orchestrator Gateway spécifie les passerelles pour votre activité de passerelle entrante Unica Interact.



**Remarque :** La passerelle entrante prête à l'emploi, dont le nom est "UBX", est disponible sous ce nœud, avec tous les paramètres requis et leur valeur respective. Il n'est pas nécessaire de modifier des valeurs de la configuration Platform. Les seules modifications nécessaires sont celles des fichiers de propriétés référencés dans ces configurations. En cas de mise à niveau depuis une version précédente d'Interact, toute configuration existante utilisant ces passerelles continue à fonctionner comme d'habitude.

## Nom de catégorie

### Description

Nom de votre passerelle.

## className

### Description

Cette propriété définit le nom de classe complet de cette implémentation de passerelle.

## classPath

### Description

Cette propriété définit l'identificateur URI du fichier JAR qui inclut l'implémentation de cette passerelle. Si elle reste vide, le chemin d'accès aux classes de l'application Unica Interact hôte est utilisé. Elle n'est utilisée que lorsque le type est `Custom`.

## Interact | activityOrchestrator | passerelles | Données de paramètre

Vous pouvez ajouter des paramètres de passerelle pour vos fichiers de configuration de passerelle tels que `OMO-conf_inbound_UBX_interactEventNameMapping` et `OMO-conf_inbound_UBX_interactEventPayloadMapping`.

## Chemin d'accès à l'arborescence de configuration

```
Interact | activityOrchestrator | gateways | <gatewayName> | Parameter  
Data | <partitionName> | processTimeoutMillis | value=XXX
```

```
Interact | activityOrchestrator | gateways | <gatewayName> | Parameter  
Data | <partitionName> | OMO-processTimeoutMillis | value=XXX
```

## Interact | ETL | patternStateETL

Les propriétés de configuration de cette catégorie définissent les paramètres du processus ETL.

### Nouveau nom de la catégorie

#### Description

Indiquez un nom qui identifie de manière unique la configuration. Notez que vous devez fournir ce nom exact lorsque vous exécutez le processus ETL autonome. Pour faciliter la saisie de ce nom sur la ligne de commande, évitez d'indiquer de espaces ou des signes de ponctuation. Indiquez un nom de type `ETLProfile1`.

### runOnceADay

#### Description

Indique si le processus ETL autonome de cette configuration doit s'exécuter une fois chaque jour. Les réponses valides sont **Oui** ou **Non**. Si vous indiquez **Non**, `processSleepIntervallnMinutes` détermine le planning d'exécution du processus.

### preferredStartTime

#### Description

Heure préférée de début du processus ETL autonome. Indiquez l'heure au format HH:MM:SS AM/PM, par exemple `01:00:00 AM`.

### preferredEndTime

#### Description

Heure préférée de fin du processus ETL autonome. Indiquez l'heure au format HH:MM:SS AM/PM, par exemple 08:00:00 AM.

## **processSleepIntervallInMinutes**

### **Description**

Si vous n'avez pas configuré le processus ETL autonome pour qu'il s'exécute une fois par jour (comme indiqué dans la propriété **runOnceADay**), cette propriété indique la fréquence des exécutions du processus ETL. Par exemple, si vous indiquez 15, le processus ETL autonome attend 15 après la fin de son exécution avant de se relancer.

## **maxJDBCInsertBatchSize**

### **Description**

Nombre maximum d'enregistrements d'un lot JDBC avant de soumettre la requête. La valeur par défaut est égale à 5 000. Notez qu'il ne s'agit pas du nombre maximal d'enregistrements traités par le processus ETL en une itération. Lors de chaque itération, le processus ETL traite tous les enregistrements disponibles provenant de la table UACI\_EVENTPATTERNSTATE. Cependant, tous ces enregistrements sont fragmentés en blocs **maxJDBCInsertSize**.

## **maxJDBCFetchBatchSize**

### **Description**

Nombre maximum d'enregistrements d'un lot JDBC à extraire dans la base de données intermédiaire.

Il se peut que vous deviez augmenter cette valeur afin de régler les performances du processus ETL.

## **communicationPort**

### **Description**

Port réseau sur lequel le processus ETL autonome écoute les demandes d'arrêt. Dans des circonstances normales, il n'y a aucune raison de modifier la valeur par défaut.

## **queueLength**

### **Description**

Valeur employée pour le réglage des performances. Les collections des données d'état de modèle sont extraites et transformées en objets qui sont ajoutés à une file d'attente en vue de leur traitement et de leur inscription dans la base de données. Cette propriété contrôle la taille de la file d'attente.

## **completionNotificationScript**

### **Description**

Indique le chemin d'accès absolu d'un script à exécuter lorsque le processus ETL est achevé. Si vous spécifiez un script, trois arguments sont transmis au script de notification d'achèvement : heure de début, heure de fin et nombre total d'enregistrements de modèle d'événement traités. L'heure de début et l'heure de fin sont des valeurs numériques représentant le nombre de millisecondes écoulées depuis 1970.

## **Interact | ETL | patternStateETL | <NomETLEtatDeModèle> | RuntimeDS**

Les propriétés de configuration de cette catégorie définissent les paramètres de la source de données d'exécution d'ETL.

## **type**

### **Description**

Liste des types de base de données pris en charge pour la source de données que vous définissez.

## dsname

### Description

Nom JNDI de la source de données. Ce nom doit également être utilisé dans la configuration de la source de données de l'utilisateur, ce qui permet de garantir que ce dernier a accès aux sources de données cible et d'exécution.

## driver

### Description

Nom du pilote JDBC, par exemple :

Oracle : `oracle.jdbc.OracleDriver`

Microsoft SQL Server : `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2 : `com.ibm.db2.jcc.DB2Driver`

MariaDB : `org.mariadb.jdbc.Driver`

## serverURL

### Description

URL de la source de données, par exemple :

Oracle : `jdbc:oracle:thin:@`

`<hôte_base_de_données>:<port_base_de_données>:>nom_service_base_de_données>`

Microsoft SQL Server : `jdbc:sqlserver://`

`<hôte_base_de_données>:<port_base_de_données> ;databaseName=`

`<nom_base_de_données>`

IBM DB2 : `jdbc:db2://`

`<hôte_base_de_données>:<port_base_de_données>/`

`<nom_base_de_données>`

MariaDB : `jdbc:mariadb://`

`<hôte_base_de_données>:<port_base_de_données>/`

`<nom_base_de_données>`

## connectionpoolSize

### Description

Valeur indiquant la taille du pool de connexions, pour le réglage des performances. Les données d'état de modèle sont lues et transformées simultanément en fonction des connexions de base de données disponibles. L'augmentation de la taille du pool de connexions permet d'augmenter le nombre de connexions simultanées à la base de données, tout en respectant les limitations de mémoire et des fonctionnalités de lecture/écriture de la base de données. Par exemple, si cette valeur est égale à 4, quatre travaux s'exécutent simultanément. Si vous disposez d'un grand volume de données, vous pouvez augmenter cette valeur et indiquer 10 ou 20, tant que vous disposez d'une mémoire suffisante et que les performances de la base de données sont acceptables.

## schéma

### Description

Nom du schéma de la base de données à laquelle cette configuration se connecte.

## connectionRetryPeriod

### Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Unica Interact retente automatiquement la demande de connexion à la base de données en cas d'échec. Unica 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 cette propriété prend la valeur 0, Unica Interact effectue de nouvelles tentatives de manière illimitée. Si cette propriété prend la valeur -1, aucune nouvelle tentative de connexion n'est effectuée.

## connectionRetryDelay

### Description

La propriété `connectionRetryDelay` indique la durée en secondes pendant laquelle Unica Interact attend avant de tenter de se reconnecter à la base de données après un échec. Si cette propriété prend la valeur -1, aucune nouvelle tentative n'est effectuée.

## Interact | ETL | patternStateETL | <NomETLEtatDeModèle> | TargetDS

Les propriétés de configuration de cette catégorie définissent les paramètres de la source de données cible d'ETL.

### type

#### Description

Liste des types de base de données pris en charge pour la source de données que vous définissez.

### dsname

#### Description

Nom JNDI de la source de données. Ce nom doit également être utilisé dans la configuration de la source de données de l'utilisateur, ce qui permet de garantir que ce dernier a accès aux sources de données cible et d'exécution.

### driver

#### Description

Nom du pilote JDBC, par exemple :

Oracle : `oracle.jdbc.OracleDriver`

Microsoft SQL Server : `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2 : `com.ibm.db2.jcc.DB2Driver`

MariaDB : `org.mariadb.jdbc.Driver`



## serverURL

### Description

URL de la source de données, par exemple :

Oracle : jdbc:oracle:thin:@

<hôte\_base\_de\_données>:<port\_base\_de\_données>:>nom\_service\_base\_de\_données>

Microsoft SQL Server : jdbc:sqlserver://

<hôte\_base\_de\_données>:<port\_base\_de\_données> ;databaseName=

<nom\_base\_de\_données>

IBM DB2 : jdbc:db2://

<hôte\_base\_de\_données>:<port\_base\_de\_données>/

<nom\_base\_de\_données>

MariaDB : jdbc:mariadb://

<hôte\_base\_de\_données>:<port\_base\_de\_données>/

<nom\_base\_de\_données>

## connectionpoolSize

### Description

Valeur indiquant la taille du pool de connexions, pour le réglage des performances. Les données d'état de modèle sont lues et transformées simultanément en fonction des connexions de base de données disponibles. L'augmentation de la taille du pool de connexions permet d'augmenter le nombre de connexions simultanées à la base de données, tout en respectant les limitations de mémoire et des fonctionnalités de lecture/écriture de la base de données. Par exemple, si cette valeur est égale à 4, quatre travaux s'exécutent simultanément. Si vous disposez d'un grand volume de données, vous pouvez augmenter cette valeur et indiquer 10 ou 20, tant que vous disposez d'une mémoire suffisante et que les performances de la base de données sont acceptables.

## schéma

### Description

Nom du schéma de la base de données à laquelle cette configuration se connecte.

## connectionRetryPeriod

### Description

La propriété `ConnectionRetryPeriod` spécifie la durée, en secondes, pendant laquelle Unica Interact retente automatiquement la demande de connexion à la base de données en cas d'échec. Unica 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 cette propriété prend la valeur 0, Unica Interact effectue de nouvelles tentatives de manière illimitée. Si cette propriété prend la valeur -1, aucune nouvelle tentative de connexion n'est effectuée.

## connectionRetryDelay

### Description

La propriété `ConnectionRetryDelay` indique la durée en secondes pendant laquelle Unica Interact attend avant de tenter de se reconnecter à la base de données après un échec. Si cette propriété prend la valeur -1, aucune nouvelle tentative n'est effectuée.

## Interact | ETL | patternStateETL | <NomETLEtatDeModèle> | Rapport

Les propriétés de configuration de cette catégorie définissent les paramètres du processus d'agrégation des rapports ETL.

## activer

### Description

Active ou désactive l'intégration des rapports avec ETL. Par défaut, cette propriété est désactivée.

Si sa valeur est `disable`, cette propriété désactive les mises à jour sur la table `UARI_DELTA_PATTERNS`. Elle ne désactive pas intégralement la production de rapports.



**Remarque :** Pour désactiver l'intégration des rapports à ETL, vous devez également modifier le déclencheur `TR_AGGREGATE_DELTA_PATTERNS` à désactiver dans la table de transfert `UACI_ETLPATTERNSTATERUN`.

## **retryAttemptsIfAggregationRunning**

### **Description**

Nombre de fois que le processus ETL tente de vérifier si l'agrégation des rapports est effectuée si l'indicateur de verrouillage est défini. Par défaut, cette propriété est définie sur 3.

## **sleepBeforeRetryDurationInMinutes**

### **Description**

Délai de veille en minutes entre deux tentatives consécutives. Par défaut, cette propriété est définie sur 5 minutes.

## **aggregationRunningCheckSql**

### **Description**

Cette propriété permet de définir un SQL personnalisé, qui peut être exécuté pour déterminer si l'indicateur de verrouillage de l'agrégation des rapports est défini. Par défaut, cette propriété est vide.

Si cette propriété est définie, le processus ETL exécute le SQL ci-après pour obtenir l'indicateur de verrouillage.

```
select count(1) AS ACTIVERUNS from uari_pattern_lock where islock='Y'  
=> If ACTIVERUNS is > 0, lock is set
```

## **aggregationRunningCheck**

### **Description**

Active ou désactive la vérification si l'agrégation des rapports est en cours d'exécution avant que le processus ETL ne soit exécuté. Par défaut, cette propriété est activée.

# Chapitre 15. Unica Interact Simulateur

Cette section décrit toutes les propriétés de configuration de l'environnement du simulateur Unica Interact.

Il n'est pas recommandé d'effectuer d'autres actions liées à la simulation, telles que l'ajout, la copie, la suppression et la modification, lorsqu'une simulation est en cours.

## Interact | simulateur

La catégorie de configuration définit les paramètres à définir pour exécuter le scénario d'analyse de couverture du module Simulateur.

### **numberOfThreads**

#### **Description**

Le nombre d'unités d'exécution utilisées pour exécuter la simulation

#### **Valeur par défaut**

1

### **maxOffersToInclude**

#### **Description**

Le nombre maximum d'offres renvoyées dans chaque appel `getOffers` pour chaque ID d'audience dans le scénario d'analyse de couverture.

#### **Valeur par défaut**

10

### **insertBatchSize**

#### **Description**

Définissez la taille de chaque lot pour la conservation des enregistrements obtenus.

#### **Valeur par défaut**

1000

## Interact | simulateur | scenarioDataSource

Ces configurations sont requises pour exécuter le scénario de l'analyse de la couverture du simulateur.

### **jndiName**

#### **Description**

Cette propriété `jndiName` sert à identifier la source de données JNDI (Java Naming and Directory Interface) qui est définie sur le serveur d'applications (Websphere ou WebLogic) pour les tables de phase de conception d'Interact.

#### **Valeurs par défaut**

Aucune valeur par défaut définie.

### **Schéma**

#### **Description**

Nom du schéma qui contient les tables associées au module de la source de données de la phase de conception d'Interact. Interact insère la valeur de cette propriété devant tous les noms de table. Par exemple, `UACI_IntChannel` devient `schema.UACI_IntChannel`.

Il est 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 nommer votre schéma pour pouvoir exécuter le scénario de couverture.

#### **Valeur par défaut**

Aucune valeur par défaut définie.

### **type**

#### **Description**

Type de la base de données pour la source de données utilisée par les tables de la phase de conception d'Interact auxquelles accède le simulateur Interact.

**Valeur par défaut**

sqlserver

**Valeur valide**

sqlserver | Db2 | Oracle | MariaDB

**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.

**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.

**Valeur par défaut**

-1

## Gestion des erreurs du simulateur

Cette section liste les codes de statut inscrits par l'application dans la table UACI\_SimulationHistory, qui fait partie de la base de données de la phase de conception d'Interact.

En cas d'erreur, l'application affichera le message du scénario qui a échoué sur la page d'exécution du simulateur. Le code de statut détaillé peut être trouvé dans la table UACI\_SimulationHistory de la base de données.

Ci-dessous la liste des possibles codes de statut pour un scénario d'historique d'exécutions : // codes de statut 0-99 pour les informations

Statut	Code	Niveau de gravité	Statut HTTP	Possible message d'interface utilisateur
<i>SUCCESS</i>	0	<i>INFO</i> (information)	<i>OK</i>	Exécution réussie de la simulation
<i>EXECUTION EN COURS</i>	1	<i>INFO</i> (information)	<i>OK</i>	En cours d'exécution
<i>ANNULATION EN COURS</i>	2	<i>INFO</i> (information)	<i>OK</i>	Annulation en cours
<i>ANNULE</i>	3	<i>INFO</i> (information)	<i>OK</i>	Annulé(s)
<i>EXPORTING_TO_-CSV</i>	4	<i>INFO</i> (information)	<i>OK</i>	Exportation au format CSV
<i>EXPORTED_TO_-CSV</i>	5	<i>INFO</i> (information)	<i>OK</i>	Exporté au format CSV

// codes de statut 101-999 pour les erreurs

Statut	Code	Niveau de gravité	Statut HTTP	Possible message d'interface utilisateur
--------	------	-------------------	-------------	--



NOT_ENABLED	101	WARN	SERVICE_UNAVAILABLE	Simulation non activée sur ce serveur d'exécution
ERROR_RETRIEVE_SCENARIO	102	ERROR	INTERNAL_SERVER_ERROR	Erreur lors de la récupération des informations du scénario pour la simulation
INVALID_SCENARIO	103	ERREUR	BAD_REQUEST	Informations du scénario de simulation non valides
ERROR_CREATE_RESULT_TABLE	104	ERREUR	INTERNAL_SERVER_ERROR	Erreur lors de la création de la table de stockage des résultats pour la simulation
ERROR_RETRIEVE_AUDIENCE	105	ERREUR	INTERNAL_SERVER_ERROR	Erreur lors de la récupération des ID d'audience pour la simulation
ERROR_CONNECT_DATABASE	106	ERREUR	INTERNAL_SERVER_ERROR	Erreur de connexion à la base de données {0} pour la simulation
ERROR_PERSIST_RESULT	107	ERREUR	INTERNAL_SERVER_ERROR	Erreur de persistance des résultats dans la base de données pour la simulation
SCENARIO_NOT_FOUND	108	ERREUR	NOT_FOUND	Impossible de trouver un scénario prêt pour l'exécution
GENERIC_ERROR	109	ERREUR	INTERNAL_SERVER_ERROR	Erreur du serveur lors de l'exécution de la simulation

ERROR_UPDATE_RESULT	110	ERREUR	INTERNAL_SERVER_ERROR	Erreur lors de la mise à jour des résultats pour la simulation
ERROR_INVALID_IC	111	ERREUR	BAD_REQUEST	Canal interactif non déployé

// // les codes de statut 1001 et supérieurs concernent uniquement l'interface utilisateur, ils ne seront pas conservés dans la base de données

<b>Statut</b>	<b>Code</b>	<b>Niveau de gravité</b>	<b>Statut HTTP</b>	<b>Possible message d'interface utilisateur</b>
SIMULATION_ALREADY_RUNNING	1001	WARN	PRECONDITION_FAILED	Une simulation est déjà en exécution pour ce scénario
SIMULATION_NOT_FOUND	1002	WARN	NO_CONTENT	Aucune simulation en cours pour ce scénario
SIMULATION_RUNNING	1003	INFO	OK	En cours d'exécution
SIMULATION_NOT_RUNNING	1004	INFO	OK	Simulation non en exécution

# Chapitre 16. Propriétés de configuration de l'environnement de conception d'Unica Interact

Cette section décrit toutes les propriétés de configuration de l'environnement de conception d'Unica Interact.

## Campaign | partitions | partition[n] | reports

La propriété **Campaign | partitions | partition[n] | rapports** définit les différents types de dossiers pour les rapports.

### **offerAnalysisTabCachedFolder**

#### **Description**

La propriété `offerAnalysisTabCachedFolder` indique l'emplacement du dossier qui contient la spécification des rapports d'offre (étendus) transmis en une fois et répertoriés dans l'onglet analyse, accessible via le lien analyse du panneau 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 segmentation (étendus) transmis en une fois et répertoriés dans l'onglet analyse, accessible via le lien analyse du panneau 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='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 (étendus) transmis en une fois et répertoriés dans l'onglet analyse, accessible via le lien analyse du panneau 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']
```

## campaignAnalysisTabDeliverOnDemandFolder

### Description

La propriété `campaignAnalysisTabDeliverOnDemandFolder` indique l'emplacement du dossier qui contient les rapports d'Unica Deliver 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='Deliver Reports']
```

## campaignAnalysisTabInteractOnDemandFolder

### Description

Chaîne du dossier de serveur de rapports pour les rapports Unica Interact.

### **Valeur par défaut**

```
/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']
```

### **Disponibilité**

Cette propriété ne s'applique que si vous installez Unica Interact.

## **interactiveChannelAnalysisTabOnDemandFolder**

### **Description**

Chaîne de dossier du 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 ne s'applique que si vous installez Unica Interact.

## **Campaign | partitions | partition[n] | UnicaInsightsReports**

La propriété Campaign | partitions | partition[n] | UnicaInsightsReports définit les différents types de dossiers pour les rapports.

## **interactAnalysisSectionFolder**

### **Description**

Chaîne du dossier de serveur de rapports pour les rapports Unica Interact.

### **Valeur par défaut**

```
/folder[@name='Unica Interact']
```

### **Disponibilité**

Cette propriété n'est applicable que si vous installez Unica Interact et activez Unica Insights.

## **interactiveChannelAnalysisTabOnDemandFolder**

### **Description**

Chaîne du dossier de serveur de rapports pour les rapports Unica Interact.

### **Valeur par défaut**

```
folder[@name='Unica Interact - Object Specific Reports']
```

### **Disponibilité**

Cette propriété n'est applicable que si vous installez Unica Interact et activez Unica Insights.

## **Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking**

Ces propriétés de configuration définissent les paramètres du module d'historique des réponses et des contacts d'Unica Interact.

### **isEnabled**

#### **Description**

Si la valeur est définie sur *oui*, cela active le module d'historique des réponses et des contacts d'Unica Interact qui copie l'historique des réponses et des contacts d'Unica Interact des tables intermédiaires de l'environnement d'exécution d'Unica Interact dans les tables de l'historique des réponses et des contacts de Unica Campaign. La propriété `interactInstalled` doit également être définie sur *oui*.

#### **Valeur par défaut**

non

#### **Valeurs valides**

oui | non

## Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica 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 l'opération d'extraction, de transformation et de chargement prend plus de 24 heures pour s'exécuter et rate l'heure de début du jour suivant, elle saute ce jour et s'exécute à l'heure planifiée le jour d'après. Par exemple, si l'opération ETL est configurée pour s'exécuter entre 1:00 et 3:00 et que le processus démarre à 1:00 le lundi et se termine à 2:00 le mardi, l'exécution suivante, planifiée initialement pour 1:00 le mardi, sera sautée et l'opération ETL suivante démarrera à 1:00 le mercredi.

La planification de l'opération ETL ne tient pas compte du passage à l'heure d'été. Par exemple, s'il est planifié que l'opération ETL s'exécute entre 01h00 et 03h00, elle pourrait s'exécuter à 00h00 ou à 02h00 lors du passage à l'heure d'été.

### Valeur par défaut

Non

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## processSleepIntervallInMinutes

### Description

Le nombre de minutes pendant lesquelles le module de l'historique des réponses et des contacts d'Unica Interact attend entre les copies des données des tables de transfert d'exécution d'Unica Interact dans les tables de l'historique des réponses et des contacts d'Unica Campaign.



### **Valeur par défaut**

60

### **Valeurs valides**

Tout entier supérieur à zéro.

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica 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 peut être appliquée que si vous avez installé Unica 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**

2:00

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## **purgeOrphanResponseThresholdInMinutes**

### **Description**

Le nombre de minutes pendant lesquelles Unica 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**

Tout entier supérieur à zéro.

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica 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'Unica Interact. Pendant chaque itération, le module d'historique des réponses et des contacts d'Unica Interact traite tous les enregistrements disponibles des tables de transfert. Cependant, tous ces enregistrements sont fragmentés en blocs `maxJDBCInsertSize`.

### **Valeur par défaut**

1000

### **Valeurs valides**

Tout entier supérieur à zéro.

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## **maxJDBCFetchBatchSize**

### **Description**

Nombre maximum d'enregistrements d'un lot JDBC à extraire dans la base de données intermédiaire. Vous devrez peut-être augmenter cette valeur pour régler 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'un bloc de données JDBC lues pendant ETL (extraire, transformer, charger). Dans certains cas, une taille de bloc 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'il 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, cinq arguments sont transmis au script de notification d'achèvement : heure de début, heure de fin, nombre total d'enregistrements CH traités et nombre total d'enregistrements RH. L'heure de début et l'heure de fin sont des valeurs numériques représentant le nombre de millisecondes écoulées depuis 1970. L'argument de statut indique si le travail d'extraction, de transformation et de chargement a abouti ou échoué. La valeur 0 indique que ce travail a abouti. La valeur 1 indique que ce travail a échoué ou qu'il y a des erreurs.

### **Valeur par défaut**

Aucun

## **fetchSize**

### **Description**

Vous permet de définir la fetchSize JDBC lors de la récupération d'enregistrements dans les tables de transfert.

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**

Aucun

## **daysBackInHistoryToLookupContact**

### **Description**

Limite les enregistrements pouvant faire l'objet d'une recherche lors des requêtes d'historique des réponses à ceux effectués au cours du nombre de jours passés spécifié. Dans le cas des bases de données comportant un grand nombre d'enregistrements d'historique des réponses, cette propriété peut permettre de réduire le temps de traitement des requêtes en limitant la période de recherche au nombre de jours indiqué.

Lorsque la valeur pour daysBackInHistoryToLookupContact est supérieure à zéro, une contrainte de date est ajoutée à la requête de jointure RHy. Cela s'avère utile lorsque la table UA\_DtlContactHist est partitionnée en fonction de la date. La contrainte de date limite la recherche d'enregistrements au sein de la contrainte de date."

La valeur par défaut 0 indique que tous les enregistrements font l'objet de la recherche.

### **Valeur par défaut**

0 (zéro)

## Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [Source de données de l'environnement d'exécution]

Ces propriétés de configuration définissent la source de données du module d'historique des réponses et des contacts d'Unica Interact.

### **jndiName**

#### **Description**

La propriété `systemTablesDataSource` permet d'identifier la source de données JNDI (Java™ Naming and Directory Interface) qui est définie sur le serveur d'applications (Websphere ou WebLogic) pour les tables de l'environnement d'exécution Unica Interact.

La base de données d'exécution d'Unica Interact est remplie 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 peut être appliquée que si vous avez installé Unica Interact.

### **databaseType**

#### **Description**

Type de base de données pour la source de données d'exécution d'Unica Interact.

#### **Valeur par défaut**

SQLServer

#### **Valeurs valides**

SQLServer | Oracle | DB2® | MariaDB

## Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## schemaName

### Description

Nom du schéma qui contient les tables de transfert 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 génération de rapports ou d'apprentissage.

## contacté

### Description

Valeur attribuée à la colonne `ContactStatusID` de la table `UA_DtlContactHist` (tables système de Unica 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 Unica Campaign - Guide d'administration.

### Valeur par défaut

2

### Valeurs valides

Un nombre entier supérieur à zéro.

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

Ces propriétés de configuration définissent les réponses (acceptation ou refus) associées à la génération de rapports et à l'apprentissage.

### accepter

#### Description

Valeur attribuée à la colonne `ResponseTypeID` de la table `UA_ResponseHistory` (tables système de Unica 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 Unica Campaign - Guide d'administration.

#### Valeur par défaut

3

#### Valeurs valides

Un nombre entier supérieur à zéro.

#### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

### rejet

#### Description

Valeur attribuée à la colonne `ResponseTypeID` de la table `UA_ResponseHistory` (tables système de Unica 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 Unica Campaign - Guide d'administration.



### **Valeur par défaut**

8

### **Valeurs valides**

Tout entier supérieur à zéro.

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## **Campaign | partitions | partition[n] | Interact | rapport**

Ces propriétés de configuration définissent les noms des rapports lors de l'intégration à 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 des déploiements de canal

## Campaign | partitions | partition[n] | Interact | apprentissage

Ces propriétés de configuration vous permettent de régler le module d'auto-apprentissage.

### 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'Unica Interact possède la valeur `BuiltInLearning`.

#### Valeur par défaut

95

#### Valeurs valides

Entier compris entre 0 et 95, divisible par 5 ou 99.

### validateonDeployment

#### Description

Si la valeur est `No`, Unica Interact ne valide pas le module d'apprentissage lors du déploiement. Si la valeur est `Yes`, Unica Interact valide le module d'apprentissage lors du déploiement.

#### Valeur par défaut

Non

## Valeurs valides

Oui | Non

## maxAttributeNames

### Description

Nombre maximal d'attributs d'apprentissage que l'utilitaire d'apprentissage d'Unica Interact doit surveiller.

Cette propriété s'applique uniquement si la propriété `Interact > offerserving > optimizationType` associée à l'environnement d'exécution d'Unica Interact possède la valeur `BuiltInLearning`.

### Valeur par défaut

10

## Valeurs valides

N'importe quel nombre entier.

## maxAttributeValues

### Description

Nombre maximal de valeurs distinctes que le module d'apprentissage d'Unica 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'Unica 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'Unica Interact possède la valeur `BuiltInLearning`.

### Valeur par défaut

Autre

### Valeurs valides

Chaîne ou 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, quel que soit le score. L'activation de `percentRandomSelection` remplace la propriété de configuration `offerTieBreakMethod`. Lorsque `percentRandomSelection` est activé, cette propriété est définie quel que soit l'état (activé ou désactivé) de l'apprentissage et qu'il s'agisse de l'apprentissage externe ou de l'auto-apprentissage.

### Valeur par défaut

5

## Valeurs valides

Tout entier de 0 (qui désactive la fonction `percentRandomSelection`) à 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'Unica Interact possède la valeur `BuiltInLearning`.

### Valeur par défaut

0.15

### Valeurs valides

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**

Entier supérieur ou égal à zéro.

## **enablePruning**

### **Description**

Si la valeur est définie sur `Oui`, le module d'apprentissage d'Unica 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 | apprentissage | learningAttributes | [attribut d'apprentissage]

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'Unica Interact possède la valeur `BuiltInLearning`.

### Valeur par défaut

Aucune valeur par défaut définie.

## Campaign | partitions | partition[n] | Interact | déploiement

Ces propriétés de configuration définissent les paramètres de déploiement.

## chunkSize

### Description

Taille maximum de fragmentation en Ko pour chaque package de déploiement d'Unica Interact.

### Valeur par défaut

500

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## Campaign | partitions | partition[n] | Interact | serverGroups | [Groupe de serveurs]

Ces propriétés de configuration définissent les réglages du groupe de serveurs.

## **serverGroupName**

### **Description**

Nom du groupe de serveurs d'exécution d'Unica Interact. Ce nom s'affiche sur l'onglet Synthèse du canal interactif.

### **Valeur par défaut**

Aucune valeur par défaut définie.

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [Groupe de serveurs] | instanceURLs | [URL de l'instance]

Ces propriétés de configuration définissent les serveurs d'exécution d'Unica Interact.

## **instanceURL**

### **Description**

Adresse URL du serveur d'exécution d'Unica Interact. Un groupe de serveurs peut contenir plusieurs serveurs d'exécution d'Unica Interact ; cependant, chaque serveur doit être créé sous une nouvelle catégorie.

### **Valeur par défaut**

Aucune valeur par défaut définie.

### **Exemple**

```
http://server:port/interact
```

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.



## Campaign | partitions | partition[n] | Interact | diagramme

Ces propriétés de configuration définissent l'environnement d'exécution d'Unica Interact utilisé pour l'exécution en mode test des diagrammes temps réel.

### **serverGroup**

#### **Description**

Nom du groupe de serveurs d'Unica Interact utilisé par Unica 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 peut être appliquée que si vous avez installé Unica Interact.

### **dataSource**

#### **Description**

Utilisez la propriété `dataSource` pour identifier la source de données physique utilisée par Unica 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'Unica Interact.

#### **Valeur par défaut**

Aucune valeur par défaut définie.

#### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## eventPatternPrefix

### Description

La propriété `eventPatternPrefix` est une valeur de chaîne qui est ajoutée en préfixe aux noms de modèle d'événement afin de permettre leur utilisation dans les expressions des processus Sélection ou Décision au sein des diagrammes temps réel.

Si vous changez cette valeur, vous devez déployer des modifications globales dans le canal interactif pour que cette configuration mise à jour soit appliquée.

### Valeur par défaut

`EventPattern`

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## Campaign | partitions | partition[n] | Interact | whiteList | [NiveauD'audience] | 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 affectations d'offres globales.

## DefaultCellCode

### Description

Le code de cible par défaut utilisé par Unica 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 de code de cible défini dans Unica Campaign

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## Campaign | partitions | partition[n] | Interact | whiteList | [NiveauD'audience] | 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

Unica 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 de code de cible défini dans Unica Campaign

#### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica 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 affectations d'offres individuelles.

### DefaultCellCode

#### Description

Le code de cible par défaut utilisé par Unica 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 de code de cible défini dans Unica Campaign

### **Disponibilité**

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## **Campaign | partitions | partition[n] | server | internal**

Les propriétés de cette catégorie spécifient les paramètres d'intégration et les limites d'ID interne pour la partition Unica Campaign sélectionnée. Si votre installation de Unica Campaign comporte plusieurs partitions, définissez ces propriétés pour chaque partition que vous souhaitez affecter.

### **internalIdLowerLimit**

#### **Catégorie de configuration**

`Campaign|partitions|partition[n]|server|internal`

#### **Description**

Les propriétés `internalIdUpperLimit` et `internalIdLowerLimit` permettent de limiter les ID internes Unica Campaign à une plage spécifiée. Notez que les valeurs sont inclusives : cela signifie que Unica Campaign peut utiliser les limites supérieure et inférieure.

#### **Valeur par défaut**

0 (zéro)

### **internalIdUpperLimit**

#### **Catégorie de configuration**

`Campaign|partitions|partition[n]|server|internal`

### Description

Les propriétés `internalIdUpperLimit` et `internalIdLowerLimit` permettent de limiter les ID internes Unica Campaign à une plage spécifiée. Les valeurs sont inclusives : cela signifie que Unica Campaign peut utiliser les limites supérieure et inférieure. Si Unica Collaborate est installé, définissez la valeur sur 2147483647.

### Valeur par défaut

4294967295

## deliverInstalled

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Indique qu'Unica Deliver est installé. Si vous sélectionnez `yes`, les fonctionnalités Unica Deliver sont disponibles dans l'interface de Unica Campaign.

Le programme d'installation affecte à cette propriété la valeur `yes` pour la partition par défaut de votre installation d'Unica Deliver. Pour les partitions supplémentaires où vous avez installé Unica Deliver, vous devez configurer cette propriété manuellement.

### Valeur par défaut

Non

### Valeurs valides

Oui | Non

## interactInstalled

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

## Description

Après l'installation de l'environnement de conception d'Unica Interact, cette propriété de configuration doit avoir la valeur `Yes` pour activer l'environnement de conception d'Unica Interact dans Unica Campaign.

Si Unica Interact n'est pas installé, spécifiez `No`. La valeur `No` ne supprime pas les menus et options d'Unica Interact de l'interface utilisateur. Pour supprimer les menus et options, vous devez manuellement annuler l'enregistrement d'Unica Interact via l'utilitaire `configTool`.

## Valeur par défaut

non

## Valeurs valides

oui | non

## Disponibilité

Cette propriété ne s'applique que si vous avez installé Unica Interact.

## MO\_UC\_integration

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Permet l'intégration avec Unica Plan pour cette partition si l'intégration est activée dans les paramètres de configuration de la zone **Platform**. Pour plus d'informations, voir Unica Plan and Unica Campaign Integration Guide.

### Valeur par défaut

Non

### Valeurs valides

Oui | Non

## MO\_UC\_BottomUpTargetCells

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Pour cette partition, autorise les cibles du bas dans la liste des populations ciblées si la fonction **MO\_UC\_integration** est activée. Lorsque cette propriété a pour valeur `Yes`, les populations ciblées descendantes et ascendantes sont visibles, mais les populations ciblées ascendantes sont en lecture seule. Pour plus d'informations, voir Unica Plan and Unica Campaign Integration Guide.

### Valeur par défaut

Non

### Valeurs valides

Oui | Non

## Legacy\_campaigns

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Pour cette partition, autorise l'accès aux campagnes créées avant l'intégration de Unica Plan et de Unica Campaign. S'applique uniquement si **MO\_UC\_integration** a pour valeur `Yes`. Les campagnes existantes incluent également les campagnes créées dans Unica Campaign 7.x et liées à des projets Plan 7.x. Pour plus d'informations, voir Unica Plan and Unica Campaign Integration Guide.

### Valeur par défaut

Non

### Valeurs valides

Oui | Non

## Unica Plan - Intégration d'offre

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Permet d'utiliser Unica Plan pour effectuer des tâches de gestion de cycle de vie d'une offre sur cette partition, si la fonction **MO\_UC\_integration** est activée pour cette partition. L'intégration d'offre doit être activée dans les paramètres de configuration de la zone **Platform**. Pour plus d'informations, voir Unica Plan and Unica Campaign Integration Guide.

### Valeur par défaut

Non

### Valeurs valides

Oui | Non

## UC\_CM\_integration

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Permet l'intégration de segment en ligne Digital Analytics pour une partition Unica Campaign. Si vous paramétrez cette valeur sur `Yes`, le processus de sélection dans un diagramme permet de sélectionner des **segments Digital Analytics** en entrée. Pour configurer l'intégration de Digital Analytics pour chaque partition, sélectionnez **Paramètres > Configuration > Unica Campaign | partitions | partition[n] | Coremetrics**.

### Valeur par défaut

Non

### Valeurs valides

Oui | Non



## linkInstalled

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Indique que Link est installé. Lorsque vous sélectionnez Oui, la fonction Gérer les connexions Link est disponible dans l'interface d'Unica Campaign. Le programme d'installation définit cette propriété sur Non pour la partition par défaut dans votre installation de Link. Pour les partitions supplémentaires où vous avez installé Link, vous devez configurer cette propriété manuellement.

### Valeur par défaut

Non

### Valeurs valides

Oui | Non

## numRowsReadToParseDelimitedFile

### Catégorie de configuration

`Campaign|partitions|partition[n]|server|internal`

### Description

Cette propriété est utilisée lors du mappage d'un fichier délimité en tant que table utilisateur. Elle est également utilisée par la zone de processus Scoring lors de l'importation d'un fichier de sortie de score à partir d'IBM SPSS Modeler Advantage Enterprise Marketing Management Edition. Pour importer ou mapper un fichier délimité, Unica Campaign doit l'analyser afin d'identifier les colonnes, les types de données (types de zones) et les largeurs de colonne (longueurs de zones).

La valeur par défaut 100 signifie que Unica Campaign examine les 50 premières entrées de ligne et les 50 dernières entrées de ligne du fichier délimité. Unica Campaign alloue ensuite la longueur de zone en fonction de la valeur la plus élevée qu'il a trouvée parmi ces entrées. Dans la plupart des cas,

la valeur par défaut est suffisante pour déterminer les longueurs de zones. Toutefois, dans les fichiers délimités de très grande taille, une nouvelle zone peut dépasser la longueur estimée calculée par Unica Campaign, ce qui peut provoquer une erreur durant l'exécution du diagramme. Par conséquent, si vous mappez un fichier de très grande taille, vous pouvez augmenter cette valeur afin que Unica Campaign examine un plus grand nombre d'entrées de ligne. Par exemple, la valeur 200 signifie que Unica Campaign examine les 100 premières entrées de ligne et les 100 dernières entrées de ligne du fichier.

Avec la valeur 0, c'est la totalité du fichier qui est examinée. En règle générale, cela n'est nécessaire que si vous importez ou mappez des fichiers possédant des largeurs de zones de données variables qui ne peuvent être identifiées en lisant les premières et les dernières lignes. La lecture du fichier entier pour les très grands fichiers peut augmenter le temps de traitement nécessaire pour le mappage des tables et les exécutions de processus Scoring.

#### **Valeur par défaut**

100

#### **Valeurs valides**

0 (toutes les lignes) ou tout entier positif

## Campaign | surveillance

Les propriétés de cette catégorie indiquent si la fonctionnalité 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 au statut du diagramme.

Cette propriété n'est pas disponible dans les versions antérieures à Unica 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 de suivi.

Cette propriété n'est pas disponible dans les versions antérieures à Unica 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 Unica Campaign 7.0.

### Valeur par défaut

FAUX

### Valeurs valides

TRUE | FALSE

## 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 Unica Campaign est configuré afin d'utiliser les communications SSL (Secure Sockets Layer), définissez la valeur de cette propriété pour que HTTPS soit utilisé. 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.
- `SSL_port` est 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 définie est `TRUE`, le serveur de connecteurs JMX Unica Campaign d'Unica Interact est activé. Unica Campaign n'a pas de sécurité JMX.

Si la valeur définie est `FALSE`, vous ne pouvez pas vous connecter au serveur de connecteurs JMX de Unica Campaign.

Cette surveillance JMX est destinée uniquement au module d'historique des réponses et des contacts d'Unica Interact.

### Valeur par défaut

`FAUX`

### Valeurs valides

`TRUE` | `FALSE`

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## protocol

### Description

Protocole d'écoute du serveur de connecteurs JMX de Unica Campaign, si la propriété `monitorEnabledForInteract` est paramétrée sur `yes`.

Cette surveillance JMX est destinée uniquement au module d'historique des réponses et des contacts d'Unica Interact.

### Valeur par défaut

JMXMP

### Valeurs valides

JMXMP | RMI

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## port

### Description

Port d'écoute du serveur de connecteurs JMX de Unica Campaign, si la propriété `monitorEnabledForInteract` est paramétrée sur `yes`.

Cette surveillance JMX est destinée uniquement au module d'historique des réponses et des contacts d'Unica Interact.

### Valeur par défaut

2004

### Valeurs valides

Un nombre entier entre 1 025 et 65 535.

### Disponibilité

Cette propriété ne peut être appliquée que si vous avez installé Unica Interact.

## Campaign | partitions | partition[n] | Interact | outboundChannels

Ces propriétés de configuration vous permettent d'optimiser les canaux sortants des messages déclenchés.



**Remarque :** Les passerelles prêtes à l'emploi, dont le nom est "EMail", "MobilePush" et "UBX", sont disponibles sous ce nœud.

### category name

#### Description

Cette propriété définit le nom de ce canal sortant. Ce nom doit être unique parmi tous les canaux sortants.

### nom

#### Description

Nom de votre canal sortant.



**Remarque :** Vous devez redémarrer le serveur d'applications pour que les modifications soient appliquées.

## Campaign | partitions | partition[n] | Interact | outboundChannels | Données de paramètre

Ces propriétés de configuration vous permettent d'optimiser les canaux sortants des messages déclenchés.

### category name

#### Description

Cette propriété définit le nom de ce paramètre. Ce nom doit être unique parmi tous les paramètres de ce canal sortant.

## valeur

### Description

Cette propriété définit les paramètres, au format de paires nom/valeur, requis par cette passerelle sortante.

## Campagne | partitions | partition[n] | Interact | Simulateur

Ces propriétés de configuration définissent le groupe de serveurs que vous souhaitez utiliser pour exécuter des simulations d'API.

### serverGroup

#### Description

Indiquez le groupe de serveurs d'exécution qui est utilisé pour exécuter des simulations d'API.

#### Valeur par défaut

defaultServerGroup

## offerArbitration

### Description

OfferArbitration vous permet de réaliser des configurations diverses liées aux offres pour la phase de conception d'Interact. Il vous permet de définir Campaign comme propriété facultative de stratégie. Il dispose aussi d'options de configuration permettant l'utilisation d'API Centralized Offers Management si elles sont installées.

### isCampaignRequiredForStrategy

Ce paramètre peut avoir deux valeurs.

- Oui : Il est nécessaire de sélectionner une campagne lorsque vous créez/éditez/copiez une stratégie.
- Non : Campaign n'est pas requis.

### **Valeur par défaut**

Non

### **relyOnOfferManagement**

Définissez cette option de configuration sur "Oui" si Centralized Offer Management est installée. Cela permet d'utiliser des API Centralized Offers Management dans la phase de conception d'Interact afin d'accéder aux informations liées aux offres. Ce paramètre peut avoir deux valeurs.

- Oui : Activez l'utilisation des API Centralized Offers Management.
- Non : N'utilisez pas d'API Centralized Offers Management.

### **Valeur par défaut**

Non

### **offerManagementURL**

URL du serveur Centralized Offers Management.

### **Valeur par défaut**

Aucune valeur par défaut définie

### **Exemple**

http://server:port/Offer

### **Disponibilité**

Cette propriété ne s'applique que si vous avez installé Centralized Offers Management.

### **connectionTimeout**

Délai de connexion de l'API en secondes.



## **Valeur par défaut**

5 secondes

# Chapitre 17. 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 mise en œuvre de code Java™ détaillé ni d'appels SOAP au serveur Unica 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. 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 Unica Interact.

Votre installation Unica Interact inclut deux connecteurs pour la personnalisation offre lancée côté client :

- [A propos de Unica Interact Message Connector \(à la page 614\)](#). 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 Unica 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 Unica Interact Web Connector \(à la page 636\)](#). Avec Message Connector, (également appelé JS Connector), les pages Web peuvent utiliser du JavaScript™ 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 Unica Interact Message Connector

Unica Interact Message Connector permet aux emails et aux autres médias électroniques d'appeler 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 email). Il contient à la fois une URL de balise `IMG` qui passe les informations lorsque le document s'ouvre sur le serveur Unica 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 Unica Interact suite aux clics :

```
<a href="http://www.example.com/MessageConnector/offerClickthru.jsp?msgId=1234&linkId=1&userid=1&referral=test"></a>
```

Dans l'exemple suivant, une balise `IMG` est imbriquée dans une balise `A`, et génère le comportement suivant :

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, ainsi que les paramètres de client qui incluent l'ID utilisateur, le niveau de revenu, et le type de revenu.
2. Ces informations sont passées via un appel d'API au serveur d'exécution Unica 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'arrivée 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, telle 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 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 \(à la page 616\)](#).
- 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 \(à la page 627\)](#).
- Installation de l'application Web Message Connector comme indiqué dans [Déploiement et exécution de Message Connector \(à la page 629\)](#).
- Création des balises `IMG` et `A` dans vos spots marketing (e-mails 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 \(à la page 631\)](#).

## 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 Unica Interact, comme pour `<Unica 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 Unica 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 Unica Interact sur lequel Message Connector s'exécute et auquel les balises de la page Message Connector doivent se connecter.
  - `<interactUsername>`. Le nom d'utilisateur de Unica Interact authentifie Message Connector avec l'exécution de Unica Interact. Il est requis lorsque l'authentification de l'API Unica Interact (`Affinium|interact|general|API`) est définie sur `True`.
  - `<interactPassword>`. Le mot de passe de Unica Interact authentifie Message Connector avec l'exécution de Unica Interact. Il est requis lorsque l'authentification de l'API Unica Interact (`Affinium|interact|general|API`) est définie sur `True`. Seuls les mots de passe chiffrés à l'aide de l'utilitaire `encryptPasswords` de Platform sont valides. Les mots de passe en texte normal ne sont pas autorisés.
  - `<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 niveau d'audience, et qui indique le niveau d'audience par défaut si aucun n'est spécifié par le lien de Message Connector. Au moins un niveau d'audience doit être configuré.

Tous les paramètres de configuration sont décrits plus en détail dans [Paramètres de configuration de Message Connector \(à la page 618\)](#).

4. Lorsque vous avez terminé les changements 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 Unica Interact, en général `<Unica 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 30. Paramètres généraux de Message Connector**

**Liste des paramètres facultatifs et obligatoires de la section `generalSettings` du fichier de configuration de Message Connector**

<code>&lt;interactURL&gt;</code>	Adresse URL du serveur d'exécution Unica 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.	<code>http://local-host:7001/interact</code>
----------------------------------	---	--

**Tableau 30. Paramètres généraux de Message Connector**

Liste des paramètres facultatifs et obligatoires de la section `generalSettings` du fichier de configuration de Message Connector

(suite)

<code>&lt;interactUsername&gt;</code>	Le nom d'utilisateur de Unica Interact authentifie Message Connector avec l'exécution de Unica Interact. Il est requis lorsque l'authentification de l'API Unica Interact ( <code>Affinium interact general API</code> ) est définie sur <code>True</code> .	
<code>&lt;interactPassword&gt;</code>	Le mot de passe de Unica Interact authentifie Message Connector avec l'exécution de Unica Interact. Il est requis lorsque l'authentification de l'API Unica Interact ( <code>Affinium interact general API</code> ) est définie sur <code>True</code> . Seuls les mots de passe chiffrés à l'aide de l'utilitaire <code>encryptPasswords</code> de Platform sont valides. Les mots de passe en texte normal ne sont pas autorisés.	
<code>&lt;defaultDateTimeFormat&gt;</code>	Format de date par défaut.	<code>MM/dd/yyyy</code>
<code>&lt;log4jConfigFileLocation&gt;</code>	Emplacement du fichier de propriétés Log4j. Il est	<code>config/MessageConnectorLog4j.properties</code>

### Tableau 30. Paramètres généraux de Message Connector

Liste des paramètres facultatifs et obligatoires de la section `generalSettings` du fichier de configuration de Message Connector

(suite)

	relatif à la variable d'environnement <code>\$MESSAGE_CONNECTOR_HOME</code> si elle est définie. Sinon, cette valeur est relative au chemin de la racine de l'application Web Message Connector.
--	--

### 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 31. Paramètres par défaut de Message Connector

Éléments contenus dans la section `defaultParameterValues` du fichier `MessageConnectorConfig.xml`.

Élément	Description	Valeur par défaut
<code>&lt;interactiveChannel&gt;</code>	Nom du canal interactif par défaut.	
<code>&lt;interactionPoint&gt;</code>	Nom du point d'interaction par défaut.	
<code>&lt;debugFlag&gt;</code>	Détermine si le débogage est activé. Les valeurs admises sont <code>true</code> et <code>false</code> .	<code>false</code>



### Tableau 31. Paramètres par défaut de Message Connector

Éléments contenus dans la section `defaultParameterValues` du fichier

*MessageConnectorConfig.xml*.

(suite)

Élément	Description	Valeur par défaut
<code>&lt;contactEventName&gt;</code>	Nom par défaut de l'événement de contact publié.	
<code>&lt;acceptEventName&gt;</code>	Nom par défaut de l'événement d'acceptation publié.	
<code>&lt;imageUrlAttribute&gt;</code>	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.	
<code>&lt;landingPageUrlAttribute&gt;</code>	Adresse URL par défaut de la page d'arrivée 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 32. Paramètres de comportement de Message Connector****Éléments contenus dans la section `behaviorSettings` du fichier*****MessageConnectorConfig.xml***

Élément	Description	Valeur par défaut
<code>&lt;imageErrorLink&gt;</code>	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.	<code>/images/default.jpg</code>
<code>&lt;landingPageError-Link&gt;</code>	URL vers laquelle le connecteur redirige si une erreur se produit lors du traitement d'une requête d'une page d'arrivée de clics. Ce paramètre est obligatoire.	<code>/jsp/default.jsp</code>
<code>&lt;alwaysUseExistingOffer&gt;</code>	Détermine si l'offre en mémoire cache doit être renvoyée, même si elle a déjà expiré. Les valeurs admises sont <code>true</code> et <code>false</code> .	<code>false</code>
<code>&lt;offerExpireAction&gt;</code>	L'action à entreprendre si l'offre initiale est trouvée, mais a déjà expiré. Les valeurs admises sont les suivantes : <ul style="list-style-type: none"> <li>• <code>GetNewOffer</code></li> <li>• <code>RedirectToErrorPage</code></li> <li>• <code>ReturnExpiredOffer</code></li> </ul>	<code>RedirectToErrorPage</code>

## Paramètres de stockage

Le tableau suivant contient la liste des paramètres facultatifs et obligatoires contenus dans la section `storageSettings` du fichier `MessageConnectorConfig.xml`.

**Tableau 33. Paramètres de stockage de MessageConnector**

**Éléments contenus dans la section `storageSettings` du fichier**

**`MessageConnectorConfig.xml`.**

Élé- ment	Description	Valeur par défaut
<code>&lt;persist- tence- Mode&gt;</code>	Lorsque le cache persiste de nouvelles entrées dans la base de données. Les valeurs autorisées sont <code>WRITE-BEHIND</code> (où les données sont écrites initialement dans la mémoire cache et mises à jour ultérieurement dans la base de données) et <code>WRITE-THROUGH</code> (où les données sont écrites dans la mémoire cache et dans la base de données en même temps).	<code>WRITE-THROUGH</code>
<code>&lt;max- Cache- Size&gt;</code>	Nombre maximal d'entrées dans le mémoire cache.	<code>5000</code>
<code>&lt;max- Per- sis- tence- Batch- Size&gt;</code>	Taille de lot maximale lors de la conservation des entrées dans la base de données.	<code>200</code>
<code>&lt;mac- Cache- Per- Per- &gt;</code>	Durée maximale en secondes pendant laquelle une entrée demeure dans la mémoire cache avant d'être conservée dans la base de données.	<code>3</code>

**Tableau 33. Paramètres de stockage de MessageConnector**

Éléments contenus dans la section `storageSettings` du fichier

*MessageConnectorConfig.xml*.

(suite)

Élé- ment	Description	Valeur par défaut
<code>sist- Inter- val&gt;</code>		
<code>&lt;maxe- lemen- ton- disk&gt;</code>	Nombre maximal d'entrées dans le cache disque.	5000
<code>&lt;ca- che- Entry- Time- ToEx- pire- InSe- conds&gt;</code>	Durée maximale pendant laquelle les entrées présentes dans le cache disque sont conservées avant expiration.	60 000
<code>&lt;jdbc- Set- tings&gt;</code>	Section du fichier XML contenant des informations spécifiques si une connexion JDBC est utilisée. Cet élément et la section <code>&lt;dataSourceSettings&gt;</code> s'excluent mutuellement.	Configuré par défaut pour se connecter à une base de données Serveur SQL 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.

### Tableau 33. Paramètres de stockage de MessageConnector

Éléments contenus dans la section `storageSettings` du fichier

*MessageConnectorConfig.xml*.

(suite)

Élé- ment	Description	Valeur par défaut
<code>&lt;data- Sour- ceSet- tings&gt;</code>	Section du fichier XML contenant des informations spécifiques si une source de données est utilisée. Cet élément et la section <code>&lt;jdbcSet- tings&gt;</code> s'excluent mutuellement.	Configuré par défaut pour se connecter à la source de données InteractDS définie sur le serveur d'applications Web local.

### Niveaux d'audience

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 niveau d'audience défini dans ce fichier de configuration.

**Tableau 34. Paramètres de niveaux d'audience MessageConnector**

Liste des paramètres facultatifs et obligatoires contenus dans la section `audienceLevels` du fichier `MessageConnectorConfig.xml`.

Élément	Élément	Description	Valeur par défaut
<code>&lt;audienceLevel&gt;</code>		Élément contenant la configuration du niveau d'audience. Fournissez un attribut de nom, comme dans <code>&lt;audienceLevel name="Customer"&gt;</code> .	
	<code>&lt;messageLogTable&gt;</code>	Nom de la table de journal. Cette valeur est obligatoire.	<code>UACI_MESSAGE_CONNECTOR_LOG</code>
<code>&lt;fields&gt;</code>	<code>&lt;field&gt;</code>	Définition d'une ou de plusieurs zones d'ID audience pour ce <code>audienceLevel</code> .	
	<code>&lt;name&gt;</code>	Nom de la zone d'ID audience, comme indiqué dans l'exécution Interact.	
	<code>&lt;httpParameterName&gt;</code>	Nom du paramètre correspondant à cette zone d'ID audience.	
	<code>&lt;dbColumnName&gt;</code>	Nom de la colonne correspondante dans la base de données	

### Tableau 34. Paramètres de niveaux d'audience MessageConnector

Liste des paramètres facultatifs et obligatoires contenus dans la section `audienceLevels` du fichier `MessageConnectorConfig.xml`.

(suite)

Élément	Élément	Description	Valeur par défaut
		pour cette zone d'ID audience.	
	<code>&lt;type&gt;</code>	Type de la zone d'ID audience, tel qu'il est indiqué dans l'exécution Interact. Les valeurs peuvent être <code>string</code> ou <code>numeric</code> .	

## Création des tables de Message Connector

Avant de déployer Unica Interact Message Connector, vous devez d'abord créer les tables dans la base de données dans laquelle sont stockées les données d'exécution Unica Interact. Vous devez créer une table pour chaque niveau d'audience que vous avez défini. Pour chaque niveau d'audience, Unica 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 d'Unica Interact. Consultez les feuilles de travail que vous avez complétées dans le *Unica Interact guide d'installation* pour plus de détails sur la connexion à la base de données contenant les tables d'exécution Unica Interact.

1. Lancez votre client de base de données et connectez-vous à la base de données dans laquelle vos tables d'exécution Unica Interact sont stockées.
2. Exécutez le script approprié dans le répertoire `<Unica Interact_home>/msgconnector/scripts/ddl`. Le tableau suivant répertorie les exemples de script SQL que vous pouvez utiliser pour créer manuellement les tables Message Connector suivantes :

**Tableau 35. Scripts de création des tables Message Connector**

Type de source de données	Nom du script
IBM® DB2®	<code>db_scheme_db2.sql</code>
Microsoft™ SQL Server	<code>db_scheme_sqlserver.sql</code>
Oracle	<code>db_scheme_oracle.sql</code>
MariaDB	<code>db_scheme_mariadb.sql</code>

Notez que ces scripts sont fournis à titre d'exemple. Si vous utilisez une autre convention de dénomination ou une autre structure pour les valeurs d'ID d'audience, 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 niveau d'audience.

Les tables créées doivent contenir les informations suivantes :

**Tableau 36. Informations créées par les exemples de script SQL**

**Liste des colonnes et des descriptions créées par les exemples de script SQL.**

Nom de colonne	Description
<code>LogId</code>	Clé primaire de cette entrée.
<code>MessageId</code>	Identificateur unique de chaque instance de messagerie.
<code>LinkId</code>	Identificateur unique de chaque lien dans le support électronique (tel qu'un e-mail).



Nom de colonne	Description
OfferImageUrl	Adresse URL d'accès à l'image associée à l'offre renvoyée.
OfferLandingPage-Url	Adresse URL d'accès à la page d'arrivée associée à l'offre renvoyée.
TreatmentCode	Code de traitement de l'offre renvoyée.
OfferExpiration-Date	Date et heure d'expiration de l'offre renvoyée.
OfferContactDate	Date et l'heure du renvoi de l'offre au client.
AudienceId	ID d'audience du support électronique.

Notez ce qui suit concernant cette table :

- Selon le niveau d'audience, il existe une seule colonne AudienceId pour chaque composant de la clé d'audience.
- 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

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 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 Unica 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 Unica Interact, en vue d'une utilisation par Message Connector comme indiqué dans [Création des tables de Message Connector \(à la page 627\)](#)

De la même manière que vous déployez d'autres applications 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 `<Unica Interact_home>/msgconnector/MessageConnector.war`.  
Remplacez `<Unica Interact_home>` par le répertoire réel dans lequel le serveur d'exécution d'Unica Interact est installé.
3. Les étapes suivantes sont requises pour configurer un connecteur de message avec JBOSS
  - a. Ajoutez un module de base de données.
  - b. Notez le nom du module donné lors de sa création.
  - c. Accédez à `<INSTALL_DIR>/Interact/msgconnector/MessageConnector.war/WEB-INF/jboss-deployment-structure.xml`, supprimez la mise en commentaire de la section dans la dépendance et donnez au module le même nom que celui donné lors de la création du module. Par exemple : - Si le module créé lors de la configuration de JBoss pour DB2 est ---> com.ibm, alors le fichier `jboss-depolymnt-structure.xml` doit disposer des entrées ci-dessous : 

```
<?xml
version="1.0" encoding="UTF-8"?><deployment><dependencies><module
name="com.ibm" export="true"/></dependencies></deployment></jboss-
deployment-structure>
```

Message Connector est désormais prêt à être utilisé. Lorsque vous avez configuré votre installation Unica 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 email), 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 passer 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 de l'audience à laquelle le destinataire du message appartient.

Notez que si l'ID audience est un ID composite, tous ces composants doivent être inclus dans le lien.

Vous pouvez également inclure des paramètres facultatifs qui comprennent le niveau d'audience, 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 (niveau d'audience, 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 échantillons de liens, voir [Paramètres de demande HTTP "IMG" et "A" \(à la page 632\)](#)

## Paramètres de demande HTTP "IMG" et "A"

Lorsque Message Connector reçoit une demande, soit parce qu'un utilisateur final a ouvert un e-mail contenant une balise `IMG` encodée par Message Connector, soit parce que l'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 de l'e-mail), soit la balise `A` (chargée lorsque la personne affichant l'e-mail 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	Requis ?	Valeur par défaut
msgId	Identificateur unique de l'instance d'e-mail 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 cet e-mail 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 d'audience auquel le destinataire de cette communication appartient.	Non	L'audienceLevel indiqué par défaut dans l'élément <code>audienceLevels</code> trouvé dans le fichier <code>MessageConnectorConfig.xml</code> .
ic	Nom du canal interactif cible (IC)	Non	Valeur de l'élément <code>interactiveChannel</code> trouvé dans la section <code>defaultParameterValues</code> du fichier <code>MessageConnectorConfig.xml</code> , qui est "interactiveChannel" par défaut.
ip	Nom du point d'interaction (IP).	Non	Valeur de l'élément <code>interactionPoint</code> trouvé dans la section <code>defaultParameterValues</code> du fichier <code>MessageConnectorConfig.xml</code> , qui est "headBanner" par défaut.
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 offre cible.	Non	Valeur de l'élément <code>imageUrlAttribute</code> trouvé dans la section <code>defaultParameterValues</code> du fichier <code>MessageConnectorConfig.xml</code> .
offerLandingPageUrl	URL de la page d'arrivée correspondant à l'offre cible.	Non	Aucun.
offerLandingPageUrlAttr	Nom de l'attribut de l'offre contenant l'URL de la page	Non	Valeur de l'élément <code>landingPageUrlAttribute</code> trouvé dans la section <code>defaultParame-</code>

Nom du paramètre	Description	Requis ?	Valeur par défaut
	d'arrivée correspondant à l'offre cible.		<code>terValues</code> du fichier <code>MessageConnectorConfig.xml</code> .
<code>contactEvent</code>	Nom de l'événement du contact.	Non	Valeur de l'élément <code>contactEventName</code> trouvé dans la section <code>defaultParameterValues</code> du fichier <code>MessageConnectorConfig.xml</code> , qui est "contact" par défaut.
<code>responseEvent</code>	Nom de l'événement d'acceptation.	Non	Valeur de l'élément <code>acceptEventName</code> trouvé dans la section <code>defaultParameterValues</code> du fichier <code>MessageConnectorConfig.xml</code> , qui est "accept" par défaut.
<code>débogage</code>	Indicateur de débogage. Définissez ce paramètre sur "true" uniquement pour l'identification des incidents et si vous le support technique vous l'a demandé.	Non	Valeur de l'élément <code>debugFlag</code> trouvé dans la section <code>defaultParameterValues</code> du fichier <code>MessageConnectorConfig.xml</code> , qui est "false" par défaut.
<code>&lt;audience id&gt;</code>	ID audience 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 Unica Interact et le passe à l'exécution Unica Interact.

Les valeurs du paramètre Type peuvent être les suivantes :

- chaîne
- numeric
- date/heure

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/dd/yyyy"`.

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 Unica 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 Unica 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 e-mail :

```
<a
  href="http://www.example.com/MessageConnector/offerClickthru.jsp?msgId=234
    &linkId=1&userid=1&referral=xyz" >
  
</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 Unica 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'arrivée, l'identificateur de message unique (`msgId`), l'identificateur de lien (`linkId`) et l'identificateur de l'utilisateur (`userid`) sont transmis au serveur, ainsi qu'un paramètre supplémentaire (`referral`) qui est transmis à l'adresse URL cible de la redirection.

## A propos de Unica Interact Web Connector

Unica Interact WebConnector (également appelé JavaScript™ Connector, ou JSConnector) fournit un service sur le serveur d'exécution Unica Interact qui permet au code JavaScript™ d'appeler l'API Unica Interact Java™. Cela permet aux pages Web d'appeler Unica Interact pour une 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 Unica Interact. De cette façon, chaque fois que la page se charge, des appels sont envoyés à l'API Unica Interact pour garantir que les meilleures offres s'affichent sur la page de chargement pour le visiteur du site.

Utilisez Unica Interact Web Connector 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 Unica Interact Web Connector sont installés automatiquement sur votre serveur d'exécution Unica Interact, dans le répertoire `<Unica Interact_home>/jsconnector`. Dans le répertoire `<Unica Interact_home>/jsconnector`, vous trouverez un fichier `ReadMe.txt` contenant des notes et des détails importants sur les fonctionnalités 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 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 Unica 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, à savoir `<Unica Interact_Home>/jsconnector`.

La manière dont vous définissez les propriétés Java™ dépend de votre serveur d'applications Web. Par exemple, dans WebLogic, vous éditeriez 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 Unica 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 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 Unica 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 Unica Interact.

Avant de déployer Web Connector, vous devez avoir installé le serveur d'exécution 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 Unica Interact.

1. Copiez le répertoire `jsconnector` contenant les fichiers du Web Connector à partir de votre serveur d'exécution Unica Interact sur le système sur lequel le serveur d'applications Web (tel que WebSphere® Application Server) est déjà configuré et en cours d'exécution.

Le répertoire `jsconnector` se trouve sous le répertoire d'installation d'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 Unica 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`. Voir [Configuration du Web Connector \(à la page 640\)](#) pour plus d'informations.

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.

La manière dont vous définissez les propriétés Java™ dépend de votre serveur d'applications Web. Par exemple, dans WebLogic, vous éditeriez 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=/Unica
InteractFiles/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 Unica Interact est entièrement configuré et opérationnel, l'étape suivante consiste à vous connecter à la page de Configuration de Web Connector, à l'adresse `http:// <host>:<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 paramètres de configuration du Unica Interact Web Connector 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 Unica Interact lui-même, ou un système distinct exécutant un serveur d'applications 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 Unica Interact, une instance de Web Connector est installée automatiquement lorsque vous installez et déployez Unica 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 638).

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 37. Récapitulatif des paramètres de configuration de Web Connector**

**Pour chaque section de la page Paramètres de configuration de Web Connector, une brève description sur la méthode d'utilisation est fournie. Des informations complémentaires sont disponibles à un autre emplacement.**

Section	Paramètres
Paramètres de base	<p>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 Unica 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.</p> <p>Pour plus de détails, voir <a href="#">Options de base de configuration de Web Connector</a> (à la page 644).</p>
HTML Display Types	<p>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</p>

Section	Paramètres
	<p>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.</p> <p>Les paramètres de configuration de cette page correspondent à la section <code>interactionPoints</code> du fichier de configuration <code>jsconnector.xml</code>.</p> <p>Pour plus de détails, voir <a href="#">Types d'affichage HTML de la configuration Web Connector (à la page 648)</a>.</p>
Enhanced Pages	<p>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 spécifiques de chargement de page et les points d'interaction définis pour ce mappage.</p> <p>Les paramètres de configuration de cette page correspondent à la section <code>pageMapping</code> du fichier de configuration <code>jsconnector.xml</code>.</p> <p>Pour plus de détails, voir <a href="#">Enhanced Pages - Configuration de Web-Connector (à la page 651)</a>.</p>

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 Digital Analytics for On Premises 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 naviguez 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 Unica 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 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 Unica 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 définis.

La sauvegarde des fichiers de configuration est stockée

dans `<jsconnector_home>/conf/archive/`

`jsconnector.xml.<date>.<time>`, ainsi que dans

`jsconnector.xml.20111113.214933.750-0500` (où la chaîne de date est 20111113 et la chaîne temporelle, 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 (`<Unica 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 Unica 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 38. Paramètres appliqués à tout le site pour l'installation de Web Connector****Paramètres concernant toute l'installation de Web Connector**

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
<b>Interact API URL</b>	URL de base du serveur d'exécution Unica Interact.   <b>Remarque</b> : 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).	<interact-URL>
<b>Web Connector URL</b>	URL de base utilisée pour générer l'URL de clics.	<jsConnectorURL>
<b>Interactive Channel name for the target website</b>	Nom du canal interactif que vous avez défini sur le serveur Unica Interact qui représente ce mappage de page.	<interactive-Channel>
<b>Audience Level of Visitors</b>	Référentiel Unica Campaign du visiteur entrant ; utilisé dans l'appel d'API à l'exécution Unica Interact.	<audience-Level>
<b>Audience ID Field Name in the Profile Table</b>	Nom de la zone audienceId field qui sera utilisé dans l'appel d'API Unica Interact. Les identificateurs d'audience multi-zone ne sont actuellement pas pris en charge.	<audience-IdField>
<b>Data type of the Audience ID Field</b>	Type de données de la zone d'ID audience ("numérique" ou "chaîne") à utiliser dans l'appel de l'API Unica Interact.	<audienceIdFieldType>

**Tableau 38. Paramètres appliqués à tout le site pour l'installation de Web Connector****Paramètres concernant toute l'installation de Web Connector****(suite)**

<b>Paramètre</b>	<b>Description</b>	<b>Paramètre équivalent dans jsconnector.xml</b>
<b>Cookie Name that represents the Session ID</b>	Nom du cookie qui contiendra l'ID de session.	<code>&lt;sessionId-Cookie&gt;</code>
<b>Cookie Name that represents the Visitor ID</b>	Nom du cookie qui contiendra l'ID du visiteur.	<code>&lt;visitorId-Cookie&gt;</code>

**Fonctions facultatives**

Les options de configuration des fonctionnalités facultatives sont des paramètres globaux facultatifs pour l'installation du Web Connector que vous configurez. Vous pouvez indiquer les valeurs suivantes :

**Tableau 39. Paramètres appliqués à tout le site pour l'installation de Web Connector****Paramètres facultatifs concernant toute l'installation de Web Connector**

<b>Para- mètre</b>	<b>Description</b>	<b>Para- mètre équi- valent dans jscon- nec- tor.xml</b>
Enable Debug Mode	Indique (réponse <i>yes</i> ou <i>no</i> ) si vous souhaitez utiliser un mode de débogage spécial. Si vous le définissez cette fonctionnalité, 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 <i>&lt;authorizedDebugClients&gt;</i> afin d'obtenir l'alerte.	<i>&lt;enable-Debug-Mode&gt;</i>
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.	<i>&lt;authorized-Debug-Clients&gt;</i>
Enable Digital Analytics for On Premises Page Tag	Indique (avec un <i>yes</i> ou <i>no</i> réponse) si Web Connector doit joindre la balise spécifiée Digital Analytics for On Premises à la fin du contenu de la page.	<i>&lt;enable-NetInsight-Tagging&gt;</i>

**Tableau 39. Paramètres appliqués à tout le site pour l'installation de Web Connector****Paramètres facultatifs concernant toute l'installation de Web Connector****(suite)**

<b>Para- mètre</b>	<b>Description</b>	<b>Para- mètre équi- valent dans jscon- nec- tor.xml</b>
Integra- tion		
Digital Analy- tics for On Pre- mises Tag HTML Tem- plate File	Modèle HTML/Javascript permettant d'intégrer un appel à la balise Digital Analytics for On Premises. En général, vous devez accepter la valeur par défaut, sauf si vous êtes invité à fournir un modèle différent.	<net- In- sight- Tag>

**Types d'affichage HTML de la configuration Web Connector**

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.



**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 chargés automatiquement à partir des fichiers stockés dans le répertoire `<Unica 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
<b>File name for this display type</b>	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 <code>CHANGE_ME.flt</code> . Vous devez la remplacer par une valeur significative avant de continuer.	<code>&lt;htmlSnippet&gt;</code>

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
<b>HTML Snippet</b>	<p>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.</p> <p>Chacun de ces trois types de contenus doit être placé entre les codes BEGIN et END, comme dans les exemples suivants :</p> <ul style="list-style-type: none"> <li>• <code> \${BEGIN_HTML} &lt;your HTML content&gt; \${END_HTML}</code></li> <li>• <code> \${BEGIN_CSS} &lt;your Interaction Point-specific style-sheet information&gt; \${END_CSS}</code></li> <li>• <code> \${BEGIN_JAVASCRIPT} &lt;your Interaction Point-specific JavaScript code&gt; \${END_JAVASCRIPT}</code></li> </ul> <p>Vous pouvez également entrer un certain nombre de codes spéciaux prédéfinis qui sont remplacés automatiquement lorsque la page est chargée, notamment :</p> <ul style="list-style-type: none"> <li>• <code> \${logAsAccept} </code> : 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.</li> <li>• <code> \${offer.AbsoluteLandingPageURL}</code></li> <li>• <code> \${offer.OFFER_CODE}</code></li> <li>• <code> \${offer.TREATMENT_CODE}</code></li> <li>• <code> \${offer.TextVersion}</code></li> <li>• <code> \$offer.AbsoluteBannerURL}</code></li> </ul>	Aucun équivalent, car le fragment HTML réside dans son propre fichier distinct de jsconnector.xml.

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
	<p>Chacun des codes d'offre listés représentent les attributs d'offre définis dans le modèle d'offre Unica Campaign qui a été utilisé par le vendeur pour créer des offres renvoyées par Unica Interact.</p> <p>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. Voir <a href="http://freemarker.org/docs/index.html">http://freemarker.org/docs/index.html</a> pour plus d'informations.</p>	
<b>Echantillons de codes spéciaux</b>	Contient des échantillons 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 changements apportés à cette page sont sauvegardés automatiquement lorsque vous naviguez 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'arrivée générale, avec des événements spécifiques 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.

## Infos page

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 Unica Interact.

<b>Paramètre</b>	<b>Description</b>	<b>Para- mètre équi- valent dans js- connec- tor.xml</b>
<b>URL contains</b>	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 "hypothèque.htm", vous pouvez la faire correspondre avec votre page d'informations sur les prêts hypothécaires.	<i>&lt;urlPat- tern&gt;</i>
<b>Friendly name for this page or set of pages</b>	Nom significatif associé à votre propre référence et qui décrit le rôle de ce mappage de page, tel que "Page d'information sur les prêts hypothécaires".	<i>&lt;friendly- Name&gt;</i>
<b>Also return offers as JSON data for JavaS- cript use</b>	Liste déroulante indiquant si vous souhaitez que Web Connector inclue les données d'offre brutes au format JavaScript™ Object Notation ( <a href="http://www.json.org/">http://www.json.org/</a> ) à la fin du contenu de la page.	<i>&lt;enable- RawData- Return&gt;</i>



## Événements 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 Unica 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
Événements individuels	<p>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 Unica Interact, Sélectionnez un ou plusieurs événements devant se produire lorsque la page est chargée.</p> <p>La séquence des appels d'API Unica Interact est la suivante :</p> <ol style="list-style-type: none"> <li>1. <code>startSession</code></li> <li>2. <code>postEvent</code> pour chaque événement de chargement de page individuel (si vous avez défini les événements individuels dans Unica Interact)</li> <li>3. Pour chaque point d'interaction : <ul style="list-style-type: none"> <li>• <code>getOffers</code></li> <li>• <code>postEvent(ContactEvent)</code></li> </ul> </li> </ol>	<code>&lt;event&gt;</code>


## 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 Unica Interact.



**Remarque :** Les paramètres de configuration de cette section correspondent à la section `<pageMapping>` | `<page>` | `<interactionPoints>` du fichier `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.	<code>&lt;interaction-Point&gt;</code>
<b>HTML Element ID (Unica Interact définit inner-HTML)</b>	Nom de l'élément HTML qui doit recevoir le contenu de ce point d'interaction. Par exemple, si vous avez indiqué <code>&lt;div id="welcome-banner"&gt;</code> sur la page, vous devez entrer <code>welcomebanner</code> (la valeur de l'ID) dans cette zone.	<code>&lt;htmlElement-Id&gt;</code>
<b>HTML Display Type</b>	Liste déroulante qui vous permet de sélectionner le type d'affichage HTML (fragments HTML, ou fichiers .flt définis sur une page de	<code>&lt;htmlSnippet&gt;</code>

Paramètre	Description	Paramètre équivalent dans jsconnector.xml
	configuration de Web Connector précédente) à utiliser pour ce point d'interaction.	
<b>Maximum number of offers to present (if this is a carousel or flipbook)</b>	Nombre maximum d'offres que Web Connector doit extraire du serveur Unica 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.	<max-NumberOfOffers>
<b>Event to fire when the offer is presented</b>	Nom de l'événement de contact devant être publié pour ce point d'interaction.	<contactEvent>
<b>Event to fire when the offer is accepted</b>	Nom de l'événement d'acceptation devant être publié pour ce point d'interaction.	<acceptEvent>
<b>Event to fire when the offer is rejected</b>	<p>Nom de l'événement de refus devant être publié pour ce point d'interaction.</p> <p> <b>Remarque :</b> A ce stade, cette fonctionnalité n'est pas encore utilisée.</p>	<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`. Vous 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 changements apportés à la page Web d'administration sont stockés 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 telles qu'elles apparaissent dans le fichier `jsconnector.xml`.

**Tableau 40. Options de configuration de Web Connector**

<b>Groupe de paramètres</b>	<b>Paramètre</b>	<b>Description</b>
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, telle que <code>http://host:port/jsconnector/click-Thru</code> .
	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 Unica Interact.

**Tableau 40. Options de configuration de Web Connector (suite)**

<b>Groupe de paramètres</b>	<b>Paramètre</b>	<b>Description</b>
	<code>visitorIdCookie</code>	Nom du cookie contenant l'ID d'audience.
	<code>audienceLevel</code>	Niveau d'audience campagne destiné au visiteur entrant, utilisé dans l'appel d'API à l'environnement d'exécution Unica Interact.
	<code>audienceIdField</code>	<p>Nom de la zone <code>audienceId</code> utilisée dans l'appel d'API à l'environnement d'exécution Unica Interact runtime.</p> <p> <b>Remarque :</b> Remarque : les identificateurs de niveau d'audience multi-zone ne sont actuellement pas pris en charge.</p>
	<code>audienceIdFieldType</code>	Le type de données de la zone d'ID d'audience [ <code>numeric</code>   <code>string</code> ] utilisé dans l'appel API à l'exécution Unica Interact
	<code>audienceLevelCookie</code>	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 <code>audienceLevel</code> .

**Tableau 40. Options de configuration de Web Connector (suite)**

<b>Groupe de paramètres</b>	<b>Paramètre</b>	<b>Description</b>
	<code>relyOnExistingSession</code>	Utilisé dans l'appel API à l'exécution Unica Interact. En règle générale, ce paramètre est défini sur "true".
	<code>enableInteractAPIDebug</code>	Utilisé dans l'appel API à l'exécution Unica Interact pour activer la sortie de débogage dans les fichiers journal.
	<code>pageLoadEvents</code>	Événement qui sera publié une fois cette page particulière chargée. Indiquez un ou plusieurs événements dans cette balise, dans un format de type <code>&lt;event&gt;event1&lt;/event&gt;</code> .
	<code>interactionPointValues</code>	Tous les éléments de cette catégorie jouent le rôle de valeurs par défaut pour les valeurs manquantes dans les catégories IP spécifiques.
	<code>interactionPointValuescontactEvent</code>	Nom par défaut de l'événement de contact devant être publié pour ce point d'interaction particulier.
	<code>interactionPointValuesaccept-Event</code>	Nom par défaut de l'événement d'acceptation devant être publié pour ce point d'interaction particulier.

**Tableau 40. Options de configuration de Web Connector (suite)**

Groupe de paramètres	Paramètre	Description
	<code>interactionPointValuesreject-Event</code>	Nom par défaut de l'événement de refus devant être publié pour ce point d'interaction particulier. (Remarque : à ce stade, cette fonctionnalité n'est pas encore utilisée.)
	<code>interactionPointValueshtmlSnippet</code>	Nom par défaut du modèle HTML à proposer pour ce point d'interaction.
	<code>interactionPointValuesmaxNumberOfOffers</code>	Nombre maximum d'offres par défaut à extraire de Unica Interact pour ce point d'interaction.
	<code>interactionPointValueshtmlElementId</code>	Nom par défaut de l'élément HTML à proposer pour ce point d'interaction.
	<code>interactionPoints</code>	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 <code>interactionPointValues</code> .
	<code>interactionPointname</code>	Nom du point d'interaction (IP).
	<code>interactionPointcontactEvent</code>	Nom de l'événement de contact devant être publié pour ce point d'interaction particulier.



**Tableau 40. Options de configuration de Web Connector (suite)**

<b>Groupe de paramètres</b>	<b>Paramètre</b>	<b>Description</b>
	<code>interactionPointacceptEvent</code>	Nom de l'événement d'acceptation devant être publié pour ce point d'interaction particulier.
	<code>interactionPointrejectEvent</code>	Nom de l'événement de refus devant être publié pour ce point d'interaction particulier. (Notez que cette fonctionnalité n'est pas encore utilisée.)
	<code>interactionPointhtmlSnippet</code>	Nom du modèle HTML à proposer pour ce point d'interaction.
	<code>interactionPointmaxNumberOfOffers</code>	Nombre maximum d'offres par défaut à extraire de Unica Interact pour ce point d'interaction.
	<code>interactionPointhtmlElementId</code>	Nom de l'élément HTML devant recevoir le contenu de ce point d'interaction.
	<code>enableDebugMode</code>	Indicateur booléen (valeurs admises : <code>true</code> or <code>false</code> ) pour activer le mode de débogage spécial. Si vous le définissez sur <code>true</code> , 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

**Tableau 40. Options de configuration de Web Connector (suite)**

Groupe de paramètres	Paramètre	Description
		le fichier <code>authorizedDebug-Clients</code> pour générer l'alerte.
	<code>authorizedDebugClients</code>	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.
	<code>enableRawDataReturn</code>	Indicateur booléen (valeurs admises : <code>true</code> ou <code>false</code> ) qui détermine si le Web Connector joint l'offre des données brutes au format JSON à la fin du contenu.
	<code>enableNetInsightTagging</code>	Indicateur booléen (valeurs admises : <code>true</code> ou <code>false</code> ) qui détermine si le Web Connector joint une balise Digital Analytics for On Premises à la fin du contenu.
	<code>apiSequence</code>	Représente une mise en œuvre de l'interface <code>APISequence</code> , qui dicte la séquence des appels d'API effectués par le Web Connector lorsqu'un <code>pageTag</code> est appelé. Par défaut, la mise en œuvre utilise une séquence de <code>StartSession</code> , <code>pageLoadEvents</code> , <code>getOffers</code> et <code>logContact</code> , où les deux derniers sont

**Tableau 40. Options de configuration de Web Connector (suite)**

Groupe de paramètres	Paramètre	Description
		spécifiques à chaque point d'interaction.
	<code>clickThruApiSequence</code>	Représente une mise en œuvre de l'interface APISequence, qui dicte la séquence des appels d'API effectués par le Web Connector lorsqu'un <code>clickThru</code> est appelé. Par défaut, la mise en œuvre utilise une séquence de <code>StartSession</code> et <code>logAccept</code> .
	<code>netInsightTag</code>	Représente le code HTML et le modèle JavaScript™ utilisés pour intégrer un appel à la balise Digital Analytics for On Premises. En général, il n'est pas nécessaire de changer 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 41. Options de la page d'administration de Web Connector

#### Options disponibles sur la page d'administration de Web Connector

Option	Objectif
Reload Configuration	Cliquez sur le lien <b>Reload Configuration</b> pour recharger les changements de configuration enregistrés sur disque en mémoire. Cela est nécessaire lorsque vous avez apporté des changements 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 <b>View Config</b> . Lorsque vous entrez l'URL d'une page et cliquez sur <b>View Config</b> , 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 <b>Execute Page Tag</b> , Web Connector renvoie le résultat <code>pageTag</code> basé sur le modèle d'URL. Cela simule l'appel à une balise de page. La différence entre un appel au <code>pageTag</code> 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 `WebConnectorTestPageSA.html` est inclus avec Unica Interact Web Connector (dans le répertoire `<Unica Interact_Home/jsconnector/webapp/html>`) et illustre le nombre de fonctions de Web Connector qui seraient 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[
/* #####
This is a test page that contains the WebConnector pageTag. Because the
name of this file has TestPage embedded, the WebConnector will detect a
URL
pattern match to the url pattern "testpage" in the default version of the
jsconnector.xml - the configuration definition mapped to that "testpage"
URL pattern will apply here. That means there should this page the
corresponding html element ids that correspond to the IPs for this URL
pattern (ie. 'welcomebanner', 'crosssellcarousel', and
'textservicemessage')
##### */
/* #####
This section sets the cookies for sessionId and visitorId.
```

Note that in a real production website, this is done most likely by the login

component. For the sake of testing, it's done here... the name of the cookie

has to match what's configured in the 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");

/* #####
Now set up the html element IDs that correspond to the IPs
##### */
    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><!--
#####
####
this is what is pasted from the pageTag.txt file in the conf directory of
the WebConnector installation... the var unicaWebConnectorBaseURL needs to
be
```

```

tweaked to conform to your local WebConnector environment
#####
####
-->
<!-- BEGIN: Interact Web Connector Page Tag -->

<script language="javascript" type="text/javascript">
//
    var unicaWebConnectorBaseURL=
        "[CHANGE ME - http://host:port/&lt;jsconnector&gt;/pageTag]";
    var unicaURLData = "ok=Y";
    try {
        unicaURLData += "&amp;url=" + escape(location.href)
    } catch (err) {}
    try {
        unicaURLData += "&amp;title=" + escape(document.title)
    } catch (err) {}
    try {
        unicaURLData += "&amp;referrer=" + escape(document.referrer)
    } catch (err) {}
    try {
        unicaURLData += "&amp;cookie=" + escape(document.cookie)
    } catch (err) {}
    try {
        unicaURLData += "&amp;browser=" + escape(navigator.userAgent)
    } catch (err) {}
    try {
        unicaURLData += "&amp;screenSize=" +
            escape(screen.width + "x" + screen.height)
    } catch (err) {}
    try {
        if (affiliateSitesForUnicaTag) {
</pre>
</div>
```

```

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 + "&nbsp;</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: Interact Web Connector Page Tag -->
<!--
#####
###
end of pageTag paste
#####
###
-->
</body>
</html>

```

# Chapitre 18. Intégration d'Unica Interact et de Digital Recommendations

Unica Interact peut s'intégrer à IBM Digital Recommendations et fournir des recommandations de produit déterminées par Unica Interact. Mes deux produits peuvent fournir des recommandations de produit mais avec des méthodes différentes. Digital Recommendations utilise le comportement Web d'un visiteur (filtre collaboratif) pour créer des corrélations entre les visiteurs et les offres recommandées. Unica 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, Unica Interact a recours à un profil personnel pour définir les offres qui passeront un ID de catégorie à Digital Recommendations 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 Unica Interact avec Digital Recommendations

Cette section explique comment Unica Interact peut s'intégrer à IBM Digital Recommendations pour fournir des recommandations de produits régies par Unica Interact, notamment une description du processus, et les mécanismes par lesquels l'intégration a lieu.

Unica Interact s'intègre à IBM Digital Recommendations via une interface de programme d'application (API) REST (Representational state transfer), disponible dans l'installation

Digital Recommendations. En effectuant les appels REST API avec l'ID de catégorie approprié, Unica 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'échantillon fournie avec votre installation Unica Interact), la page appelle Unica Interact pour extraire une offre. Si l'on suppose que l'offre a été configurée dans Unica 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 Unica Interact est effectué, et transmet les informations requises pour générer une offre pour ce 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'arrivée à laquelle les clics amènent le client, et l'ID de catégorie à utiliser pour déterminer les produits à recommander.
4. L'ID de catégorie est alors utilisé pour appeler Digital Recommendations 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 Unica 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 Unica 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 Digital Recommendations 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 Unica 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 Digital Recommendations, et reçoit des recommandations sur les types de couverts recommandés.

## Prérequis d'intégration

Avant de pouvoir utiliser l'intégration Digital Recommendations - Unica 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 Unica Interact, documentée dans le Guide d'administration et dans l'aide en ligne.
- Vous savez utiliser l'API REST Digital Recommendations décrite dans la documentation de Digital Recommendations 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 Digital Recommendations 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 Unica Interact utilise JSP (même si JSP n'est pas obligatoire).
- Vous avez un compte Digital Recommendations valide et la liste des ID de catégorie que Unica 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 Digital Recommendations (une adresse URL de votre environnement Digital Recommendations).

Consultez le modèle d'application fourni avec votre installation Unica Interact à titre d'exemple, ou consultez l'exemple de code dans [Utilisation de l'exemple de projet d'intégration \(à la page 674\)](#) pour plus d'informations.

## Configuration d'une offre pour l'intégration Digital Recommendations

Pour que votre page Web puisse appeler Digital Analytics Digital Recommendations afin d'extraire un produit recommandé, vous devez d'abord configurer l'offre Unica Interact avec les informations nécessaires à passer à Digital Recommendations.

Pour configurer une offre à lier à Digital Recommendations, vérifiez que les conditions suivantes sont respectées :

- Vérifiez que votre serveur d'exécution Unica Interact est configuré et fonctionne correctement.
- Vérifiez que le serveur d'exécution peut établir une connexion avec le serveur Digital Recommendations, 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 Digital Recommendations, procédez comme suit.

### 1. Créez ou éditez une offre pour Unica Interact.

Pour savoir comment ou modifier des offres, voir le Unica Interact guide d'utilisation et la documentation de 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'arrivée de l'offre.
- Un ID de catégorie Digital Recommendations associé à cette offre.

Vous pouvez récupérer manuellement l'ID de catégorie depuis votre configuration Digital Recommendations. Unica Interact ne peut pas récupérer directement les valeurs d'ID de catégorie.

Dans l'application Web de démonstration fournie avec votre installation Unica 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 Digital Recommendations) 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 Unica 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 Digital Recommendations. Les tâches restantes qui permettront à Digital Recommendations de fournir des recommandations de produits à Unica 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 Unica Interact.
- `Interact_Home/lib/JSON4J_Apache.jar`, qui est obligatoire pour gérer les données renvoyées par l'appel à l'API REST Digital Recommendations, qui renvoie des données au format JSON.

Voir [Utilisation de l'exemple de projet d'intégration \(à la page 674\)](#) pour savoir comment servir les offres à vos clients.

## Utilisation de l'exemple de projet d'intégration

Chaque installation de l'exécution Unica Interact est fournie avec un exemple de projet qui explique le processus d'intégration de Digital Recommendations - Unica 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 Digital Recommendations 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 Unica 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 Unica Interact. Utilisez par conséquent le fichier fourni avec votre installation comme point de départ pour les exemples dont vous avez besoin.

```
<%@ 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.jsverhttp.*,
    org.apache.commons.json.JSONObject,
    org.apache.commons.json.JSONArray" %>

<%

/
```

```
*****
*****

* This sample jsp program demonstrates integration of Interact and Digital
Recommendations.

*

* When the URL for this jsp is accessed via a browser. the logic will call
Interact

* to fetch an Offer. Based on the categoryID associated to the offer, the
logic

* will call Digital Recommendations to fetch recommended products. The
offer and products

* will be displayed.

* To toggle the customerId in order to demonstrate different offers, one
can simply

* append cid=<id> to the URL of this JSP.

*

* Prerequisites to understand this demo:

* 1) familiarity of Interact and its java API
* 2) familiarity of IntelligentOffer and its RestAPI
* 3) some basic web background ( html, css, javascript) to mark up a web
page

* 4) Technology used to generate a web page (for this demo, we use JSP
executed on the server side)

*

*

* Steps to get this demo to work:

* 1) set up an Interact runtime environment that can serve up offers with
the following

* offer attributes:

* ImageURL : url that links to the image of the offer

* ClickThruURL : url that links to the landing page of the offer
```



```

* CategoryID : Digital Recommendations category id associated to the
offer
* NOTE: alternate names for the attributes may be used as long as the
references to those
* attributes in this jsp are modified to match.
* 2) Obtain a valid REST API URL to the Intelligent Offer environment
* 3) Embed this JSP within a Java web application
* 4) Make sure interact_client.jar is in the WEB-INF/lib directory
(communucation with Interact)
* 5) Make sure JSON4J_Apache.jar (from interact install) is in the
*   WEB-INF/lib directory (communication with IO)
* 6) set the environment specific properties in the next two sections
*****
*****/

/

*****
*****
*   *****CHANGE THESE SETTINGS TO REFLECT YOUR
ENV*****
* Set your Interact environment specific properties here...
*****
*****/

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;

/
*****
*****
*****CHANGE THESE SETTINGS TO REFLECT YOUR
ENV*****
* Set your Digital Recommendations environment specific properties here...
*****
*****/

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 Digital Recommendations to get products
JSONObject products=getProductsFromIntelligentOffer(ioURL, cID, zoneID,
categoryId,
    intelligentOfferErrorMsg);

%>

<html>
<head>
  <title>My Favorite Store</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
on(a,b)

{if(!g){o(a);g=true;p((-1*b*j)}}},gotoprev:function(a,b){if(!g){o(a);g=tru
e;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>Welcome To My Store</b> Mr/Mrs. <%=customerId %>
<br><br>
<% if(offer != null) { %>
<!-- Interact Offer HTML -->

<div onclick="location.href='<%=offerClickThru %>'"
class="unicaofferblock_container">
  <div class="unicabackgroundimage">
    <a href="<%=offerClickThru %>"></a>
    </div>
  </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("Product Page") != null &&
      rec.getString("Product Page").trim().length()>0) {
    %>

    <li>
      <a href="<%=rec.getString("Product Page") %>"
title="<%=rec.getString("Product Name") %>">
        " width="166"
height="148" border="0" />
        <%=rec.getString("Product Name") %>
      </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>
}
%>

```



```

</div>
<% } %>

</body>
</html>

<%!
/
*****
*****
* The following are convenience functions that will fetch from Interact
and
*   Digital Recommendations
*****
*****/

/
*****
*****
* Call Digital Recommendations to retrieve recommended products
*****
*****/

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;

}

/
*****
*****
* Call Interact to retrieve offer
*****
*****/

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());
}
%>
```

# Chapitre 19. Intégration d'Unica Interact et de Digital Data Exchange

Avec Digital Data Exchange, votre site Web peut se connecter à Unica Interact pour composer un moteur d'exécution omnicanal puissant, capable de fournir les meilleures offres aux canaux les plus appropriés et d'évoluer (d'apprendre) en fonction du retour des offres pour améliorer en permanence l'efficacité du marketing.

Vous pouvez utiliser cet outil si votre équipe marketing utilise Unica Interact pour la gestion des offres omnicanaux et souhaite étendre ces offres intelligentes personnalisées à vos sites Web.

IBM Digital Data Exchange intègre les solutions marketing et tierces aux résultats numériques des études de comportement des consommateurs, à l'aide d'une API de syndication des données en temps réel et d'une solution de gestion des balises au niveau de l'entreprise.

Sans IBM Digital Data Exchange, les spécialistes du marketing dépendent des services informatiques pour connecter Unica Interact à leur site Web et appeler l'API Unica Interact depuis différentes pages Web. Avec IBM Digital Data Exchange, les spécialistes du marketing peuvent contourner IT et accéder directement à IBM Digital Data Exchange afin d'insérer des balises IBM Digital Data Exchange sur différentes pages Web.

## Prérequis

Avant d'utiliser l'intégration Unica Interact-Digital Data Exchange, vous devez vérifier que les conditions décrites dans cette section sont respectées.

Vérifiez que les prérequis suivants sont respectés.

- Vous savez utiliser l'API JavaScript Unica Interact, documentée dans le Guide d'administration et dans l'aide en ligne.
- Vous connaissez les balises Digital Data Exchange et les groupes de pages.
- Vous disposez d'un compte Digital Data Exchange valide.
- Votre fichier `interactapi.js` réside sur un hôte public, où il est accessible par les paramètres **Vendor**.

# Intégration d' Unica Interact à votre site Web à l'aide de IBM Digital Data Exchange

A l'aide de cette procédure, intégrez Unica Interact à votre site Web à l'aide de Digital Data Exchange.

1. Emplacement du fichier `Interactapi.js`.
  - a. Accédez à **Vendors > Vendor Settings** dans Digital Data Exchange.
  - b. Sélectionnez Unica Interact dans la liste déroulante **Vendor**.
  - c. Dans **Library Path**, entrez l'adresse URL à laquelle est hébergé le fichier `Interactapi.js`. L'URL ne doit pas contenir le protocole (http ou https).
  - d. Dans **Path To Public Rest Servlet**, entrez le chemin du servlet Rest.
2. Accédez à **Manage > Global Settings** dans Digital Data Exchange pour définir le nom de l'objet à utiliser comme identificateur de page dans **Unique Page Identifier**. Vous pouvez, par exemple, entrer `digitalData.pageInstanceID`.
3. Intégrez le fichier `eluminate.js` et un identificateur à la page Web dans laquelle Digital Data Exchange doit insérer des balises. Vous devez attribuer un identificateur unique à chaque page Web pour permettre à Digital Data Exchange de distinguer les différentes pages.

Par exemple, vous pouvez ajouter le script suivant à votre page d'accueil.

```
<!-- Setting Page Identifier -->
    <script>
        digitalData={pageInstanceID: "INTERACT_HomePage"};
    </script>

<!-- Including eluminate script -->
    <script type="text/javascript" src="http://libs.
        coremetrics.com/eluminate.js">
    </script>
    <script type="text/javascript">
        cmSetClientID("51310000|INTERACTTEST",false,"data.
```

```
coremetrics.com", document.domain);  
</script>
```

4. Dans Digital Data Exchange, créez des balises, des segments de code, des fonctions et d'autres éléments à ajouter à votre page Web.
5. Créez des groupes de pages pour définir ce qui doit figurer sur chaque page.

## Balises Unica Interact dans Digital Data Exchange

Utilisez les balises Digital Data Exchange par défaut pour définir des variantes correspondant aux pages Web contenant des données provenant de différents emplacements. Une fois définies, ces balises sont ajoutées à la liste des balises Unica Interact. Les balises ne contiennent pas forcément de zones à définir, ni de zones obligatoires, et peuvent être utilisées directement.

Les balises Unica Interact suivantes sont disponibles dans Digital Data Exchange, sous **Tags**.

- End Session
- Get Offers
- Load Library
- Afficher l'événement
- Set Audience
- Démarrage de session

Pour utiliser les balises Unica Interact, éditez-les pour définir les zones Tag Field, Method, Object Name, Data Type et Modifier pour chaque balise Unica Interact.

Les balises Post Event, Set Audience et Start Sessions acceptent les zones personnalisées. Cliquez sur l'icône Ajouter une zone de balise, puis sur l'icône Editer pour définir le paramètre personnalisé. Le processus est identique à celui d'autres définitions de paramètre, mis à part que le nom du paramètre est éditable et doit contenir le nom lui-même, une virgule et le type de données du paramètre. L'ordre des paramètres personnalisés dans la balise peut être modifié à l'aide des flèches de déplacement vertical.



Les balises peuvent également être associées à des fonctions JavaScript ou à des objets HTML pour être déclenchées après le déclenchement de la fonction ou lors d'un événement d'objet HTML.

## End Session

La balise End Session marque la fin d'une session Web.

Les zones suivantes sont disponibles pour la balise End Session.

**Tableau 42. Balises End Session**

### Balises End Session

Zone de la balise	Description
*Session ID	Identifie l'ID de session.
On Success Callback Function Name	Définit le nom de la fonction à appeler lorsque la méthode endSession aboutit.
On Failure Callback Function Name	Définit le nom de la fonction à appeler en cas d'échec de la méthode endSession.

Toutes les **zones de balise** signalées par \* sont obligatoires.

## Get Offers

Utilisez la balise Get Offers pour demander des offres au serveur d'exécution.

Les zones suivantes sont disponibles pour la balise Get Offers.

**Tableau 43. Balises Get Offers**

### Balises Get Offers

Zone de la balise	Description
*Session ID	Identifie l'ID de session.

**Tableau 43. Balises Get Offers****Balises Get Offers****(suite)**

<b>Zone de la balise</b>	<b>Description</b>
*Interact Point Name	Identifie le nom du point d'interaction référencé par cette méthode. Ce nom doit correspondre exactement au nom du point d'interaction défini dans le canal interactif.
*Number Requested	Identifie le nombre d'offres demandées.
On Success Callback Function Name	Définit le nom de la fonction à appeler lorsque la méthode getOffers aboutit.
On Failure Callback Function Name	Définit le nom de la fonction à appeler en cas d'échec de la méthode getOffers.

Toutes les **zones de balise** signalées par \* sont obligatoires.

La balise Get Offers doit être affectée à un groupe de pages dont le conteneur est défini sur Default.

## Load Library

La balise Load Library charge la bibliothèque JavaScript Unica Interact en tête de la page.

La balise Load Library n'a pas de paramètres. Elle utilise l'emplacement de la bibliothèque défini dans `Library Path`, dans **Vendor Settings**. Elle doit être insérée dans un groupe de pages à l'aide d'un conteneur de type `Head`, et s'exécuter sur chaque page contenant des balises Unica Interact.



**Important :** Aucune des autres balises ne fonctionne sans la balise Load Library. Sans elle, le fichier interact.js n'est pas chargé.

## Afficher l'événement

Utilisez la balise Post Event pour exécuter n'importe quel événement défini dans le canal interactif.

Les zones suivantes sont disponibles pour la balise Post Event.

**Tableau 44. Balises Post Event**

### Balises Post Event

Zone de la balise	Description
*Session ID	Identifie l'ID de session.
*Event Name	Identifie le nom de l'événement. Ce nom doit correspondre exactement au nom de l'événement défini dans le canal interactif. Ce nom est insensible à la casse.
On Success Callback Function Name	Définit le nom de la fonction à appeler lorsque la méthode postEvent aboutit.
On Failure Callback Function Name	Définit le nom de la fonction à appeler en cas d'échec de la méthode postEvent.

Toutes les **zones de balise** signalées par \* sont obligatoires.

Des paramètres facultatifs peuvent être ajoutés avec la fonction de zone de balise personnalisée. Les noms de balise personnalisés doivent être constitués du nom du paramètre, d'une virgule et du type de données.

## Set Audience

La balise Set Audience sert à définir l'ID et le niveau d'audience d'un visiteur.

Les zones suivantes sont disponibles pour la balise Set Audience.

## Tableau 45. Balises Set Audience

### Balises Set Audience

Zone de la balise	Description
*Session ID	Identifie l'ID de session.
*Audience ID	Identifie l'ID d'audience. Les noms doivent correspondre aux noms de colonne physique de toutes les tables contenant l'ID d'audience. L'ID d'audience ne peut pas contenir plus de 17 chiffres significatifs. Si un ID d'audience est composé de plus de 17 chiffres significatifs, il doit être partitionné ou changé en chaîne.
*Audience Level	Définit le niveau d'audience.
On Success Callback Function Name	Définit le nom de la fonction à appeler lorsque la méthode setAudience aboutit.
On Failure Callback Function Name	Définit le nom de la fonction à appeler en cas d'échec de la méthode setAudience.

Toutes les **zones de balise** signalées par \* sont obligatoires.

Des paramètres facultatifs peuvent être ajoutés avec la fonction de zone de balise personnalisée. Les noms de balise personnalisés doivent être constitués du nom du paramètre, d'une virgule et du type de données.

## Démarrage de session

La balise Start Session crée et définit une session Web.

Les zones suivantes sont disponibles pour la balise Start Session.

**Tableau 46. Balises Start Session****Balises Start Session**

<b>Zone de la balise</b>	<b>Description</b>
*Session ID	Identifie l'ID de session.
*Interact Channel	Définit le nom du canal interactif auquel cette session fait référence. Ce nom doit correspondre exactement au nom du canal interactif défini dans Campaign.
*Audience ID	Identifie l'ID d'audience. Les noms doivent correspondre aux noms de colonne physique de toutes les tables contenant l'ID d'audience.
*Audience Level	Définit le niveau d'audience.
*Rely on Existing Session	Définit si cette session utilise une session nouvelle ou existante.
*Debug	Active ou désactive les informations de débogage.
On Success Callback Function Name	Définit le nom de la fonction à appeler lorsque la méthode startSession aboutit.
On Failure Callback Function Name	Définit le nom de la fonction à appeler en cas d'échec de la méthode startSession.

Toutes les **zones de balise** signalées par \* sont obligatoires.

Des paramètres facultatifs peuvent être ajoutés avec la fonction de zone de balise personnalisée. Les noms de balise personnalisés doivent être constitués du nom du paramètre, d'une virgule et du type de données.

La balise Start Session doit être affectée à un groupe de pages dont le conteneur est défini sur `Default`.

## Exemple de paramétrage des balises

Cet exemple montre une configuration simple du paramétrage des balises Start Session, Post Event, Get Offers et End Session.

Quelle que soit la balise, vous pouvez extraire la valeur de la zone depuis le cookie à l'aide de la méthode cookie, ou depuis l'objet JavaScript à l'aide de la méthode javascriptobject.

Ces balises prennent en charge d'autres paramètres que cet exemple ne montre pas.

### Exemple de paramétrage de la balise Start Session

Cliquez sur **Balises > Balises IBM > Interact > Saisissez : Start Session** pour créer une balise Start Session. Editez la balise et définissez les paramètres suivants.

#### Paramètres de Session ID

- **Méthode** : `Constante`
- **Constante** : `5 555`
- **Type de données** : `Chaîne`
- **Modificateur** : `<null>`

#### Paramètres d'Interactive Channel

- **Méthode** : `Constante`
- **Constante** : `WSCDemo`
- **Type de données** : `Chaîne`
- **Modificateur** : `<null>`

#### Paramètres d'Audience ID

- **Méthode** : `Constante`
- **Constante** : `USERS_ID,2002,numeric`
- **Type de données** : `Chaîne`
- **Modificateur** : `<null>`

#### Paramètres d'Audience Level

- **Méthode** : Constante
- **Constante** : WSCUserId
- **Type de données** : Chaîne
- **Modificateur** : <null>

#### Paramètres de Rely On Existing Session

- **Méthode** : Constante
- **Constante** : Faux
- **Type de données** : Booléen
- **Modificateur** : <null>

#### Déboguer

- **Méthode** : Constante
- **Constante** : True
- **Type de données** : Booléen
- **Modificateur** : <null>

#### Paramètres d'On Success Callback Function Name

- **Méthode** : Non affecté
- **Valeur** : <null>

#### Paramètres d'On Failure Callback Function Name

- **Méthode** : Non affecté
- **Value** : <null>

### Exemple de paramétrage de la balise Get Offers

Cliquez sur **Balises > Balises IBM > Interact > Saisissez : Get Offers** pour créer une balise Get Offers. Editez la balise et définissez les paramètres suivants.

#### Paramètres de Session ID

- **Méthode** : Constante
- **Constante** : 5 555
- **Type de données** : Chaîne
- **Modificateur** : <null>

#### Paramètres d'Interact Point Name

- **Méthode** : Constante
- **Constante** : AuroraHomepageHeaderBannerLeft
- **Type de données** : Chaîne
- **Modificateur** : <null>

#### Paramètres de Number Requested

- **Méthode** : Constante
- **Constante** : 1
- **Type de données** : entier
- **Modificateur** : <null>

#### Paramètres d'On Success Callback Function Name

- **Méthode** : Constante
- **Constante** : onOfferReturnSuccess
- **Type de données** : string
- **Modificateur** : <null>

#### Paramètres d'On Failure Callback Function Name

- **Méthode** : Constante
- **Constante** : onOfferReturnError
- **Type de données** : string
- **Modificateur** : <null>



## Exemple de paramétrage de la balise Post Event

Cliquez sur **Balises > Balises IBM > Interact > Saisissez : Post Event** pour créer une balise Post Event. Editez la balise et définissez les paramètres suivants.

### Paramètres de Session ID

- **Méthode** : Constante
- **Constante** : 5 555
- **Type de données** : Chaîne
- **Modificateur** : <null>

### Paramètres d'Event Name

- **Méthode** : Constante
- **Constante** : ACCEPTOFFER
- **Type de données** : Chaîne
- **Modificateur** : <null>

### Paramètres d'On Success Callback Function Name

- **Méthode** : Constante
- **Constante** : onSuccessTestFunction
- **Type de données** : Chaîne
- **Modificateur** : <null>

### Paramètres d'On Failure Callback Function Name

- **Méthode** : Constante
- **Constante** : onErrorTestFunction
- **Type de données** : Chaîne
- **Modificateur** : <null>

### Paramétrage d'une zone supplémentaire

- **Zone de la balise** : UACIOfferTrackingCode:string
- **Méthode** : JavaScriptObject
- **Nom de l'objet** : oa.treatmentCode
- **Type de données** : Chaîne
- **Modificateur** : <null>

## Exemple de paramétrage de la balise End Session

Cliquez sur **Balises > Balises IBM > Interact > Saisissez : End Session** pour créer une balise End Session. Editez la balise et définissez les paramètres suivants.

Paramètres de Session ID

- **Méthode** : Constante
- **Constante** : 5 555
- **Type de données** : Chaîne
- **Modificateur** : <null>

Paramètres d'On Success Callback Function Name

- **Méthode** : Non affecté
- **Valeur** : <null>

Paramètres d'On Failure Callback Function Name

- **Méthode** : Non affecté
- **Valeur** : <null>

## Exemples de fonction

Lors de la création d'une balise, pour les fonctions utilisées pour les paramètres On Success Callback Function Name et On Failure Callback Function Name, il suffit de préciser le nom de la fonction si celle-ci est déjà présente sur la page Web.

Vous pouvez aussi utiliser les utilitaires Digital Data Exchange pour créer des fonctions et les ajouter à vos pages Web.

L'exemple suivant montre comment afficher une offre renvoyée par Unica Interact sur votre page Web. Vous devez intégrer ce script à la page ou utiliser le fragment de code Digital Data Exchange pour l'y injecter.

```
<script>
oa = {treatmentCode: ""};
function acceptOffer(treatmentCode) {
oa.treatmentCode = treatmentCode;
}
function onOfferReturnSuccess(response) {
var offer = response.offerList[0].offers[0];
var attributes = offer.attributes;
var offerText = "";
var offerLinkURL = "#";
for(var i = 0; i<attributes.length; i++)
{
if(attributes[i].n == "OfferTerms")
{
offerText = attributes[i].v;
}
else if(attributes[i].n == "OfferLinkURL")
{
offerLinkURL = attributes[i].v;
}
}

var link = "<a href=\"'+offerLinkURL+'\" onclick=\"acceptOffer('"+offer.treatmentCode+"')\">"+offerText+"</a>";
document.getElementById("offerContainer").innerHTML="
<div style=\"text-align:center;padding:
10px 0;background-color:#f5f5f5;\">"+link+"</div>";
}
function onOfferReturnError(response) {
```

```
(JSON.stringify(response));  
}  
</script>
```

## Vérifiez la configuration de votre intégration

Utilisez l'outil de test Digital Data Exchange et le fichier `Interact.log` pour traiter les problèmes de configuration.

Vous pouvez utiliser l'outil de test de Digital Data Exchange pour vérifier avec l'encyclopédie si la configuration fonctionne comme prévu. Pour ouvrir l'outil de test, cliquez sur **Deployment > Test Tool** dans Digital Data Exchange.

Le fichier `Interact.log` permet de visualiser de manière détaillée les appels de l'API Unica Interact. Pour les déboguer, ajoutez les fonctions de rappel On Success et On Failure aux différentes balises.

# Chapter 20. Intégration d'Unica Interact et d'Unica Journey

Unica Interact peut s'intégrer à Journey afin qu'une communication continue avec les utilisateurs puisse être établie en fonction des entrées d'Interact. Les segments Interact ou les informations d'audience peuvent être envoyés vers Journey, ce qui permet un dialogue continu avec le client. Dans l'application Interact, la nouvelle fonction est ajoutée pour publier les informations d'audience vers Journey. Cette option est activée via une configuration Journey au niveau du canal sortant des messages déclenchés en offrant la possibilité de mapper les zones d'Interact avec les zones Journey, ainsi qu'un nouveau canal Kafka sortant avec Journey.

Les sections suivantes décrivent le fonctionnement de cette intégration.

## Présentation

Interact peut utiliser les fonctionnalités Journey pour une communication continue avec les utilisateurs en cas de messages déclenchés. Les messages déclenchés permettent aux administrateurs de définir des modèles d'événement/d'événement ainsi que d'autres conditions après qu'une offre a été faite aux utilisateurs. Interact peut faciliter le mappage des champs de Journey avec les champs d'audience et les attributs d'offre. Interact Runtime peut déclencher un journey de communication avec les utilisateurs en transmettant les détails de l'audience au système Journey via le canal Kafka sortant sélectionné dans l'écran Messages déclenchés.

- Le mappage des champs Interact-Journey peut être défini sous l'onglet Passerelle dans le canal interactif, lorsqu'une passerelle de type Journey sortant est sélectionnée. Le mappage basé sur la configuration est supprimé et n'est plus pris en charge.
- Le canal sortant configuré par les utilisateurs peut être sélectionné dans la liste des canaux de messages déclenchés.
- Les détails de mappage des champs Interact-Journey pour tous les canaux sont transférés à Interact Runtime via le déploiement du canal interactif.

- Les détails de la connexion Kafka doivent être configurés via les paramètres de configuration de la passerelle dans l'environnement d'exécution Interact.
- Si l'événement ou le modèle d'événement correspond à l'API postEvent Interact et que d'autres conditions des messages déclenchés sont également remplies, le système déclenche le message sortant avec les champs d'audience conformément au mappage des champs Interact-Journey via le canal Journey-Kafka.

## Cartographie des champs Interact-Journey

Interact fournit un mécanisme pour mapper les champs Interact et Journey et identifier les informations à envoyer. Ces mappages de champs Journey peuvent être définis dans l'onglet Gateway d'un canal interactif, lorsque l'option Journey Outbound est sélectionnée. Ces mappages de champs Journey peuvent être définis dans l'onglet Gateway d'un canal interactif, lorsque l'option Journey Outbound est sélectionnée. Le mappage basé sur la configuration est supprimé et n'est plus pris en charge.

### Détails du trajet (Avant la version 12.1.0.3)

Une catégorie appelée « Journey » est disponible sous outboundChannels, qui définit les informations de mappage Interact Journey.

La catégorie de trajet a les paramètres suivants :

- Nouveau nom de catégorie: le nom de la configuration.
- Nom : Le nom du canal sortant qui apparaît dans la liste des messages déclenchés.
- EntrySourceCode : Il s'agit d'un champ obligatoire spécifique au parcours qui identifie la source des données entrantes et déclenche un ou plusieurs parcours. EntrySourceCode fait partie du message sortant d'Interact à Journey.
- DataDefinition : nom de la définition de données pour ce mappage. Ce champ est uniquement à des fins d'information dans Interact afin que les utilisateurs puissent identifier les fieldMappings. Le champ n'est pas utilisé dans Interact pour le traitement des messages.

## Cartographie des champs

Une fois les détails de base du Journey enregistrés, une nouvelle catégorie nommée "FieldMapping" s'affiche sous Journey.

La catégorie FieldMapping réplique la définition des données dans Journey. La catégorie permet de définir un champ Journey similaire au champ de définition des données dans Journey.

**Nouveau nom de catégorie** : Le nom du champ Interact qui est mappé avec ce champ Journey. Le format du nom du champ Interact est "Prefix.fieldname" et il est insensible à la casse pour Interact. Les noms de champ du côté Journey sont sensibles à la casse, assurez-vous donc que la casse correspondante est utilisée pour le mappage de champ. Le préfixe peut avoir deux valeurs possibles.

- **OFFRE** – Si le champ Voyage est mappé avec un attribut Offre, le nom du champ/nom de l'attribut doit être préfixé par « OFFRE ».
- **PROFIL** – Si le champ Journey est mappé sur un attribut de profil, le nom du champ Interact doit être préfixé par « PROFIL ».



**Note:** Les attributs en temps réel et les paramètres de session doivent également être considérés comme des attributs de profil et doivent être précédés de "PROFIL"

Les champs/paramètres/attributs Interact qui ne sont pas spécifiques à l'offre doivent être préfixés par "PROFIL".

**FieldName** : le nom du champ Journey auquel la valeur du champ Interact mentionné ci-dessus est mappée et envoyée à Journey via un message sortant. Le nom du champ Journey mentionné ici fait partie des messages sortants.

**Type de données** : il existe trois types de données parmi lesquels sélectionner, similaires à la définition des données de Journey.

- Chaîne
- Numérique
- Datetime

Le type de données est utilisé pour valider et formater la valeur du champ Interact avant de l'envoyer à Journey.

**DefaultValue** : Si la valeur du champ Interact n'est pas disponible, le système affecte la valeur par défaut au champ Journey.

**Obligatoire** : l'exécution d'Interact valide si le champ Journey est défini comme obligatoire. Si le champ est défini comme obligatoire et que la valeur du champ Interact correspondant n'est pas disponible, une erreur est consignée et le message est ignoré.



**Note**: Si defaultValue est défini pour ce champ, le système utilise la valeur par défaut plutôt que de rejeter le message sortant.

**DatetimeFormat** : cette propriété s'applique uniquement au type de données datetime. Il s'agit du format datetime, qui formate la valeur du champ Interact.

**MaxLength** : cette propriété s'applique uniquement au type de données chaîne. Cela fournit une validation supplémentaire à la valeur du champ. Si MaxLength est défini et que la valeur du champ Interact correspondant dépasse MaxLength, une erreur est consignée et le message est ignoré.

Les utilisateurs doivent configurer un mappage de champ pour chaque champ Journey, qui fait partie du message sortant vers Journey.

Vous pouvez définir les mappages de champs Interact-Journey à l'aide des options de Journey sortant disponibles sous l'onglet Passerelle dans le canal interactif.

- Il permet aux spécialistes du marketing de sélectionner la définition de données existante de Journey.
- Il permet aux utilisateurs de sélectionner les sources d'entrée qui existent dans les Journeys de type Unica Interact.
- Une fois la source d'entrée et la définition des données sélectionnées, en cliquant sur le bouton Récupérer, les détails de la définition des données sélectionnées renseignés sous les champs Journey s'affichent. Cela fournit des détails sur les champs de Journey tels que le nom, le type de données, le champ obligatoire/significatif sur les Journeys.



- Pour le champ Journey, les utilisateurs peuvent mapper les champs Interact. Il peut s'agir du champ Profil, du RTA ou de l'attribut Offre d'Interact.
- Si le champ Journey est obligatoire, la valeur par défaut est requise. Si la valeur ne peut pas être récupérée à partir des champs Interact côté Runtime, la valeur par défaut est utilisée et envoyée à Journey.

## Messages déclenchés

L'écran Messages déclenchés répertorie les canaux sortants disponibles à sélectionner. Les canaux sortants Journey sont également visibles dans la liste des canaux de messages déclenchés.

# Configurations d'exécution d'Interact

## Passerelle

Toutes les configurations propres à Journey pour les détails de connexion Kafka sont configurées sous `Affinium|interact|triggeredMessage|gateways`. Un nouveau modèle appelé "Journey" est disponible sous passerelles, qui doit être utilisé si les informations de canal sortant doivent être envoyées au système Journey. Les configurations obligatoires suivantes sont requises pour que la passerelle sortante Journey fonctionne.

- Détails de connexion Kafka : Les détails de connexion Kafka doivent être configurés en tant que paramètres vers la passerelle Journey. Les paramètres doivent inclure tous les détails requis tels que l'URL de courtier Kafka, la rubrique Kafka, les paramètres d'authentification, etc. pour permettre à l'environnement d'exécution Interact d'envoyer des messages sortants vers Journey. Pour trouver les paramètres Kafka dont la configuration est requise, voir la section [Propriétés de configuration de l'environnement d'exécution d'Unica Interact \(on page 411\)](#) qui contient des détails sur la création de Kafka Producer. Voici les exemples de paramètres que vous devez définir dans le cas où 'Authentication = none'.
  - providerURL : Kafka\_server\_ip:port
  - topic: Kafka\_topicname
  - Authentication : mode d'authentification Kafka.

- `validationTimeoutMillis` : Utilisé au moment de la validation du message sortant. Vous pouvez fixer toute valeur appropriée en millisecondes selon vos besoins.
- `deliveryTimeoutMillis` : Utilisé au moment de l'envoi du message. Vous pouvez fixer toute valeur appropriée en millisecondes selon vos besoins.

## Deployment

La configuration de Journey pour les informations de mappage de zones Interact-Journey fait partie du package de déploiement de canal interactif et est transférée vers Interact Runtime lorsque le "canal interactif" est déployé.

# Chapter 21. Intégration Unica Interact et Unica Deliver

Unica Deliver est une solution de messagerie marketing basée sur le web à l'échelle de l'entreprise que vous pouvez utiliser pour effectuer des messages de masse sortants tels que des e-mails, SMS, Push et WhatsApp, ainsi que des campagnes de messagerie transactionnelle.

Interact possède de riches capacités pour proposer les meilleures offres possibles en temps réel aux utilisateurs. Unica Interact peut s'intégrer à Deliver afin qu'une communication continue puisse être établie avec les utilisateurs sur la base des entrées d'Interact. Les informations sur l'audience d'Interact peuvent être poussées vers Deliver, permettant ainsi un dialogue continu avec le client. L'intégration Interact-Deliver vise à tirer parti des capacités sortantes de Deliver pour configurer et contacter un utilisateur avec les offres appropriées en fonction de ses événements et activités.

L'intégration Deliver-Interact est activée via l'onglet Passerelle sous l'interface utilisateur du canal interactif. Cela permet de sélectionner le type de communication Deliver avec les modèles Deliver pertinents et de mapper les champs Interact avec les champs Deliver.

## Présentation

Interact peut utiliser les capacités de Deliver pour communiquer en continu avec les utilisateurs via des messages déclenchés. Les messages déclenchés permettent aux administrateurs de définir des événements ou des modèles d'événements ainsi que d'autres conditions pour qu'une offre soit faite aux utilisateurs.

Interact permet aux administrateurs de sélectionner le type de message et le modèle de Deliver requis sous l'onglet Source. Actuellement, les messages par e-mail, SMS, MobileApp et WhatsApp sont pris en charge. Les champs Deliver présents dans le modèle peuvent être mappés aux champs d'audience Interact et aux attributs Offre.

L'environnement d'exécution Interact peut déclencher un message de communication pour les utilisateurs avec les valeurs de champ personnalisées vers le système de Deliver via le canal sortant sélectionné dans l'écran Messages déclenchés.

- Le mappage de champ Interact-Deliver peut être défini sous l'onglet passerelle dans le canal interactif, lorsqu'une passerelle de type Deliver Outbound est sélectionnée.
- Les champs de Deliver présents pour le mappage sont basés sur le type de source et le document de Deliver associé sélectionné. Les utilisateurs peuvent créer le modèle à partir des onglets Deliver Messaging Editor et Quick Builder.
- Le canal sortant configuré par les utilisateurs peut être sélectionné dans la liste des canaux de messages déclenchés.
- Les détails de mappage des champs Interact-Deliver pour tous les canaux sont transférés à Interact Runtime via le déploiement du canal interactif.
- Les détails de connexion de l'API Deliver TMS doivent être configurés via le modèle Deliver dans l'environnement d'exécution Interact.

Si l'événement ou le modèle d'événement correspond à l'API postEvent Interact et que d'autres conditions des messages déclenchés sont également remplies, le système déclenche le message sortant avec les champs d'audience personnalisés et les attributs d'offre conformément au mappage des champs Interact-Deliver via le canal de livraison.

## Cartographie Interact-Deliver

Interact fournit un mécanisme pour mapper les champs Interact et Deliver et identifier les informations à envoyer. Ces mappages de champs de Deliver peuvent être définis dans l'onglet Passerelle d'un canal interactif, lorsque l'option de Deliver sortante est sélectionnée. Les champs de Deliver à mapper sont récupérés en fonction du type de source et du modèle sélectionnés.

### **Fournir une cartographie des champs**

Une fois les détails de base de la passerelle enregistrés dans l'onglet "Général", les administrateurs peuvent passer à l'onglet "Source".

Ici, les administrateurs peuvent sélectionner le type de source pour le type de message par lequel les utilisateurs sont contactés. Actuellement, les e-mails, SMS, MobileApp et WhatsApp sont pris en charge.

Une fois le type de source sélectionné, les utilisateurs peuvent sélectionner dans la liste des modèles disponibles. Ces modèles sont créés à partir de « l'éditeur de messagerie » ou de Quick Builder (modèles d'e-mail) dans Deliver. La sélection d'un modèle de Deliver particulier répertorie tous les champs associés au modèle.

## Messages déclenchés

L'écran Messages déclenchés répertorie les canaux sortants disponibles à sélectionner. Les canaux sortants Deliver créés dans la configuration sont également visibles dans la liste des canaux de messages déclenchés.

# Configurations d'exécution d'Interact

## Passerelle

Toutes les configurations propres à Deliver pour les détails de l'API TMS Deliver sont configurées sous `Affinium|interact|triggeredMessage|gateways`. Un nouveau modèle appelé « Journey » est disponible sous l'onglet Passerelle, qui doit être utilisé si les informations de canal sortant doivent être envoyées au système. Les paramètres obligatoires requis dans le modèle Deliver sont les suivants :

- `deliverURL` : URL de l'API transactionnelle Deliver.
- `username` : Nom d'utilisateur du compte hébergé Deliver.
- `dataSourceName` : Source de données du compte hébergé Deliver.

## Deployment

La configuration de Deliver pour les informations de mappage de champ Interact-Deliver fait partie du package de déploiement de canal interactif et est transférée vers Interact Runtime lorsque le « canal interactif » est déployé.

# Chapter 22. Configurer les passerelles

Utilisez des passerelles de message déclenché pour envoyer des informations sur les offres à partir des canaux de communications sortantes.

Vous pouvez utiliser les passerelles sortantes suivantes avec des messages déclenchés. Ces passerelles sont disponibles sous "Affinium|interact|triggeredMessage|passerelles".

Vous pouvez recevoir des informations dans Interact en utilisant les passerelles entrantes, qui sont disponibles sous "Affinium|interact|activityOrchestrator|". Les fichiers de passerelle sont disponibles sous `INTERACT_HOME/conf/gateways/`. Par ailleurs les configurations liées sont prêtes à l'emploi sous le nœud "Affinium|Campaign|partitions|partition1|Interact|outboundChannels" pour la phase de conception et sous les nœuds "Affinium|interact|triggeredMessage|passerelles" , "Affinium|interact|activityOrchestrator|passerelles".

- Unica Interact Passerelle entrante pour IBM Universal Behavior Exchange
- Unica Interact Passerelle sortante pour IBM Universal Behavior Exchange
- Unica Interact Passerelle sortante d'e-mail (Transact) pour IBM Marketing Cloud
- Unica Interact Passerelle sortante pour IBM Mobile Push Notification

Le mappage de ces passerelles peut être défini sous l'onglet Passerelle sous canal interactif. Si vous souhaitez continuer à utiliser le mappage basé sur les fichiers de propriétés, vous devez définir les paramètres JVM suivants.

Pour les passerelles sortantes

`OUTBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES`. Par exemple :

`DOUTBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES=EMail`

Pour les passerelles entrantes

`INBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES`. Par exemple :

`DOUTBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES=UBX`

S'il existe plusieurs passerelles, ajoutez les noms de passerelle séparés par des virgules.

# Utilisation de Unica Interact Inbound Gateway for IBM Universal Behavior Exchange

Afin de pouvoir utiliser la passerelle entrante Unica Interact pour IBM Universal Behavior Exchange, vous devez créer un nœud final et un événement dans UBX, configurer Interact, et configurer un nœud final d'abonnés UBX.

Utilisez les configurations suivantes comme exemple pour votre configuration.

## A. Création d'un nœud final et d'un événement dans UBX

Il s'agit d'un modèle de nœud final et d'événement que vous pouvez utiliser à titre d'exemple.

Pour créer un nœud final et un événement dans UBX, procédez comme suit.

1. Utilisez le client d'API REST pour publier les requêtes dans UBX.
2. Enregistrez un nœud final dans UBX avec JSON. Voir l'exemple suivant.

```
Appel de méthode : PUT URL : https://ubx-qa1-api.adm01.com/v1/endpoint
En-têtes : Content-Type : application/json Accept-Charset :
UTF-8 Autorisation : Bearer 912586bf-190d-48f9-8488- 26f1bf532ef3
(Remarque : il s'agit de la clé d'authentification générée à partir
de l'interface utilisateur UBX.) Body { "name":"Interactubxdk1",
"description":"Interactubxdk1", "providerName":"IBM",
" url":"http : //Host:port/ubxEndPoint/UBXEndPoint",
"endpointTypes":{ "event":{ "source":{ "enabled":true }},
"destination":{ "enabled":true,
"url":"http ://Host:port/ubxEndPoint/UBXEndPoint",
"destinationType":"push" } } },
"marketingDatabasesDefinition":{ "marketingDatabases":[ { "name":"IDS
ync", "identifiers":[ { "name ":"interactprofileid",
"type":"INTERACTID" } ] } ] } }
```

3. Enregistrez un type d'événement dans UBX avec JSON. Voir l'exemple suivant.

```

Enregistrement d'événement pour l'événement Interact dans l'appel de
méthode UBX : URL POST : https://ubx-qal-api.adm01.com/v1/eventtype
En-têtes : Content-Type : application/json Accept-Charset :
UTF-8 Autorisation : Bearer 912586bf -190d-48f9-8488-26f1bf532ef3
Remarque : il s'agit de la clé d'authentification
générée à partir de l'interface utilisateur UBX.) Bearer
912586bf-190d-48f9-8488-26f1bf532ef3 Body { "name":
"recommendedOffers", "description": OMO, "code": "offres
recommandées" }

```

#### 4. Publiez un événement dans UBX avec JSON. Voir l'exemple suivant.

```

{ "channel" : "mobile", "identifiers" : [ { "name" :
"interactprofileid", "value" : "55" } ], "events" : [ { "code" :
"recommendedOffers", "timestamp" : "2015-12-28T20:16:12Z" } ] }

```

## B. Configuration de Unica Interact pour la passerelle entrante Unica Interact pour IBM Universal Behavior Exchange

Pour configurer Unica Interact, procédez comme suit.

1. Dans l'interaction | activitéOrcestrateur | propriété de configuration des **récepteurs**, ajoutez un nouveau récepteur. Définissez **Type** sur `Kafka` ou `Personnalisé`. Si vous choisissez `Custom`, entrez **ClassName** et **ClassPath**. Si vous choisissez `Kafka`, laissez **ClassPath** et **ClassName** vides.
2. Ajoutez les paramètres `providerUrl`, `topic`, `authentication`, `group.id` et `zookeeper.connect` pour votre récepteur.
3. Dans propriété de configuration des **Interact** | **activityOrcestrateur** | **passerelles**, la catégorie UBX est disponible par défaut.
4.
  - Utilisation de mappages basés sur des fichiers de propriétés.
    - Créez un nouveau dossier, par exemple, le dossier `Interactubx12` sous le répertoire `<Interact_Home>\conf\inbound\UBX` et copiez les fichiers de propriétés dans ce nouveau dossier. Le nom du dossier doit correspondre au nom du nœud final d'abonnés que vous avez créé dans UBX.



- Configurez le fichier `interactEventNameMapping.properties`

Utilisez ce fichier pour mapper la valeur de la zone d'événement de contenu définie dans le fichier `interactEventPayloadMapping.properties` en tant que `[EventName]` au nom d'événement Interact. Le fichier `interactEventNameMapping.properties` se trouve dans le répertoire `<Install dir>\conf\inbound\UBX. {Nom de l'événement UBX}={Nom de l'événement Interact}` Exemple : `RecommendedOffers=recommendedOffers`

Si la prise en charge des données utiles d'une source spécifique est nécessaire, ce fichier peut également être placé dans le répertoire `<Install dir>\conf\inbound\UBX\{source}`. La valeur de la source doit correspondre à la valeur de la zone de source dans la charge d'événement Universal Behavior Exchange, généralement le nom du nœud final Universal Behavior Exchange. Si la prise en charge des données à l'aide de versions spécifiques est nécessaire, ce fichier peut également être placé dans le répertoire `<Install dir>\conf\inbound\UBX\{source}\version-{version}`. La valeur de la version doit correspondre à la valeur du champ de version dans la charge d'événement Universal Behavior Exchange.

Pour prendre en charge plusieurs données d'instance Universal Behavior Exchange, ce fichier peut également être placé dans le répertoire `<Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID}`. La valeur de `clientID` doit correspondre à la valeur de `clientID` dans la charge d'événement Universal Behavior Exchange.

- Configurez le fichier `interactEventPayloadMapping.properties` et ajoutez vos définitions de champ.

Utilisez le fichier `interactEventPayloadMapping.properties` pour mapper la zone entrante aux paramètres d'API Interact. Le fichier `interactEventPayloadMapping.properties` se trouve dans le répertoire `<Install dir>\conf\inbound\UBX.`

Paramètres de l'API Interact: la valeur doit commencer par une définition de type de champ, suivie soit d'une valeur statique lorsque la valeur est entre guillemets doubles, soit d'un nom de champ provenant des données de charge utile. (FIELD\_TYPE)"STATIC\_VALUE" ou (FIELD\_TYPE)PAYLOAD\_FIELD\_NAME. FIELD\_TYPE peut être une chaîne, un nombre ou une date/heure.

Par exemple :

```
[SessionID]=(String)interactprofileid
[EventName]=(String)code
[AudienceIDFieldNames]=(String)"change_me" [AudienceIDField
Values]=(String)interactprofileid
[AudienceLevel]=(String)"change_me" [InteractChannel]=(String)
)"change moi"
```

Données d'événement : ces propriétés sont utilisées pour mapper les attributs d'événement qui peuvent être utilisés dans vos communications de canal sortant. Le côté gauche contient les noms de variable que vous utilisez dans votre communication de canal sortant. La valeur doit commencer par une définition de type de champ, suivie soit d'une valeur statique lorsque la valeur est entre guillemets doubles, soit d'un nom de champ provenant des données utiles. (FIELD\_TYPE)"STATIC\_VALUE" ou (FIELD\_TYPE)PAYLOAD\_FIELD\_NAME. FIELD\_TYPE peut être une chaîne, un nombre ou une date/heure.

Si la prise en charge des données utiles d'une source spécifique est nécessaire, ce fichier peut également être placé dans le répertoire `<Install dir>\conf\inbound\UBX\{source}`. La valeur de la source doit correspondre à la valeur de la zone de source dans la charge d'événement Universal Behavior Exchange, généralement le nom du nœud final Universal Behavior Exchange. Si la prise en charge des données utilisant des versions spécifiques est requise, ce fichier peut également être placé dans le répertoire `<Install dir>\conf`

`\inbound\UBX\{source}\version-{version}`. La valeur de la version doit correspondre à la valeur du champ de version dans la charge d'événement Universal Behavior Exchange. Pour prendre en charge plusieurs données d'instance Universal Behavior Exchange, ce fichier peut également être placé dans le répertoire `<Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID}`. La valeur de clientID doit correspondre à la valeur de clientID dans la charge d'événement Universal Behavior Exchange.

- Redémarrez le serveur Interact.
- Créez un canal interactif et ajoutez-lui un événement.
- Utilisation de mappages basés sur l'interface utilisateur :

Utilisez l'option **Generic Inbound** disponible sous l'onglet **Gateway** sur le canal interactif. Créez une passerelle avec le nom 'UBX' et définissez les mappages d'événements.

Il est nécessaire que la requête entrante provenant d'UBX contienne le paramètre d'en-tête ICName. Les mappages du canal interactif spécifique seront initialisés en fonction de cette valeur d'en-tête.

5. Effectuez les étapes mentionnées dans les sections C et D ci-dessous.
6. Déployez le canal interactif.
7. Publiez un événement dans UBX depuis n'importe quel éditeur ou, pour essayer, avec un client d'API REST.

Exemple de corps d'événement :

```
{ "channel" : "mobile", "identifiers" : [ { "name" :
  "interactprofileid", "value" : "55" } ], "events" : [ { "code" :
  "recommendedOffers", "timestamp" : "2015-12-28T20:16:12Z" } ] }
```

8. Consultez le journal Unica Interact pour savoir si l'événement de message de déclenchement est déclenché. Vous pouvez vous servir de cet événement pour traiter les schémas et les messages déclenchés en fonction de votre cas d'utilisation spécifique.

Vous pouvez aussi vous reporter aux journaux du nœud final pour en savoir plus.

## C. Configuration duUnica Interact Passerelle entrante pour le point de terminaison IBM Universal Behavior Exchange

Vous devez également suivre les instructions pour procéder aux configurations suivantes.

- Nœud final UBX avec Kafka
- Fichier ubxInboundEndpoint.properties de noeud final
- Fichier inboundTopicNameConfig.properties de nœud final
- Fichier de nœud final log4j2.xml

### Fichier ubxInboundEndpoint.properties

Utilisez le fichier ubxInboundEndpoint.properties pour configurer l'emplacement dans lequel envoyer la charge d'événement Universal Behavior Exchange à la passerelle entrante Unica Interact pour IBM Universal Behavior Exchange. Le fichier ubxInboundEndpoint.properties se trouve dans le répertoire `<répertoire d'installation du point de terminaison de la passerelle sur le serveur d'applications>`.

- **providerURL** Obligatoire : une liste de paires d'hôtes ou de ports à utiliser pour établir la connexion initiale avec le cluster Kafka. Cette liste doit avoir la forme de hôte1:port1.
- **authentication** par défaut est none – Pour les différents modes d'authentification et leurs propriétés associées, voir Interact | orchestrateur d'activité | rubrique **récepteurs**.
- **group.id** Obligatoire - N'importe quelle chaîne (exemple: InteractTMGateway)
- **zookeeper.connect** Facultatif - `<host>:<port>` (exemple: localhost:2181)

Le redémarrage de l'application Web du nœud final de la passerelle (ubxInboundEndpoint.war) est requis sur le serveur Web ou le serveur d'applications

- Nœud final UBX avec Kafka
- Fichier ubxInboundEndpoint.properties de noeud final
- Fichier inboundTopicNameConfig.properties de nœud final
- Fichier log4j.properties de noeud final

### **fichier inboundTopicNameConfig.properties**

Le nœud final de la passerelle entrante Unica Interact pour IBM Universal Behavior Exchange envoie l'événement à Interact en écrivant des données dans une rubrique Kafka. Spécifiez le nom de la rubrique dans laquelle les données sont publiées à l'aide du fichier inboundTopicNameConfig.properties.

Exemple :

```
topic=UBXTopic
```

Le redémarrage de l'application Web du nœud final de la passerelle (ubxInboundEndpoint.war) est requis sur le serveur Web ou le serveur d'application pour que les modifications effectuées dans ce fichier soient appliquées.

### **fichier log4j2.xml**

Configurez différents niveaux de journalisation pour le nœud final à l'aide du fichier log4j2.xml. Le fichier log4j2.xml se trouve dans le répertoire <répertoire d'installation du nœud final de la passerelle sur le serveur d'applications>.

### **La description**

Définissez le niveau de journalisation pour com.ibm.web.offerorchestration.inbound.common et com.ibm.web.offerorchestration.inbound.ubx en conséquence

## **D. Déploiement de la passerelle entrante Unica Interact pour IBM Universal Behavior Exchange et du nœud final**

- La guerre de passerelle entrante Unica Interact est présente à l'emplacement `<Interact_Home>\UBXInboundEndpoint\ubxInboundEndpoint.war`.
- Copiez ce fichier war, ainsi que le dossier de configuration sur les serveurs d'application Unica pris en charge. Ce serveur envoie des données à la rubrique Kafka entrante Interact, qui seront utilisées ultérieurement par la passerelle entrante Unica Interact pour IBM Universal Behavior Exchange.

Le Unica Interact passerelle entrante pour le point de terminaison IBM Universal Behavior Exchange est configurée pour accepter les demandes d'Universal Behavior Exchange et les envoyer au Unica Interact Passerelle entrante pour IBM Universal Behavior Exchange.

Vous devez effectuer les tâches ci-dessous afin de configurer le nœud final de la passerelle des abonnés Universal Behavior Exchange.

1. Configurez une nouvelle propriété système Java (`-DubxInboundEndpointConfigPath`) en modifiant le fichier de configuration sur le serveur web ou dans la console d'administration du serveur d'applications.

La propriété `-DubxInboundEndpointConfigPath` doit pointer vers le répertoire d'installation du point de terminaison sur le serveur (c'est-à-dire le dossier conf qui est copié dans la section ci-dessus).

Ce répertoire contient des fichiers de configuration pour l'environnement Kafka cible et divers niveaux de journalisation pour le nœud final. Par exemple -

```
DubxInboundEndpointConfigPath=c:\ubxInboundEndpoint.
```

2. Déployer le Unica Interact Inbound Gateway pour le fichier d'archive Web du point de terminaison IBM Universal Behavior Exchange (`ubxInboundEndpoint.war`) à partir du répertoire d'installation, comme décrit dans la documentation du serveur Web ou du serveur d'applications.

Pour vérifier que le nœud final a été installé correctement, entrez l'adresse suivante dans un navigateur et recherchez le message

Le point de terminaison UBX est UP.

```
http://[Server]:[Port]/[ContextRoot]/UBXEndPoint
```



**Note:** Vous devez protéger les informations accessibles au public Unica Interact Passerelle entrante pour le point de terminaison IBM Universal Behavior Exchange en ajoutant les règles de pare-feu requises pour accepter la demande HTTP d'IBM Universal Behavior Exchange Server uniquement.

Par exemple, vous pouvez utiliser les instructions suivantes pour configurer et déployer Unica Interact Passerelle entrante pour le point de terminaison IBM Universal Behavior Exchange sur WebSphere Application Server.

1. Ouvrez la console d'administration.
2. Sélectionnez **Serveurs > (Développez Types de Serveur) > server\_name > (Développez Java™ et Gestion) > Définition du Processus > Machine Virtuelle Java.**
3. Dans les arguments JVM génériques, ajoutez la propriété  
`-DubxInboundEndpointConfigPath=<Répertoire d'installation du point de terminaison de la passerelle d'abonné Exchange Universal Behavior sur le serveur d'applications>.` Par exemple, ajoutez la propriété  
`-DubxInboundEndpointConfigPath=C:\ubxInboundEndpoint.`
4. Cliquez sur **OK** pour enregistrer les modifications apportées à la configuration principale.
5. Redémarrez le serveur d'applications.

### Déployer le noeud final dans WebSphere Application Server

1. Connectez-vous à la console d'administration.
2. Accédez à **Applications > Types d'application > Applications d'entreprise Websphere.** Cliquez sur **Installer.**
3. Utilisez l'option **Préparation de l'installation de l'application** pour localiser le fichier war du point de terminaison (`ubxInboundEndpoint.war`) à installer, puis cliquez sur **Suivant.**
4. Cliquez sur **Suivant** dans les pages suivantes pour atteindre les **racines de contexte de carte pour les modules Web.**
5. Utilisez les **racines de contexte de carte pour les modules Web** pour localiser la racine de contexte et remplacez la valeur par `/UBXEndPoint`, cela devient la racine de contexte. Cliquez sur **suivant.**
6. Cliquez sur **Terminer.**

7. Une fois l'installation de l'application terminée, cliquez sur **Enregistrer** pour conserver les modifications sur la configuration principale.
8. De retour dans les applications répertoriées et installées, cochez la case pour `ubxInboundEndpoint_war` et cliquez sur **Démarrer** pour charger.



**Note:**

UBXInboundEndpoint est mis à jour pour prendre en charge Kafka par défaut.

UBXInboundEndpoint sera fourni prêt à l'emploi avec Interact.

Si vous utilisez déjà UBXInboundEndpoint avec la file d'attente JMS, vous pouvez continuer à utiliser le fichier war UBXInboundEndpoint existant.

## Utilisation de Unica Interact Outbound Gateway for IBM Universal Behavior Exchange

Afin de pouvoir utiliser Unica Interact Outbound Gateway for IBM Universal Behavior Exchange, vous devez configurer Unica Interact, UBX et la passerelle.

Utilisez les configurations suivantes comme exemple pour votre configuration.

Si vous utilisez UBX comme canal de communications sortantes, Unica Interact agit en tant que type de diffuseur de noeud final qui publie des événements dans UBX. A partir de UBX, ces événements peuvent être envoyés à l'abonné.

Avant de commencer la configuration, demandez un accès sortant à la machine hôte. Vous devez disposer d'un accès réseau sur la machine hôte.

### Enregistrement de noeuds finaux et d'événements dans UBX

Un point de terminaison approprié doit être enregistré auprès d'UBX comme prérequis. Contactez Acoustic pour créer des points de terminaison et des événements dans UBX.



## Configuration de Unica Interact et de la passerelle

1. Sous le dossier **Interact**, dans le fichier `httpConnectionConfig.properties`, spécifiez le délai d'expiration.

Par exemple:

```
connectTimeoutMs=180000
```

Lorsque OMO est configuré pour utiliser une connexion HTTP, un proxy HTTP peut être configuré en option avec une authentification entre Interact et le point de terminaison. Pour activer le proxy pour des passerelles sortantes, mettez à jour les valeurs des propriétés suivantes.

- `proxyHost`=< adresse IP du serveur proxy >
- `proxyPort`=< Port d'écoute du serveur proxy >
- `targetUsername`=< nom d'utilisateur pour se connecter au serveur proxy. laissez vide si aucune authentification requise >
- `targetPassword`=< mot de passe pour se connecter au serveur proxy. laissez vide si aucune authentification requise >

Par exemple :

```
authKey=912586bf-190d-48f9-8488-26f1bf532ef3 [Clé d'authentification
utilisée pour enregistrer le point de terminaison et l'événement de
l'éditeur dans UBX] interactProfileIdFieldName=interactprofileid [Nom
du champ du fichier ubxContentMapping.properties]
```

2. Utilisation de mappages basés sur des fichiers de propriétés

- Mettez à jour les valeurs de `interactprofileid` et `eventName` dans le fichier `ubxContentMapping.properties`.

Vous pouvez transmettre le nom de l'événement dans trois formats : lorsque la valeur est entre guillemets doubles, il s'agit d'une valeur statique ; lorsque la valeur est au format `offre.offerAttributeName`, elle correspond à l'attribut d'offre `offerAttributeName` ; et lorsque la valeur est au format `profile.profileAttributeName`, elle correspond à l'attribut de profil `profileAttributeName`. La valeur de nom d'événement doit correspondre au code

utilisé pour enregistrer l'événement dans UBX. Ce code est sensible à la casse. Redémarrez le serveur d'applications.

Par exemple :

```
eventName="abandoned_shopping_carts" eventName=offer.Card
eventName=profile.EMAIL
```

- Utilisation de mappages basés sur l'interface utilisateur.
    - Créez une passerelle avec le nom 'UBX' de type Generic Outbound dans un canal interactif.
    - Créez une propriété de canal avec le nom de la propriété en tant qu'interactProfileIdFieldName et le champ d'interaction en tant que nom de champ requis du point de terminaison. Par exemple : interactProfileIdFieldName=interactprofileid .
    - Dans la section Mapping, vous pouvez définir eventName dans les trois formats suivants:
      - eventName=abandoned\_shopping\_carts (la valeur par défaut peut être attribuée pour obtenir le eventName statique)
      - eventName=offer.Card (la valeur est extraite des attributs de l'offre)
      - eventName=profile.EMAIL (la valeur est extraite des attributs de profil)
3. Ajoutez une chaîne sous propriété de configuration **Interact | MessageDéclencheur | canal**.
  4. Définissez le même canal au moment de la conception sous **Campagne | cloisons | partition [n] | Interagir | canaux sortants**
  5. Créez une règle de message de déclenchement avec un nom d'événement qui utilise le canal que vous avez ajouté aux étapes précédentes.
  6. Déployez le canal interactif.
  7. A partir du client de test d'API, démarrez la session du canal interactif dans laquelle la règle de message déclenché est configurée et l'événement de publication qui déclenche l'offre dans UBX.

# Utilisation de Unica Interact Outbound Gateway for IBM Mobile Push Notification

Pour pouvoir utiliser cette passerelle sortante Mobile Push ou de diffuseurs, vous devez configurer Unica Interact, IBM Marketing Cloud et la passerelle.

Utilisez les configurations suivantes comme exemple pour votre configuration.

## Configuration d'IBM Marketing Cloud

1. Assurez-vous que vous disposez d'un compte IBM Marketing Cloud avec un accès push. Notez également votre ID client, votre secret client et votre jeton d'actualisation.
2. Dans l'onglet **Données**, créez une nouvelle base de données. Ajoutez un nouvel identifiant d'utilisateur mobile à la base de données avec les champs par défaut.
3. Dans l'onglet **Rechercher**, recherchez par le champ `ID utilisateur mobile`. Passez le curseur de la souris sur le premier champ `Pas d'e-mail`. Vous verrez l'ID du destinataire au bas de la fenêtre du navigateur. Ajoutez cet ID de destinataire au Unica Interact tableau de profil.

## Configuration de Unica Interact Outbound Gateway for IBM Mobile Push Notification

1. Configurez le fichier `silverpopEngagePushConfig.properties`.

Par exemple :

```
OauthServiceURL=<protocol>://<hostname>/<other_information>
pushServiceURL=<protocol>://<hostname>/<other_information>
```

2. Configurez le fichier `silverpopEngagePushContentMapping.properties`.



### Note:

Si vous utilisez des mappages basés sur l'interface utilisateur, effectuez les actions suivantes.



- a. Créez une passerelle avec le nom "MobilePush" de type Generic Outbound dans un canal interactif.
- b. Créez les mappages suivants dans la section Mappage.

Par exemple :

```
Attributs de la table de profil Interact : appKey=appKey
engageRecipientId=recipientId mobileUserId=mobileUserId
deviceType=deviceType Attributs de l'offre
Interact : simpleSubject=simpleSubjectAttr
simpleMessage=simpleMessageAttr simpleActionData=simpleActionDataAttr
simpleActionType=simpleActionTypeAttr
simpleActionLabel=simpleActionLabelAttr
personalizeAttributeList=personalizeAttributeList contentId=ContentID
campaignId=campa
```

## Configuration Unica Interact

1. Créez les attributs d'offre suivants.

```
simpleActionDataAttr: string simpleActionLabelAttr: String
simpleActionTypeAttr: string simpleMessageAttr: string
simpleSubjectAttr: string contentID: string campaignId=string
personalizeAttributeList=string
```

2. Créez un modèle d'offre avec les attributs d'offre et les valeurs d'offre suivants.

```
simpleActionDataAttr : www.ibm.com simpleActionLabelAttr : URL
ouverte simpleActionTypeAttr : url simpleMessageAttr : <Entrez le
texte de votre message ici> simpleSubjectAttr : <Entrez l'objet
ici> contentID : ID du modèle de message push créé dans Engage.
PersonalizeAttributeList : une liste séparée par des virgules de
paires de valeurs de nom d'attribut que vous souhaitez placer dans
```

la section `personalizationDefaults` de la charge utile à envoyer à Engage.

Lorsque vous utilisez l'attribut `contentID`, les autres attributs `simples..` sont ignorés car les détails complets sont extraits du modèle Engage.

#### Exemple `personalizedAttributeList`

```
personalizeAttributeList=discount=10,Offercost=20
campaignId=campaignname que vous souhaitez utiliser pour cette
campagne.
```

### 3. Votre table de profil comporte les colonnes et les valeurs suivantes.

```
appKey : gcsTQo6v79 recipientId : 13472242 deviceType : android ou
ios
```

### 4. Accédez à `INTERACT_HOME/conf/gateways/outbound/common`. Sous le dossier **Interact**, dans le fichier `httpConnectionConfig.properties`, spécifiez le délai d'expiration.

Par exemple:

```
connectTimeoutMs=6000
```

Lorsque OMO est configuré pour utiliser une connexion HTTP, un proxy HTTP peut être configuré en option avec une authentification entre Interact et le point de terminaison. Pour activer le proxy pour des passerelles sortantes, mettez à jour les valeurs des propriétés suivantes.

- `proxyHost`=< adresse IP du serveur proxy >
- `proxyPort`=< Port d'écoute du serveur proxy >
- `targetUsername`=< nom d'utilisateur pour se connecter au serveur proxy. laisser vide si aucune authentification requise >
- `targetPassword`=< mot de passe pour se connecter au serveur proxy. laisser vide si aucune authentification requise >

### 5. Créez un canal et un gestionnaire sous **Interact | triggerMessage** et utilisez la passerelle [Mobile\_Push] que vous avez créée ci-dessus dans ce canal. Ce canal est utilisé dans le message déclenché pour envoyer des messages push.

6. Créez un canal interactif et ajoutez un message déclenché qui utilise l'offre créée précédemment dans la règle de déclenchement.
7. Déployez le canal interactif.
8. À partir du client de test API, effectuez une `startSession` pour le canal interactif où la règle de message déclenché est configurée et le `postEvent` qui déclenche l'offre à Mobile Push.
9. Consultez les journaux Unica Interact pour vous assurer que la commande push a été envoyée avec succès. Le code d'état `202` signifie livraison réussie.

## Utilisation d'Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud

Vous pouvez utiliser cette intégration avec Silverpop, Unica Interact et Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud afin d'envoyer des offres par e-mail déclenchées à vos clients.

Vérifiez que les prérequis suivants sont respectés.

- Créez une table des profils d'audience client avec une colonne d'e-mail. Utilisez cette table de profils pour votre canal interactif.
- Demandez qu'un accès réseau à la machine hôte soit activé pour votre canal sortant.

### Ajout d'un répartiteur pour l'intégration de passerelle

Le répartiteur ajoute votre offre dans une file d'attente pour Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud de sorte que votre e-mail d'offre puisse être envoyé.

Vous devez ajouter un répartiteur afin d'utiliser HCL Email (Transact) Outbound Gateway for IBM Marketing Cloud.

1. Accédez à **Interact | triggeredMessage | répartiteurs | <nomRépartiteur>** dans les propriétés de configuration.
2. Ajoutez un **nouveau nom de catégorie** pour votre répartiteur.

3. Sélectionnez un **type**. Vous avez le choix entre InMemoryQueue, Kafka, JMSQueue et Custom.
4. Entrez le **nom de classe**.
5. Entrez le **chemin d'accès aux classes**.

## Configuration du paramètre OMO- conf\_outbound\_common\_httpConnectionConfig

Naviguez jusqu'à `INTERACT_HOME/conf/gateways/outbound/common`. Sous le dossier **Interact**, dans le fichier `httpConnectionConfig.properties`, spécifiez le dépassement du délai d'attente.

Par exemple : `connectTimeoutMs=60000` Lorsque OMO est configuré pour utiliser une connexion HTTP, un proxy HTTP peut être configuré avec une authentification entre Interact et le nœud final. Pour activer le proxy pour des passerelles sortantes, mettez à jour les valeurs des propriétés suivantes.

- `proxyHost=<Adresse IP du serveur proxy>`
- `proxyPort=<Port d'écoute du serveur proxy>`
- `targetUsername=<Nom d'utilisateur à utiliser pour la connexion au serveur proxy. Laisser cette propriété vide si aucune authentification n'est requise.>`
- `targetPassword=<Mot de passe à utiliser pour la connexion au serveur proxy. Laisser cette propriété vide si aucune authentification n'est requise.>`

## Configuration du paramètre OMO- conf\_outbound\_silverpop\_silverpopConfig

Dans le fichier `silverpopConfig.properties`, définissez des valeurs pour `OauthServiceURL`, `xmlAPIServiceURL`, `clientID`, `clientSecret` et `refreshToken`. Adressez-vous à l'administrateur de Marketing Cloud pour obtenir des valeurs propres à votre client à partir du fichier `transact.xml`.

## Configuration du paramètre OMO-conf\_outbound\_silverpop\_silverpopContentMapping

Vous devez configurer le paramètre OMO-conf\_outbound\_silverpop\_silverpopContentMapping pour votre passerelle.

Si le mappage basé sur le fichier de propriétés est utilisé, mettez à jour le fichier de propriétés avec les mappages suivants.

- Si un mappage basé sur l'interface utilisateur est utilisé, créez une passerelle avec le nom "EMail".
- Dans les sections Mappage, créez des mappages pour les propriétés suivantes.

Dans le fichier `silverpopContentMapping.properties`, définissez les valeurs de votre mappage de contenu.

- a. Définissez la propriété `campaignId`. La valeur de cette propriété correspond à un nom d'attribut d'offre spécifié dans vos modèles d'offre.
- b. Définissez la propriété `e-mail`. La valeur de cette propriété correspond au nom de colonne dans votre table de profils. Ajoutez une colonne `e-mail` dans votre tableau de profil et spécifiez les identifiants d'e-mail. Ce sont les identifiants de messagerie des destinataires.
- c. Définissez les attributs de votre offre dans `additionalOfferPfAttributesUsedInEmail`. Cette propriété définit les attributs de votre modèle d'offre requis pour le modèle de mailing. Vous pouvez utiliser `additionalProfilePfAttributesUsedInEmail` pour définir des champs à partir de votre table de profil. Vous pouvez utiliser `*` pour prendre en compte tous les attributs d'offre et les valeurs de colonne.

## Configuration du paramètre deliveryTimeoutMillis

Pour augmenter le délai d'expiration du serveur Unica Interact lors de sa connexion au serveur Marketing Cloud, définissez le paramètre `deliveryTimeoutMills`.



1. Accédez à **Interact | triggeredMessage | passerelles | <nomPasserelleSilverpop> | deliveryTimeoutMillis** dans les propriétés de configuration.
2. Définissez la **valeur**. Par exemple, vous pouvez définir la **valeur** sur 60000. Cela augmente le délai d'expiration du serveur à 60000 millisecondes.

## Ajout d'un gestionnaire de canal pour Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud

Ajoutez un gestionnaire de canal dans l'environnement d'exécution d'Unica Interact.

1. Accédez à **Interact | triggeredMessage | canaux | <nomCanalSilverpop> | <nomGestionnaire>** dans les propriétés de configuration.
2. Ajoutez un **nouveau nom de catégorie** pour votre gestionnaire de canal.
3. Définissez le nom du répartiteur que vous avez ajouté au préalable.
4. Définissez le nom de la passerelle.
5. Définissez le **mode**. Si **Basculément** est sélectionné, ce gestionnaire n'est utilisé que si aucun des gestionnaires de priorité supérieure définis dans ce canal n'est parvenu à envoyer des offres. Si **Module complémentaire** est sélectionné, ce gestionnaire est utilisé que d'autres gestionnaires aient réussi ou non à envoyer des offres.
6. Définissez la **priorité** de ce gestionnaire.

## Ajout d'un canal sortant pour Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud

Ajoutez un canal sortant dans l'environnement de conception d'Unica Interact.

1. Accédez à **Campaign | partitions | partition[n] | Interact | outboundChannels** dans les propriétés de configuration.
2. Ajoutez un **nouveau nom de catégorie** pour votre canal sortant.
3. Ajoutez un **nom** pour votre canal sortant. Assurez-vous que le nom de canal est identique à celui que vous avez ajouté dans la propriété de configuration **Interact | triggeredMessage | canaux | <nomCanalSilverpop>**.

## Configuration du mailing transactionnel avec Unica Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud

Vous devez configurer votre mailing transactionnel pour pouvoir envoyer votre offre par e-mail.

1. Dans Marketing Cloud (Transact), cliquez sur **Data > Create Database**. Cliquez ensuite sur **Create** pour créer une table de profils. Vous pouvez également importer la table de profils dans laquelle vous avez ajouté la colonne d'e-mail.
2. Cliquez sur **Automation > Transactional messages > Create Group**. Sélectionnez **Transact** pour l'option **Event Trigger**. Vous devez également sélectionner la source de données que vous avez créée précédemment. Cliquez sur **Save & Activate**.

L'offre envoyée via Marketing Cloud doit contenir l'attribut que vous avez défini pour `campaignId` dans le fichier `silverpopContentMapping.properties`. La valeur de cet attribut d'offre correspond à l'`campaignId` généré pour le groupe de messages automatisés.

3. Cliquez sur **Content > Create Mailings** et sélectionnez la source de contenu de l'étape précédente. Entrez le corps du mailing. Cliquez sur **Automate**. Sélectionnez **Assign Mailing to Existing Group of Automated Messages**. Cliquez sur **Submit & Activate**.

La ligne objet et le corps du mailing peuvent être personnalisés à l'aide des attributs d'offre et des attributs de profil. Utilisez la syntaxe `%%Attribute_Name%%` pour définir les attributs.

4. Le serveur Marketing Cloud n'accepte les soumissions de passerelles sortantes qu'à partir des adresses IP configurées à l'avance. Pour ajouter une adresse IP, accédez à **Settings > Org Admin > Security Settings > Access Restrictions**.
5. Si vous utilisez WebSphere Application Server, vous devez importer le certificat SSL de Marketing Cloud. Cette opération n'est pas requise pour les utilisateurs de WebLogic.
  - a. Dans la console WebSphere Application Server, accédez à **Certificat SSL et gestion des clés > Magasins de clés et certificats > NodeDefaultTrustStore > Certificats de signataires > Extraire d'un port**.
  - b. Définissez l'hôte et le port.
  - c. Redémarrez WebSphere Application Server.