Unica Interact V12.1.1
Administratorhandbuch

Contents

Chapter 1. Administration von Unica Interact..... 1

Unica Interact-Grundlagen..... 1

Zielgruppenebenen.....1

Designzeitumgebung..... 2

Ereignis und Ereignismuster..... 3

Interaktive Kanäle.....9

Interaktive Ablaufdiagramme..... 9

Interaktionspunkte..... 10

Angebote.....10

Profile..... 11

Laufzeitumgebung..... 12

Laufzeitsitzungen.....13

Touchpoints.....13

Strategie und Verfahrensregeln..... 13

FlexOffers..... 14

Gateways..... 16

Unica Interact Architektur.....25

Überlegungen zum Unica Interact-Netz.....27

Unica Interact Serverports und Netzwerksicherheit..... 28

Anmelden bei Unica..... 31

Chapter 2. Sicherheitsmanagement.....33

Authentifizierung der Unica Interact JSP-Seiten.....33

Chapter 3. Konfigurieren von Benutzern.....34

Den Laufzeitumgebungsbenutzer konfigurieren..... 34
Designumgebungsbenutzer konfigurieren.....36
Beispiel für Designumgebungsberechtigungen..... 39
Chapter 4. Verwalten von Unica Interact Datenquellen..... 43
Unica Interact-Datenquellen.....43
Datenbanken und die Anwendungen..... 44
Unica Campaign Systemtabellen..... 46
Laufzeittabellen..... 46
Testlauftabellen..... 49
Die Standarddatentypen für dynamisch erstellte Tabellen überschreiben.....50
Überschreiben der Standarddatentypen.....51
Standarddatentypen für dynamisch erstellte Tabellen..... 51
Profildatenbank.....52
Lerntabellen.....55
Kontaktverlauf für sitzungsübergreifende Antwortverfolgung.....55
Datenbankscripts zur Aktivierung von Unica-Funktionen ausführen..... 56
Zum Thema Sitzungsübergreifende Kontaktverfolgung.....58
Informationen zur Verfolgung von Kontakt- und Antwortverlauf..... 61
Kontakt- und Antworttypen..... 61
Zusätzliche Kontakttypen.....62
Zusätzliche Antworttypen..... 63
Zuordnung der Staging-Tabellen der Laufzeitumgebung zu den Unica Campaign-Verlaufstabellen..... 67
Konfigurieren der JMX-Überwachung für das Kontakt- und Antwortverlaufsmodul..... 77
Informationen zur sitzungsübergreifenden Antwortverfolgung.....78

Erkennung und Unterdrückung von Duplikaten aktivieren..... 79
Sitzungsübergreifender Antwortprozess.....80
Konfiguration der Quellendaten für die sitzungsübergreifende Antwortverfolgung..... 82
Konfigurieren von Kontakt- und Antwortverlaufstabellen für die sitzungsübergreifende Antwortverfolgung.....83
Aktivieren der sitzungsübergreifenden Antwortverfolgung..... 88
Sitzungsübergreifender Abgleich von Angebot und Antwort.....89
Verwenden eines Datenbankladedienstprogramms mit der Laufzeitumgebung..... 93
Aktivieren eines Datenbankladedienstprogramms mit der Laufzeitumgebung.....95
ETL-Prozess für Ereignismuster.....96
Ausführen des eigenständigen ETL-Prozesses..... 96
Stoppen des eigenständigen ETL-Prozesses..... 99
Chapter 5. Services für Angebote.....101
Angebotsberechtigung..... 101
Erstellen einer Liste von möglichen Angeboten..... 101
Berechnen des Marketing-Score.....103
Den Lernprozess beeinflussen..... 105
Angebote unterdrücken.....105
Aktivieren der Angebotsunterdrückung.....106
Angebotsunterdrückungstabelle.....106
Angebotsunterdrückung ignorieren..... 107
Globale Angebote und individuelle Zuweisungen.....108
Definieren der Standardzellencodes.....109
Definieren von Angeboten, die nicht in einer Verfahrensregel verwendet werden..... 109

Informationen zur globalen Angebotstabelle..... 110
Zuweisen von globalen Angeboten..... 111
Globale Angebotstabelle..... 111
Informationen zur Bewertungsüberschreibungstabelle..... 116
Konfigurieren von Bewertungsüberschreibungen..... 117
Bewertungsüberschreibungstabelle..... 117
Übersicht über das integrierte Lernen von Unica Interact..... 122
Unica Interact-Lernmodul..... 123
Aktivieren des Lernmoduls..... 126
Lernattribute..... 126
Definieren eines Lernattributs..... 129
Definieren von dynamischen Lernattributen..... 130
Unica Interact AutoBinning..... 131
Konfigurieren der Laufzeitumgebung für die Erkennung von externen Lernmodulen..... 133
Chapter 6. Informationen zur Unica Interact-API..... 134
Unica Interact-API-Datenfluss..... 135
Einfaches Beispiel für Interaktionsplanung..... 140
Entwerfen der Unica Interact-API-Integration..... 147
Zu berücksichtigende Punkte..... 148
API-Authentifizierung..... 149
Chapter 7. Verwalten der Unica Interact-API..... 151
Ländereinstellungen und die Unica Interact-API..... 151
Informationen zur JMX-Überwachung..... 152
Konfigurieren von Unica Interact zur Verwendung der JMX-Überwachung mit dem RMI-Protokoll..... 153

Konfigurieren von Unica Interact zur Verwendung der JMX-Überwachung mit dem JMXMP-Protokoll....153

Konfigurieren von Unica Interact für die Verwendung der jconsole-Scripts zur JMX-Überwachung....154

JMX-Attribute....155

JMX-Operationen....178

Thread Überwachung....180

Chapter 8. Klassen und Methoden für die Java-, SOAP- und REST-API von Unica Interact....182

Unica Interact API-Klassen....182

Methoden zur Übergabe der Authentifizierungsparameter, wenn die API-Authentifizierung vor API-Aufrufen aktiviert ist....183

Java™ Anordnung über HTTP-Voraussetzungen....183

SOAP-Voraussetzungen....184

Voraussetzungen für REST....185

API JavaDoc....186

API-Beispiele....187

Arbeiten mit Sitzungsdaten....187

Informationen zur Klasse InteractAPI....188

endSession....189

executeBatch....190

getInstance....194

getOffers....195

getOffersForMultipleInteractionPoints....198

getProfile....201

getVersion....204

postEvent....205
setAudience....209
setDebug....212
startSession....213
Reservierte Parameter....222
Informationen zur Klasse AdvisoryMessage....238
getDetailMessage....238
getMessage....239
getMessageCode....240
getStatusLevel....240
Informationen zur Klasse AdvisoryMessageCode....241
Codes von Empfehlungsnachrichten....241
Informationen zur Klasse BatchResponse....252
getBatchStatusCode....252
getResponses....254
Informationen zur Command-Benutzeroberfläche....254
setAudienceID....255
setAudienceLevel....257
setDebug....258
setEvent....259
setEventParameters....260
setGetOfferRequests....262
setInteractiveChannel....264
setInteractionPoint....265
setMethodIdentifier....266

setNumberRequested..... 267
setRelyOnExistingSession..... 268
Informationen zur NameValuePair-Benutzeroberfläche..... 268
getName..... 269
getValueAsDate..... 269
getValueAsNumeric..... 270
getValueAsString..... 271
getValueDataType..... 271
setName..... 273
setValueAsDate..... 273
setValueAsNumeric..... 274
setValueAsString..... 274
setValueDataType..... 275
setScope(Umfang)..... 276
getScope()..... 277
Informationen zur Klasse Offer..... 277
getAdditionalAttributes..... 277
getDescription..... 279
getOfferCode..... 279
getOfferName..... 280
getScore..... 281
getTreatmentCode..... 282
Informationen zur Klasse OfferList..... 283
getDefaultString..... 283
getRecommendedOffers..... 284

Informationen zur Klasse Response....285
getAdvisoryMessages....285
getApiVersion....286
getOfferList....287
getAllOfferLists....287
getProfileRecord....288
getSessionID....289
getStatusCode....290
Chapter 9. Klassen und Methoden für die Unica Interact-JavaScript-API....292
JavaScript-Voraussetzungen....292
Arbeiten mit Sitzungsdaten....292
Mit Callback-Parameter arbeiten....294
Informationen zur Klasse InteractAPI....295
startSession....295
getOffers....303
getOffersForMultipleInteractionPoints....305
setAudience....308
getProfile....310
endSession....311
setDebug....312
getVersion....313
executeBatch....314
Beispiel für JavaScript-API....315
Beispiel für JavaScript-Antwortobjekt "onSuccesss"....330
Chapter 10. Informationen zur ExternalCallout-API....333

IAffiniumExternalCallout-Benutzeroberfläche....333
Hinzufügen eines Web-Service zur Verwendung mit dem Makro EXTERNALCALLOUT....334
getNumberOfArguments....335
getValue....335
Initialisieren....336
shutdown....337
Beispiel für die ExternalCallout-API....338
Benutzeroberfläche IInteractProfileDataService....341
Hinzufügen einer Datenquelle zur Verwendung mit Profildatenservices....341
Benutzeroberfläche IParameterizableCallout....342
Initialisieren....343
shutdown....344
Benutzeroberfläche ITriggeredMessageAction....344
getName....345
setName....345
Benutzeroberfläche IChannelSelector....345
selectChannels....346
Benutzeroberfläche IDispatcher....347
dispatch....348
Benutzeroberfläche IGateway....349
Deliver....349
Bestätigen....350
Chapter 11. Unica Interact-Dienstprogramme....352
Dienstprogramm RunDeployment (runDeployment.sh/.bat)....352

Bereinigung abgelaufenes Token-Dienstprogramm....359

Chapter 12. Informationen zur Lern-API....361

Konfigurieren der Laufzeitumgebung für die Erkennung von externen Lernmodulen....363

Benutzeroberfläche ILearning....364

Initialisieren....364

logEvent....365

optimizeRecommendList....366

reinitialize....367

shutdown....368

Benutzeroberfläche IAudienceID....369

getAudienceLevel....369

getComponentNames....369

getComponentValue....370

IClientArgs....370

getValue....370

IInteractSession....371

getAudienceId....371

getSessionData....371

Benutzeroberfläche IInteractSessionData....372

getDataType....372

getParameterNames....372

getValue....372

setValue....373

ILearningAttribute....374

getName..... 374
ILearningConfig..... 374
ILearningContext..... 375
getLearningContext..... 375
getResponseCode..... 375
IOffer..... 376
getCreateDate..... 376
getEffectiveDateFlag..... 376
getExpirationDateFlag..... 377
getOfferAttributes..... 377
getOfferCode..... 377
getOfferDescription..... 378
getOfferID..... 378
getOfferName..... 378
getUpdateDate..... 379
IOfferAttributes..... 379
getParameterNames..... 379
getValue..... 379
Benutzeroberfläche IOfferCode..... 380
getPartCount..... 380
getParts..... 380
LearningException..... 381
IScoreOverride..... 381
getOfferCode..... 381
getParameterNames..... 381

getValue....382
ISelectionMethod....383
Benutzeroberfläche ITreatment....383
getCellCode....383
getCellId....384
getCellName....384
getLearningScore....384
getMarketerScore....385
getOffer....385
getOverrideValues....386
getPredicate....386
getPredicateScore....386
getScore....387
getTreatmentCode....387
setActualValueUsed....388
Beispiel für eine Lern-API....388
Chapter 13. Unica Interact-WSDL....397
Chapter 14. Unica Interact Laufzeitumgebung - Konfigurationseigenschaften....420
Interact | Allgemein....420
Interact | Allgemein | API....421
Interact | Allgemein | centralizedLogger....422
Interact | Allgemein | learningTablesDataSource....423
Interact | Allgemein | prodUserDataSource....426
Interact | general | API | requestThreadPool....428
Interact | Allgemein | systemTablesDataSource....430

Interact | Allgemein | testRunDataSource....437

Interact | Allgemein | contactAndResponseHistoryDataSource....440

Interact | Allgemein | idsByType....442

Interact | Ablaufdiagramm....443

Interact | Ablaufdiagramm| ExternalCallouts | [ExternalCalloutName]....445

Interact | Ablaufdiagramm | ExternalCallouts | [ExternalCalloutName] | Parameterdaten | [parameterName]....447

Interact | Überwachung....448

Interact | Überwachung| activitySubscribers....449

Interact | Profil....452

Interact | Profil | Zielgruppen | [AudienceLevelName]....454

Interact | Profil | Zielgruppen | [AudienceLevelName] | Angebote von Raw SQL....460

Interact | profile | Zielgruppe | [AudienceLevelName | Profildatendienste | [DataSource]....463

Affinium|interact|profile|Audience Levels|[Audience Levels]requestLogTable....466

Interact | offerserving....466

Interact | Offerserving | integrierte Lernmodul Konfiguration....471

Interact | offerserving | integrierte Lernmodul Konfiguration | Datenparameter | [parameterName]....473

Interact | Offerserving | External Learning Config....475

Interact | offerserving | External Learning Config | Parameter Data | [parameterName]....476

Interact | offerserving | Einschränkungen....477

Interact | Dienste....478

Affinium|interact|Dienste|contactHist|treatmentStoreReference....479

daysBackForXSessContact....479

Interact | Dienste | contactHist....479

Interact | Dienste | contactHist | cache....480

Interact | Dienste | contactHist | contactStatusCodes....481

Interact | Dienste | contactHist | fileCache....482

Interact | Dienste | defaultedStats....483

Interact | Dienste | defaultedStats | cache....483

Interact | Dienste | eligOpsStats....484

Interact | Dienste | eligOpsStats | cache....485

Interact | Dienste | eventActivity....485

Interact | Dienste | eventActivity | cache....486

Interact | Dienste | eventPattern....486

Interact | Dienste | eventPattern | userEventCache....488

Interact | Dienste | eventPattern | advancedPatterns....489

Interact | Dienste | customLogger....493

Interact | Dienste | customLogger | cache....493

Interact | Dienste | responseHist....494

Interact | Dienste | responseHist | cache....497

Interact | Dienste | response Hist | responseTypeCodes....498

Interact | Dienste | responseHist | fileCache....499

Interact | Dienste | crossSessionResponse....500

Interact | Dienste | crossSessionResponse | cache....502

Interact | Services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode....503

Interact | Dienste | crossSessionResponse | OverridePerAudience | [Zielgruppe] | TrackingCodes | byOfferCode....504

Interact | Dienste | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode.....506

Interact | Services | threadManagement | contactAndResponseHist.....507

Interact | Services | threadManagement | allOtherServices.....509

Interact | Dienste | threadManagement | flushCacheToDB.....511

Interact | Services | threadManagement | eventHandling.....512

Interact | Dienste | configurationMonitor.....514

Interact | Dienste | CampaignSegments.....515

Interact | cacheManagement.....517

Interact | cacheManagement | Cache-Manager.....517

Interact | Caches.....520

Interact | triggeredMessage.....533

Interact | triggeredMessage | offerSelection.....534

Interact | triggeredMessage | Dispatchers.....535

Interact | triggeredMessage | gateways | <gatewayName>.....550

Interact | triggeredMessage | channels.....553

Interact | activityOrchestrator.....556

Interact | activityOrchestrator | receivers.....556

Interact | activityOrchestrator | gateways.....567

Interact | ETL | patternStateETL.....568

Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS.....571

Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS.....573

Interact | ETL | patternStateETL | <patternStateETLName> | Bericht.....576

Chapter 15. Unica Interact Simulator.....578

Interact | Simulator.....578

Interact | simulator|scenarioDataSource..... 579

Chapter 16. Unica Interact Designumgebung - Konfigurationseigenschaften..... 584

Campaign | Partitionen | Partition[n] | Berichte.....584

Campaign | Partitionen | Partition[n] | UnicaInsightsReports.....588

Campaign | Partionen | partition[n] | Interact | contactAndResponseHistTracking.....588

Campaign | Partitionen | Partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]..... 595

Campaign | partionen | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings..... 596

Campaign | partionen | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings..... 597

Campaign | partionen | partition[n] | Interact | Bericht..... 598

Campaign | partionen | partition[n] | Interact | lernen.....599

Campaign | partionen | partition[n] | Interact | lernen | learningAttributes | [learningAttribute]..... 604

Campaign | partionen | partition[n] | Interact | einsatz..... 605

Campaign | partionen | partition[n] | Interact | serverGroups | [serverGroup]..... 605

Campaign | partionen | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL].....605

Campaign | partionen | partition[n] | Interact | Ablaufsdiagramm.....606

Campaign | partionen | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers.....608

Campaign | partionen | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL..... 608

Campaign | partionen | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride..... 609

Campaign | Partitionen | Partition[n] | Server | intern..... 610

Campaign | monitoring..... 616
Campaign | partionen | partition[n] | Interact | outboundChannels..... 619
Campaign | partionen | partition[n] | Interact | outboundChannels | Parameter Datei..... 620
Campaign | partitions | partition[n] | Interact | Simulator..... 621
offerArbitrition..... 621
Chapter 17. Echtzeit-Personalisierung von Angeboten auf der Clientseite..... 623
Informationen zum Unica Interact Message Connector..... 623
Installieren des Message Connectors..... 625
Erstellen der Message Connector-Links..... 640
Informationen zum Unica Interact Web Connector..... 646
Installieren des Web Connectors auf dem Laufzeitserver..... 647
Installieren des Web Connectors als separate Webanwendung..... 648
Konfigurieren des Web Connectors..... 650
Verwenden der Administratorseite in Web Connector..... 674
Web Connector-Beispielseite..... 676
Chapter 18. Integration von Unica Interact und Digital Recommendations..... 682
Übersicht über die Integration von Unica Interact mit Digital Recommendations..... 682
Voraussetzungen für die Integration..... 684
Konfigurieren eines Angebots mit Digital Recommendations-Integration..... 685
Verwenden des Integrationsbeispielprojekts..... 686
Chapter 19. Integration von Unica Interact und Digital Data Exchange..... 702
Voraussetzungen..... 702
Unica Interact mit IBM Digital Data Exchange in Website integrieren..... 703
Unica Interact-Tags in Digital Data Exchange..... 704

Sitzung beenden..... 705
Angebot abrufen..... 705
Bibliothek laden..... 706
Ereignis senden..... 707
Zielgruppe festlegen..... 708
Sitzung starten..... 709
Beispiel für Tageinstellungen..... 710
Integrationskonfiguration überprüfen..... 716
Chapter 20. Integration von Unica Interact und Unica Journey..... 718
Übersicht..... 718
Die Feldzuordnung von Interact und Journey..... 719
Interact-Laufzeitkonfigurationen..... 722
Implementierung..... 723
Chapter 21. Integration von Unica Interact und Unica Deliver..... 724
Übersicht..... 724
Zuordnung von Interact und Deliver..... 725
Interact-Laufzeitkonfigurationen..... 726
Implementierung..... 726
Chapter 22. Gateways konfigurieren..... 728
Verwendung von Unica Interact Inbound Gateway für IBM Universal Behavior Exchange..... 729
Verwendung von Unica Interact Outbound Gateway für IBM Universal Behavior Exchange..... 738
Verwendung von Unica Interact Outbound Gateway für IBM Mobile Push Notification..... 741

Verwendung von Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud..... 744

Hinzufügen eines Dispatchers für die Gateway-Integration..... 745

Parameter "OMO-conf_outbound_common_httpConnectionConfig" konfigurieren.....745

Parameter "OMO-conf_outbound_silverpop_silverpopConfig" konfigurieren.....746

Parameter "OMO-conf_outbound_silverpop_silverpop ContentMapping" konfigurieren.....746

Parameter "deliveryTimeoutMillis" konfigurieren..... 747

Hinzufügen eines Kanal-Handlers für das Unica Interact E-Mail (Transact) Outbound-Gateway für die IBM Marketing Cloud.....747

Hinzufügen eines Kanals für abgehende Nachrichten für Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud.....748

Konfigurieren des Transaktionsmailing mit Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud.....748

Index.....

Kapitel 1. Administration von Unica Interact

Im Rahmen Ihrer Administration von Unica Interact konfigurieren und verwalten Sie Benutzer und Rollen, Datenquellen sowie optionale Produktfunktionen. Darüber hinaus überwachen und verwalten Sie die Design- und Laufzeitumgebungen. Hierfür stehen Ihnen produktspezifische Anwendungsprogrammierschnittstellen (APIs) zur Verfügung.

Die Administration von Unica Interact besteht aus mehreren Aufgaben. Zu diesen Aufgaben gehören, ohne darauf beschränkt zu sein, folgende:

- Benutzer und Rollen verwalten
- Datenquellen verwalten
- Optionale Funktionen der Unica Interact-Angebotsunterbreitung konfigurieren
- Laufzeitumgebungsleistung überwachen und warten

Bevor Sie mit der Administration von Unica Interact beginnen, sollten Sie sich mit einigen Schlüsselkonzepten in Bezug auf die Funktionsweise von Unica Interact vertraut machen, damit Sie Ihre Aufgaben einfacher ausführen können. In den folgenden Abschnitten werden die Administrationsaufgaben beschrieben, die mit Unica Interact in Verbindung stehen.

Im zweiten Teil des Administrationshandbuchs finden Sie eine Beschreibung der APIs, die in Unica Interact verfügbar sind:

- Unica Interact-API
- ExternalCallout-API
- Lern-API

Unica Interact-Grundlagen

Unica Interact ist eine interaktive Engine, die personalisierte Marketingangebote an verschiedene Zielgruppen richtet.

Dieser Abschnitt beschreibt die wichtigsten Konzepte, mit denen Sie sich vor dem Arbeiten mit Unica Interact vertraut machen sollten.

Zielgruppenebenen

Eine Zielgruppenebene ist eine Sammlung von IDs, auf die eine Kampagne ausgerichtet werden kann. Sie können Zielgruppenebenen definieren, um Ihre Kampagne auf die richtigen Zielgruppen auszurichten.

Beispielsweise kann eine Gruppe von Kampagnen über die Zielgruppenebenen "Haushalt", "Interessent", "Kunde" und "Konto" verfügen. Jede dieser Ebenen stellt eine bestimmte Ansicht der für eine Kampagne verfügbaren Marketingdaten dar.

Zielgruppenebenen sind gewöhnlich hierarchisch organisiert. Für die obigen Beispiele:

- "Haushalt" steht an der Spitze der Hierarchie und jeder Haushalt kann mehrere Kunden sowie einen oder mehrere Interessenten enthalten.
- Darauf folgt in der Hierarchie "Kunde", und jeder Kunde kann über mehrere Konten verfügen.
- "Konto" ist der niedrigste Hierarchiepunkt.

Weitere, komplexere Beispiele für Zielgruppenhierarchien bestehen in B2B-Umgebungen, wo es möglicherweise Zielgruppenebenen für Unternehmen, Firmen, Abteilungen, Gruppen, Einzelpersonen, Konten usw. gibt.

Diese Zielgruppenebenen können unterschiedliche Beziehungen zueinander haben, beispielsweise "eins-zu-eins", "viele-zu-eins" oder "viele-zu-viele". Durch die Definition von Zielgruppenebenen ermöglichen Sie die Darstellung dieser Konzepte innerhalb von Unica Campaign, sodass Anwender die Beziehungen zwischen diesen verschiedenen Zielgruppen verwalten können, um ihre Kampagnen zielgenauer auszurichten. So möchten Sie vielleicht Mailings auf einen Interessenten pro Haushalt beschränken, obwohl sich in einem Haushalt vielleicht mehrere Interessenten befinden.

Designzeitumgebung

Verwenden Sie die Designzeitumgebung, um verschiedene Unica Interact-Komponenten zu konfigurieren und in der Laufzeitumgebung bereitzustellen.

Den größten Teil der Konfiguration von Unica Interact führen Sie in der Designumgebung aus. In der Designzeitumgebung definieren Sie interaktive Kanäle, interaktive

Ablaufdiagramme, Strategien und Verfahrensregeln, Ereignisse und Ereignismuster, Interaktionspunkte, Smart Segments und FlexOffers. Nachdem Sie diese Komponenten konfiguriert haben, stellen Sie sie in der Laufzeitumgebung bereit.

Die Designzeitumgebung wird mit der Unica Campaign-Webanwendung installiert.

Ereignis und Ereignismuster

Ereignis

Ein Ereignis stellt eine aufgetretene Benutzeraktivität dar, die eine Aktion in der Laufzeitumgebung auslösen kann. Eine Website besuchen, ein Konto eröffnen, der Kundendienst anrufen usw. sind einige Beispiele für ein Ereignis.

Ereignisse werden zunächst in Interactive Channels über die Interact Designzeit-Benutzeroberfläche erstellt und dann durch Aufruf der Laufzeit-API postEvent an die Interact-Laufzeitumgebung übermittelt.

Ereignismuster

Ein Ereignismuster besteht aus einer Reihe von Ereignissen, die auf eine bestimmte Weise auftreten. Die Ereignismuster können von Vermarktern zur Verfolgung und Erfassung der Struktur von Kundenaktivitäten in Echtzeit und ihre entsprechende Behandlung verwendet werden. Ein Muster beginnt mit dem Musterzustand 'Bedingungen nicht erfüllt'. Durch das Posten von Ereignissen an Interact in ausgewählten Phasen der Kundenaktivitäten wird der Musterzustand überprüft und aktualisiert. Wenn alle definierten Ereignisse für das Muster auf die definierte Weise auftreten, wird der Musterzustand auf 'Bedingungen erfüllt' geändert und die konfigurierten Aktionen werden ausgelöst. Die Ereignismuster können in Kundensegmentierungen verwendet werden und bieten Entscheidungslogiken.

Die folgenden neun Arten von Ereignismustern werden von Interact unterstützt.

- Alle abgleichen
- Zähler
- Gewichteter Zähler
- Alle abgleichen (zeitgebunden)
- Zähler (zeitgebunden)

- Gewichteter Zähler (zeitgebunden)
- Alle abgleichen (gleitende Zeit)
- Zähler (gleitende Zeit)
- Gewichteter Zähler (gleitende Zeit)

Alle abgleichen: Es ist ein Muster, das ausgelöst wird (in den Zustand 'Bedingungen erfüllt' versetzt wird), wenn alle zusammengesetzten Ereignisse auftreten. Z.B.: Die Bedingungen des Musters sind erfüllt nur wenn alle Ereignisse d.h. 'Ereignis A' und 'Ereignis B' und 'Ereignis C' aufgetreten sind. Die Reihenfolge von dem Ereignisauftritt spielt keine Rolle.

Zähler: Dies ist ein Muster, das ausgelöst wird, wenn jedes zusammengesetzte Ereignis mehr als die vordefinierte Anzahl auftritt. Z.B.: Die Bedingungen des Muster sind erfüllt, wenn 'Ereignis A' >= 5 Mal und 'Ereignis B' >= 5 Mal auftritt. Die Reihenfolge von dem Ereignisauftritt spielt keine Rolle.

Gewichteter Zähler: Es ist ein Muster, bei dem jedes zusammengesetzte Ereignis gewichtet wird und das Muster wird ausgelöst nur wenn die Gesamtsumme eine vordefinierte Anzahl erreicht. Z.B.: Ein Muster besteht aus 'Ereignis A' mit einer Bewertung von 2 und 'Ereignis B' mit einer Bewertung von 5, d.h. die Gesamtsumme beträgt eine Bewertung von 10, dann sind die Bedingungen des Musters beim Auftritt von einer der folgenden Situationen erfüllt.

- 'Ereignis A' tritt fünfmal auf, weil 5x2=10
- 'Ereignis B' tritt zweimal auf, weil 2x5=10
- 'Ereignis A' tritt dreimal und 'Ereignis B' einmal auf, weil 3x2 + 1x5 = 11.

Es gibt keine zeitlichen Beschränkungen für die Mustertypen Alle abgleichen, Zähler und Gewichteter Zähler. Solange die geposteten Ereignisse innerhalb des definierten Start- und Enddatums liegen, werden sie für das Muster bewertet. Wenn kein Startdatum definiert ist, wird das Muster sofort nach der Bereitstellung wirksam. Wenn kein Enddatum definiert ist, wird das Muster dauerhaft wirksam. Die Vermarkter können die Funktion 'Muster zurücksetzen' verwenden, um den Musterstatus für diese drei Musterarten zurückzusetzen. Im Gegensatz dazu sind zeitgebundene Muster und gleitende Zeitmuster als zeitgebundene Muster betrachtet.

Gleitendes Zeitmuster: Die Muster 'Alle abgleichen', 'Zähler' oder 'Gewichteter Zähler' können als gleitendes Zeitmuster betractet werden, aber alle zusammengesetzten Ereignisse müssen innerhalb eines Zeitfensters auftreten. Zu jedem Zeitpunkt, an dem ein zusammengesetztes Ereignis an Interact Laufzeit gepostet wird, überprüft Interact den Auftritt aller zusammengesetzten Ereignisse des Musters im Zeitfenster ab dem aktuellen Zeitpunkt. Sollten die Ereignisauftritte und die Musterdefinitionen nicht übereinstimmen, bleibt der Musterstatus 'Bedingungen nicht erfüllt'. Andernfalls, wenn alle Ereignisse innerhalb des Zeitfensters aufgetreten sind, wird der Musterstatus auf 'Bedingungen erfüllt' gesetzt (kann Aktionen auslösen, falls konfiguriert). Danach wird der Musterzustand auf die gleiche Weise wie oben beschrieben ständig neu bewertet und fortlaufend wiederholt

Zeitgebundenes Muster: Die Muster 'Alle abgleichen', 'Zähler' oder 'Gewichteter Zähler' können als zeitgebundenes Muster betractet werden, aber alle zusammengesetzten Ereignisse müssen innerhalb eines Zeitfensters auftreten. Zu jedem Zeitpunkt, an dem ein zusammengesetztes Ereignis an Interact Laufzeit gepostet wird, überprüft Interact den Auftritt aller zusammengesetzten Ereignisse des Musters im Zeitfenster ab dem aktuellen Zeitpunkt. Sollten die Ereignisauftritte und die Musterdefinitionen nicht übereinstimmen, bleibt der Musterstatus 'Bedingungen nicht erfüllt'. Andernfalls, wenn alle Ereignisse innerhalb des Zeitfensters aufgetreten sind, wird der Musterstatus auf 'Bedingungen erfüllt' gesetzt (kann Aktionen auslösen, falls konfiguriert). Nun prüft Interact eine andere Einstellung mit dem Namen "Status 'Erweitern wahr' für zusätzlichen Zeitraum erweitern" und behält das Muster als Status "Bedingung erfüllt" für den zusätzlichen Zeitraum (keine Musterbewertung in diesem Zeitraum). Wenn zusätzliche Zeit abgelaufen ist, wird der Musterzustand auf 'Bedingungen nicht erfüllt' zurückgesetzt und die Bewertung beginnt einen neuen Zyklus. Anders gesagt, das zeitgebundenes Muster ermöglicht es dem Muster, nach Erfüllung der Bedingung für eine bestimmte Zeit anzuhalten. Die Einstellung "Status 'Erweitern wahr' für zusätzlichen Zeitraum erweitern" ist nur auf das zeitgebundene Muster anwendbar.

Beispielsweise ist P1 ein zeitgebundenen Muster, und P2 ist ein rollierendes Zeitmuster. Beide Muster bestehen aus 'Ereignis A' und 'Ereignis B' und müssen innerhalb von 7 Tagen auftreten. Zur Laufzeit wurde 'Ereignis A' am Montag und 'Ereignis B' am Samstag aufgetreten. Nachdem 'Ereignis B' aufgetreten ist, wird der Musterzustand sowohl für P1 als auch für P2 in 'Bedingungen erfüllt' geändert, da zwei Ereignisse innerhalb von 7 Tagen

aufgetreten sind. Wenn nun für P1 die Einstellung "Status 'Erweitern wahr' für zusätzlichen Zeitraum erweitern" 4 Tage beträgt, bleibt P1 bis Mittwoch im Status "Bedingung erfüllt", und dann werden alle Vorkommen zweier Ereignisse gelöscht und das Muster beginnt am Mittwoch von neuem. Im Gegensatz dazu wird der Zustand von P2 nach dem Samstag kontinuierlich ausgewertet. Sollte 'Ereignis B' am Dienstag auftreten, wird der Zustand von P1 'Bedingungen nicht erfüllt' weil 'Ereignis A' vom letzten Mittwoch bis zu diesem Dienstag nicht aufgetreten ist.

Qualifiziertes Ereignis und ausgesetztes Ereignis: Ein Ereignismuster besteht aus einer Reihe von Ereignissen. Die Ereignisse, die den Musterzustand in 'Bedingungen erfüllt' ändern, werden als qualifizierte Ereignisse bezeichnet. Die Ereignisse, die zur Anhaltung der Muster führen werden als ausgesetzte Ereignisse bezeichnet. Z.B. Ein Muster hat zwei Ereignisse 'open_bank_account', 'ATM_activity' und 'offer_credit_card'. Alle diese Ereignisse müssen innerhalb von 2 Monaten auftreten. Wenn ein Kunde innerhalb von einem Monat nach Kontoeröffnung eine Bankkreditkarte beantragt und erhalten hat, wird er von den Vermarktern noch einmal mit dem Angebot von Kreditkarte nicht belästigt. Daher können Vermarkter ein ausgesetztes Ereignis 'got_card' im Muster definieren, damit die Bewertung angehalten wird. Die Vermarkter können auch die Einstellung 'Effektive Dauer' verwenden, um festzulegen, ob das Muster dauerhaft oder nur für einen bestimmten Zeitraum ausgesetzt wird.

Ereignismakro: Neben den von Kunden definierten Ereignissen werden sechs Ereignismakros von Interact unterstützt, die in der Musterdefinition entweder als qualifizierte Ereignisse oder ausgesetzte Ereignisse einbezogen sind. Folgende sind die sechs Makros.

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

offerAccepted, offerContacted oder offerRejected für ein Angebot können als Ereignis in einem Muster betrachtet werden. offerAcceptedInCategory, offerContactedInCategory oder offerRejectedInCategory können alle Angebote haben, die einen gleichen Attributwert wie ein Ereignis in einem Muster haben.

Muster der Inaktivität: Ein Muster kann nicht nur auf 'Bedingungen erfüllt', sondern auch auf 'Bedingungen nicht erfüllt' ausgewertet werden. Diese Funktion kann von Vermarktern zur Verfolgung von Kunden-Inaktivitäten verwendet werden. Z.B.: ein Muster hat zwei Ereignisse, 'add_item_to_cart' und 'checkout'. Alle diese Ereignisse müssen innerhalb von sieben Tagen auftreten. Die Vermarkter können am 3. Tag einen Kontrollpunkt hinzufügen, falls der Kunde den Artikel noch nicht ausgecheckt hat, d. h. wäre der Musterzustand 'Bedingungen nicht erfüllt', würde eine Aktion 'send_reminder_email' für den Kunden ausgeführt.

Ereigniskategorie

Ereignisse oder Ereignismuster lassen sich in der Designumgebung bei Bedarf in Kategorien einteilen. Ereigniskategorien haben in der Laufzeitumgebung keine bestimmte Funktion.

Aktionen

Eine Aktion kann ausgelöst werden, wenn ein Ereignis auftritt oder wenn die Bedingungen des Ereignisschemas erfüllt oder nicht erfüllt werden. Sie werden in Interact Designzeit bei der Definition von Ereignissen oder Ereignismustern konfiguriert.

Interact unterstützt acht Arten von Aktionen.

- Neusegmentierung auslösen: Die Laufzeitumgebung führt erneut alle oder eine Teilmenge der interaktiven Ablaufdiagramme für die aktuelle Zielgruppenebene aus, die dem interaktiven Kanal zugeordnet ist, und verwendet dazu in der Sitzung des Besuchers die aktuellen Daten. Dies ist nützlich, um den Besucher in neue Segmente einzuordnen, nachdem die wichtigen neuen Daten im Laufzeit-Sitzungsobjekt geändert wurden, wie z. B. neue Daten von Unica Interact-API Anforderungen (z.B. die Änderung von Zielgruppen) oder Kundenaktionen (z.B. Hinzufügung neuer Artikel zu einer Wunschliste oder einen Warenkorb). Es muss darauf geachtet werden, dass

eine starke Neusegmentierung innerhalb eines einzigen Besuchs die Leistung des Touchpoints beeinträchtigen kann, was auch für den Kunden sichtbar wird.

- Angebotskontakt protokollieren: Die Laufzeitumgebung kennzeichnet die empfohlenen Angebote für den Datenbankservice, um sie im Kontaktverlauf zu protokollieren. Protokollieren Sie bei Webintegrationen den Angebotskontakt in demselben Aufruf, in dem Sie Angebote anfordern, um die Anzahl der Anforderungen zwischen dem Touchpoint und dem Laufzeitserver zu minimieren. Wenn der Touchpoint den Verfahrenscode für die Angebote nicht angibt, die Unica Interact dem Besucher unterbreitet hat, protokolliert die Laufzeitumgebung die letzte Liste von empfohlenen Angeboten.
- Angebotsannahme protokollieren: Die Laufzeitumgebung kennzeichnet das ausgewählte Angebot für den Datenbankservice, um es im Antwortverlauf zu protokollieren.
- Angebotsablehnung protokollieren: Die Laufzeitumgebung kennzeichnet das ausgewählte Angebot für den Datenbankservice, um es im Antwortverlauf zu protokollieren.
- Benutzerausdruck auslösen: Eine Ausdrucksaktion ist eine Aktion, bei der Sie den Wert einer Sitzungsvariablen definieren können, indem Sie Profilattribute, Echtzeitattribute zusammen mit Unica Interact-Makros, einschließlich Funktionen, Variablen und Operatoren, darunter EXTERNALCALLOUT, verwenden. Sie können den Rückgabewert des Ausdrucks einem beliebigen Profilattribut zuweisen
- Ereignisse auslösen: Sie können die Aktion 'Ereignisse auslösen' verwenden, um ein weiteres oder mehrere Ereignisse auszulösen, wenn ein Quellereignis auftritt. Dies ermöglicht Vermarktern, Ereignisse zu verketten.
- Angebote unterdrücken. Die Angebotsunterdrückung kann von Ereignissen und Ereignismustern ausgelöst werden. Die Unterdrückungsregeln können abhängig von bestimmten Angeboten oder einer Gruppe von Angeboten mit denselben Attributwerten definiert werden. Der Unterschied zwischen einer Aktion von Angebotsunterdrückung und den bestehenden Unterdrückungsregeln besteht darin,

dass die Aktion ohne eine Verbindung mit den Verfahrensregeln ausgelöst werden kann.
- Segmente qualifizieren. Der Benutzer kann ein Segment angeben, das als Ergebnis eines Ereignisses oder Ereignismusters aktiviert wird.

Neben dem sofortigen Aufruf von Aktionen beim Ereignisauftritt oder bei der Erfüllung von einer Musterbedingung, können Aktionen auch mit einer Verzögerung aufgerufen werden, entweder verzögert nach einer Zeitspanne oder zu einem geplanten Datum und einer geplanten Uhrzeit. Damit können die Vermarkter die Ausführung von Aktionen zu bevorzugten Zeiten kontrollieren. Die Aktionsverzögerung gilt nicht für 'Angebotsunterdrückung' und 'qualifizierte Segmente.

Interaktive Kanäle

Verwenden Sie die interaktiven Kanäle in Unica Interact, um alle Objekte, Daten und Serverressourcen zu koordinieren, die am interaktiven Marketing beteiligt sind.

Ein interaktiver Kanal ist die Darstellung eines Touchpoints in Unica Interact, wobei die Benutzeroberflächenmethode ein interaktiver Dialog ist. Diese Softwaredarstellung wird zum Koordinieren aller Objekte, Daten und Serverressourcen verwendet, die mit dem interaktiven Marketing verbunden sind.

Ein interaktiver Kanal ist ein Tool, das Sie zum Definieren von Interaktionspunkten und Ereignissen verwenden. Über die Registerkarte "Analyse" eines interaktiven Kanals können Sie außerdem auf Berichte für diesen interaktiven Kanal zugreifen.

Interaktive Kanäle enthalten zudem Produktionslaufzeit- und Staging-Serverzuordnungen. Sie können mehrere interaktive Kanäle erstellen, um die Ereignisse und Interaktionspunkte zu gliedern, wenn Sie über nur einen Satz von Produktionslaufzeit- und Staging-Servern verfügen, oder um die Ereignisse und Interaktionspunkte nach kundenorientierten Systemen zu unterteilen.

Interaktive Ablaufdiagramme

Verwenden Sie interaktive Ablaufdiagramme, um die Kunden in Segmente zu unterteilen und den Segmenten jeweils ein Profil zuzuweisen.

Ein interaktives Ablaufdiagramm ist einem Unica Campaign-Ablaufdiagramm zur Stapelverarbeitung ähnlich, unterscheidet sich aber ein wenig von diesem. Interaktive Ablaufdiagramme haben im Wesentlichen dieselbe Funktion wie Ablaufdiagramme zur Stapelverarbeitung: Sie teilen die Kunden in Gruppen auf, die als Segmente bezeichnet werden. Bei interaktiven Ablaufdiagrammen sind die Gruppen jedoch intelligente Segmente. Unica Interact verwendet diese interaktiven Ablaufdiagramme, um einem Segment ein Profil zuzuordnen, wenn ein Verhaltens- oder Systemereignis anzeigt, dass eine Neusegmentierung der Besucher erforderlich ist.

Interaktive Ablaufdiagramme enthalten ein Subset der Prozesse der Ablaufdiagramme zur Stapelverarbeitung sowie einige für interaktive Ablaufdiagramme spezifische Prozesse. Die Option "Aktualisieren" ist in interaktiven Ablaufdiagrammen nicht verfügbar.

Anmerkung: Interaktive Ablaufdiagramme können nur während einer Unica Campaign-Sitzung erstellt werden.

Anmerkung: Für den Testlauf des interaktiven Ablaufdiagramms wird empfohlen, die Servergruppe statt der Produktionsservergruppe zu verwenden.

Anmerkung: Die Formate DT_DELIM_XXX können nicht mit Ablaufdiagrammen für interaktive Sitzungen verwendet werden.

Interaktionspunkte

Ein Interaktionspunkt ist ein Ort im Touchpoint, an dem Sie ein Angebot anzeigen möchten.

Interaktionspunkte verfügen über Standardinhalt, der angezeigt wird, falls die Laufzeitumgebung keinen anderen passenden Inhalt bereitstellen kann. Interaktionspunkte können in Zonen gegliedert werden.

Angebote

Ein Angebot stellt eine einzelne Marketingnachricht dar, die über unterschiedliche Kanäle übermittelt werden kann.

In Unica Campaign erstellte Angebote können in einer oder mehreren Kampagnen verwendet werden.

Angebote können wie folgt wiederverwendet werden:

- In verschiedenen Kampagnen
- Zu verschiedenen Zeitpunkten
- Für verschiedene Personengruppen (Zellen)
- Als unterschiedliche "Versionen", bei denen sich die Felder mit Parameterangabe des Angebots unterscheiden

Sie weisen Angeboten Interaktionspunkte in den Touchpoints zu, die Besuchern präsentiert werden.

Die Zustände "ENTWURF", "VERÖFFENTLICHT" und "ZURÜCKGEZOGEN" von Centralized Offer Management werden von Interact unterstützt. Angebote mit dem Status "VERÖFFENTLICHT" können in Interact Angebote verwendet und eingesetzt werden. Falls ein Angebot mit dem Status "VERÖFFENTLICHT" in Interact neu erstellt oder zurückgezogen wird, wird der entsprechende Status "(neu erstellt)/(zurückgezogen)" mit dem Angebot angezeigt.

Note: Jedes in Interact verwendete Angebot muss einen eindeutigen Angebotscode haben. Die Groß- und Kleinschreibung wird nicht beachtet. Außerdem muss jedes Angebotsattribut einen eindeutigen Namen haben. Die Groß- und Kleinschreibung wird nicht beachtet.

Profile

Als Profil bezeichnet man den Satz Kundendaten, den die Laufzeitumgebung verwendet. Diese Daten können ein Subset der in der Kundendatenbank enthaltenen Kundendaten sein oder in Echtzeit erfasste Daten bzw. eine Kombination aus beidem.

Die Kundendaten werden zu folgenden Zwecken verwendet:

- Um einen Kunden in Echtzeitinteraktionsszenarien einem oder mehreren Smart Segments zuzuordnen.

  Sie benötigen einen Satz Profildaten für jede Zielgruppenebene, nach der Sie segmentieren möchten. Ein Beispiel: Um nach "Ort" zu segmentieren, können Sie aus den gesamten Adressdaten des Kunden nur die Postleitzahl hinzufügen.
- Um Angebote zu personalisieren
- Als Attribute, die zu Lernzwecken verfolgt werden sollen

  Beispiel: Sie können Unica Interact so konfigurieren, dass es den Familienstand eines Besuchers nachverfolgt und erfasst, wie viele Besucher der einzelnen Familienstände bestimmte Angebote annehmen. Die Laufzeitumgebung kann anhand dieser Informationen dann die Angebotsauswahl optimieren.

Diese Daten sind für die Laufzeitumgebung schreibgeschützt.

Laufzeitumgebung

Die Laufzeitumgebung stellt eine Verbindung zu Ihrem Touchpoint her und führt Interaktionen aus. Die Laufzeitumgebung kann aus einem oder mehreren Laufzeitservern mit Verbindung zu einem Touchpoint bestehen.

Die Laufzeitumgebung verwendet die von der Designumgebung bereitgestellten Informationen zusammen mit der Unica Interact-API, um Angebote für den Touchpoint anzuzeigen.

Laufzeitsitzungen

Für jeden Besucher Ihres Touchpoints existiert eine Laufzeitsitzung auf dem Server für die Laufzeitumgebung. Diese Sitzung enthält alle Besucherdaten, die die Laufzeitumgebung verwendet, um die Besucher Segmenten zuzuordnen und Angebote zu empfehlen.

Sie erstellen eine Laufzeitsitzung mit dem Aufruf `startSession`.

Touchpoints

Ein Touchpoint ist eine Anwendung, in der bzw. ein Ort, an dem eine Interaktion mit einem Kunden erfolgt. Ein Touchpoint kann ein Kanal sein, in dem der Kunde den Kontakt einleitet (eine "Inbound"-Interaktion) oder in dem Sie den Kunden kontaktieren (eine "Outbound"-Interaktion).

Häufige Beispiele sind Websites und Call Center-Anwendungen. Mithilfe der Unica Interact-API können Sie Unica Interact in Ihre Touchpoints integrieren, um dem Kunden basierend auf seiner Aktion im Touchpoint Angebote anzuzeigen. Touchpoints werden auch als kundenorientierte Systeme (CFS, Client Facing Systems) bezeichnet.

Strategie und Verfahrensregeln

Ein interaktiver Kanal kann mehrere Marketingstrategien haben. Eine Strategie, ein zentraler Punkt der Interact-Anwendung, besteht aus einer Reihe von Behandlungsregeln. Ab Version 12.0 werden Verfahrensregeln in Interact auch als Smart-Regeln bezeichnet. Verfahrensregeln ordnen ein Angebot einem Smart Segment zu. Diese Zuordnungen werden durch die benutzerdefinierte Zone, die Sie dem Angebot in der Verfahrensregel zuweisen, weiter eingeschränkt.

So können Sie zum Beispiel über eine Auswahl von Angeboten verfügen, die Sie einem Smart Segment in der Zone "Anmeldung" zuordnen, sowie über eine weitere Auswahl von Angeboten für dasselbe Segment in der Zone "Nach dem Kauf".

Jede Verfahrensregel verfügt außerdem über einen Marketing-Score. Wenn ein Kunde mehreren Segmenten zugeordnet ist und daher mehr als ein Angebot in Frage kommt, wird mithilfe des Marketing-Score ermittelt, welches Angebot von Interact vorgeschlagen wird. Die Angebot-Score kann entweder einen statische Zahlen-Score haben, der als

Marketer-Score bezeichnet wird, oder einen dynamischen Score, der als Ausdruck (auch als Prädikat in Interaktion bezeichnet) des Profils oder der Angebotsattribute definiert wird. Interact Runtime verwendet diesen Ausdruck, um die Angebotsbewertung basierend auf Attributwerten zur Laufzeit zu berechnen. Mit der Angebotsberechtigung können Sie weiter feststellen, ob ein Angebot berechtigt ist oder nicht. Ein Angebot ist nur zulässig, wenn es in den Zeitraum des Inkrafttretens (zwischen dem Datum des Inkrafttretens und des Ablaufdatums) fällt und / oder ein Ausdruck zur Laufzeit als wahr bewertet wird. Wenn Interact Endbenutzern ein Angebot vorlegt, anstatt die Werte der Angebotsattribute aus dem Angebot zu übernehmen, kann Interact die Attributwerte des Angebots überschreiben, selbst wenn ein Ausdruck zur Interact-Laufzeit aus Profildaten berechnet wird. Benutzer können parametrisierte Angebotsattribute für das Angebot in einer Behandlungsregel definieren. Die von der Laufzeitumgebung vorgeschlagenen Angebote können von einem Lernmodul, einer Liste für Angebotsunterdrückung sowie globalen und individuellen Angebotszuweisungen beeinflusst werden.

FlexOffers

Ein interaktiver Kanal kann so konfiguriert werden, dass er über mehrere FlexOffer-Zuordnungen verfügt. FlexOffers stellt eine einfachere Möglichkeit bereit, Angebote direkt den übereinstimmenden, ziegerichteten Kunden zuzuordnen. Die Zuordnung von FlexOffers kann aus einer bereits erstellten Tabelle oder durch das Importieren einer CSV-Datei mit den erforderlichen Zuordnungsdaten oder durch die Erstellung einer neuen Regeltabelle erstellt werden.

Jede Zuordnung kann mehrere Regeln und Filter aufweisen. Jede Regel kann zum Zuordnen von Angeboten auf der Basis verschiedener benutzerdefinierter Attribute verwendet werden. Diese Zuordnungen können durch die benutzerdefinierte Zonen und Zellen, die dem Angebot in der Regel zugeordnet sind, weiter eingeschränkt werden. Außerdem kann eine Regel so festgelegt werden, dass ihr eine beliebige Anzahl von benutzerdefinierten Attributen zugeordnet werden kann.

Jede Regel verfügt außerdem über eine Marketingbewertung. Wenn ein Kunde für mehr als ein Angebot anwendbar ist, hilft die Marketingbewertung dabei, das Angebot zu ermitteln, das die Interact-Anwendung vorschlägt. Die Marketingbewertung kann einen statischen Wert oder einen dynamischen Wert aufweisen, der als Ausdruck von Angebotsattributen

definiert ist. Dieser Ausdruck wird dann zum Berechnen der Marketingbewertung nach der Interact-Laufzeit eingesetzt.

Mit der Angebotsberechtigung können Sie weiter feststellen, ob ein Angebot berechtigt ist oder nicht, selbst dann, wenn die Regel aktiviert ist. Ein Angebot ist zulässig, wenn es in den Zeitraum des Inkrafttretens (zwischen dem Datum des Inkrafttretens und des Ablaufdatums) fällt und/oder ein Ausdruck zur Laufzeit als wahr bewertet wird. Wenn Interact Endbenutzern ein Angebot vorlegt, anstatt die Werte der Angebotsattribute aus dem Angebot zu übernehmen, kann Interact die Attributwerte des Angebots überschreiben, selbst wenn ein Ausdruck zur Interact-Laufzeit aus Profildaten berechnet wird. Benutzer können parametrisierte Angebotsattribute für das Angebot in der FlexOffers-Regel definieren. Die von der Laufzeitumgebung vorgeschlagenen Angebote können von einem Lernmodul, einer Liste für Angebotsunterdrückung sowie globalen und individuellen Angebotszuweisungen beeinflusst werden.

Filter können auf die Regeln angewendet werden, um die erforderlichen Angebote für die Zielkunden zu erhalten. Für jeden Filter sind Bedingungen für die Regelattribute enthalten. Diese Bedingungen während der Laufzeit bestimmen die Angebotsumgebungsgruppe, die dem Kunden angezeigt wird. Eine beliebige Anzahl von Filtern kann gemeinsam auf die Regeln angewendet werden, um die erforderlichen Angebote zu erhalten.

Beispielsweise können Sie mehrere Regeln haben, die unterschiedliche Angebote aufweisen, die mit Attributen wie dem Standort und dem Gesamtaufwand eines Kunden verknüpft sind. Sie können Filter erstellen, die Bedingungen für diese Attribute haben. Abhängig von diesen Bedingungen werden die Angebote während der Laufzeit für den Kunden angezeigt.

Die Regeln und Filter werden auf der Registerkarte FlexOffers eines interaktiven Kanals definiert.

Unter der Registerkarte 'FlexOffers' können Sie die Zuordnung erstellen, die gewünschte Servergruppe kopieren, Regeln direkt erstellen oder aus einer Datei importieren, neue Regeln und Kriterien hinzufügen, einzelne oder mehrere Regeln und Kriterien bearbeiten oder löschen, Regeln kopieren und Filter erstellen. Details hierzu finden Sie im Interact-Benutzerhandbuch.

Die FlexOffer-Zuordnung zusammen mit ihren Regeln und Filtern stellt eine Lösung zur Anpassung von Angeboten auf der Basis einer beliebigen Anzahl benutzerdefinierter Attribute und zum Abrufen dieser Angebote durch die Anwendung unterschiedlicher Bedingungen für diese Attribute bereit.

Beim Abrufen von Angeboten aus der Interact-Laufzeit werden die Filter gemäß der folgenden Logik angewendet:

UACIenabLEOfferMappingFilter: Dieser Parameter wird zusammen mit dem Filternamen während der Laufzeit während StartSession oder getOffers definiert. Der jeweilige Filter wird angewendet, um Angebote aus der FlexOffer-Zuordnungstabelle abzurufen.

UACIDisabLEOfferMappingFilter: Dieser Parameter wird zusammen mit dem Filternamen während der Laufzeit während StartSession oder getOffers definiert. Der jeweilige Filter wird nicht angewendet, wenn Angebote aus der Tabelle abgerufen werden.

Abgesehen davon werden alle Filter, die als Standard markiert sind, wenn sie nicht inaktiviert sind, auf die Tabelle angewendet, um Angebote zu erhalten.

Gateways

Eingehende und ausgehende Gateways werden von Interact unterstützt. Alle Konfigurationen wurden jedoch über Eigenschaftsdateien vorgenommen, die schwer zu verwalten und fehleranfällig sind.

Ein interaktiver Kanal kann konfiguriert werden, indem Sie mehrere Gateway-Zuordnungen definieren können.

Sie können Gateway-Zuordnungen für Folgendes erstellen.

- Ausgehende Journey
- Ausgehende Deliver
- Allgemein-Ausgehend
- Allgemein-Eingehend

Für weitere Informationen über Ausgehende Journey, siehe Abschnitt Integration von Unica Interact und Unica Journey (auf Seite 718).

Für weitere Informationen über Ausgehende Deliver, siehe Abschnitt Integration von Unica Interact und Unica Deliver (auf Seite 724).

Allgemein-Ausgehend: Es kann verwendet werden, um die Zuordnungen für die Gateways zu definieren, die für die ausgehende Kommunikation konfiguriert sind.

- Anzahl der Nachrichten: Definiert die Anzahl der Nachrichten, die über das Gateway gesendet werden
- Priorität: Definiert die Priorität des Gateways. Dies ist ein Zahlenwert. Der Zusammenhang von Name und Priorität ist eine eindeutige Identifizierung von dem Gateway innerhalb des umschließenden interaktiven Kanals. Gateway mit dem niedrigsten Prioritätswert wird zur Interact-Laufzeit bereitgestellt.
- Kanaleigenschaften – Dies wird verwendet, um die Eigenschaften im Format vom Schlüsselwert zu definieren, die für das Gateway benötigt werden.
- Zuordnung – Dies wird verwendet, um die Zuordnung zwischen dem Endpunktfeld und dem Interact-Feld zu definieren.

Allgemein-Eingehend: Es kann verwendet werden, um die Zuordnungen für die Gateways zu definieren, die für die eingehende Kommunikation konfiguriert sind.

- Priorität: Definiert die Priorität des Gateways. Dies ist ein Zahlenwert. Der Zusammenhang von Name und Priorität ist eine eindeutige Identifizierung von dem Gateway innerhalb des umschließenden interaktiven Kanals. Gateway mit dem niedrigsten Prioritätswert wird zur Interact-Laufzeit bereitgestellt.
- Kanaleigenschaften - Dies wird verwendet, um die Eigenschaften im Format vom Schlüsselwert zu definieren, die für das Gateway benötigt werden. Diese Eigenschaften werden als Parameter an die startSession-API übergeben.
- Zuordnung – Dies wird verwendet, um die Parameterzuordnung zwischen dem Interact-Ereignis und dem Endpunktereignis zu definieren. Diese Eigenschaften werden als Parameter an die postEvent-API übergeben.

Für weitere Informationen über die Konfiguration von Zuordnungen für E-Mail, MobilePush und UBX Gateway, siehe Abschnitt Gateways konfigurieren (auf Seite 728).

Die Gateways 'Allgemein - Eingehend' oder 'Allgemein - Ausgehend' implementieren

Interact bietet eine out-of-the-box Implementierung für allgemeine eingehende- und ausgehende Gateways. Die Benutzer können die Kanaleigenschaften und die Zuordnung mit dem allgemein ausgehenden- oder eingehenden Gateway konfigurieren und diese Interact Implementierungen verwenden, indem sie Gateways mit der Vorlage von InteractGateway in der Interact Laufzeit-Konfiguration `TriggeredMessage/activityOrchestration` erstellen.

Beide Implementierungen unterstützen Kafka als der Kommunikationskanal zwischen Interact und Drittsystemen.

Implementierung generischer eingehender Gateway

Um die eingehende Nachricht zu verarbeiten, benötigt Interact Informationen in Form einer Konfiguration, um die Felder der eingehenden Nachricht den entsprechenden Interact-Eigenschaften zuzuordnen. Die UI-Konfiguration muss alle obligatorischen Eigenschaften definieren, die Interact benötigt, um Interact-APIs aufrufen zu können: startSession, postEvent und endSession. Sie können die Konfiguration ausführen, indem Sie zu "Interaktiver Kanal" -> "Gateways" -> "Generischer eingehender Gateway" navigieren.

Im Folgenden finden Sie ein Beispiel für eine eingehende Nachricht über den Kafka-Kanal. Der Kafka-Aktivitätsempfänger erfiebiert die Eigenschaften "**gateway**" und "**message**", um den Kanal für die Verarbeitung zu identifizieren. "message" enthält die Informationen, die zur Verarbeitung an das eingehende Gateway gesendet werden.

```
{
  "gateway": "GenericIn",
  "message":
{
    "ICName": "SB_InteractiveChannel",
    "audienceID": [
      {
        "n": "CustomerID",
        "v": "1",
```

```
        "t": "numeric"
      }
    ],
    "events": [
      {
        "event": "EP_contact",
        "parameters": [
          {
            "n": "UACIOfferTrackingCode",
            "v": "5.2.ffffffffe4699811.4fad551",
            "t": "string"
          }
        ]
      }
    ],
    "parameters": [
      {
        "n": "country",
        "v": "INDIA",
        "t": "string"
      },
      {
        "n": "UACILogSeparationFileName",
        "v": "log123",
        "t": "string"
      }
    ],
    "CH_debug": "true",
    "CH_sessionID": "session1"
  }
}
```

Konfiguration eines eingehenden Gateways

Die Konfiguration muss eine Zuordnung der Eigenschaften eingehender Nachrichten mit den API-Parametern startSession, postEvent und endSession bereitstellen.

startSession-Eigenschaften

Interact benötigt die folgenden Informationen für die startSession -API.

- SessionId
- Name des interaktiven Kanals
- Zielgruppenebene
- Zielgruppen-ID
- Parameter
- Vorhandene Sitzung als Grundlage
- Debugging

Beachten Sie dabei die folgenden Punkte:

- Der Name des interaktiven Kanals muss über den reservierten Eigenschaftsnamen "ICName" bereitgestellt werden. Dies ist für alle eingehenden Nachrichten obligatorisch. Zum Beispiel:

  ICName: SB_InteractiveChannel
- Zielgruppenebene, Zielgruppen-ID und "Rely on existing session" werden auf der Registerkarte "Allgemein" der Konfiguration für generische eingehende Gateway zugeordnet.
- Bei der Zielgruppen-ID können Benutzer eine beliebige Endpunkteigenschaft konfigurieren, die zugeordnet und von der eingehenden Nachricht verarbeitet werden soll. Der Wert der Zielgruppen-ID muss dem unten vordefinierten Format folgen.

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

```
      }
    ]
```

- Die Zuordnung für Sitzungs-ID und Debug-Flags kann über die Registerkarte "Kanaleigenschaften" des generischen Eingehenden konfiguriert werden.
  Alle auf der Registerkarte Kanaleigenschaften definierten Eigenschaften werden zusätzlich als startSession -Parameter übergeben, sodass Benutzer auf diese Registerkarte zugreifen können, um Parameter für das Starten von Sitzungen zu definieren. Außerdem können Benutzer die startSession -Parameter wie unten dargestellt unter dem reservierten Eigenschaftsnamen "parameters" übergeben.

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

postEvent-Eigenschaften

Ereignisname und Ereignisparameter können über die Registerkarte "Zuordnung" in der generischen eingehenden Konfiguration zugeordnet werden.

Alle Ereignisparameterzuordnungen werden als Ereignisparameter für den postEvent-API-Aufruf übergeben. Darüber hinaus können Benutzer optional den reservierten Eigenschaftsnamen "parameters" unter "events" verwenden, um die Ereignisparameter wie unten dargestellt zu definieren.

```
"events": [
      {
        "event": "EP_contact",
        "parameters": [
          {
            "n": "UACIOfferTrackingCode",
            "v": "5.2.ffffffffe4699811.4fad551",
            "t": "string"
          }
        ]
      }
    ]
```

endSession -Eigenschaften

Die Eigenschaft "Sitzung beenden" wird über die Registerkarte "Allgemein" zugeordnet. Benutzer können diese Konfiguration auch überschreiben, indem sie einen Wert für die reservierte Eigenschaft "endSession" in der eingehenden Nachricht bereitstellen.

Reservierte Feldnamen für eingehende Gateways

Im Folgenden sind die reservierten Feldnamen angegeben, die das System versucht, in der eingehenden Nachricht zu suchen, wenn es diese entweder über die Gateway-Konfiguration nicht finden kann oder wenn Benutzer sie überschreiben möchten.

Im Folgenden sind die reservierten Felder enthalten, nach denen das System sucht, falls es die Zuordnung in der Gateway-Konfiguration nicht findet.

- ICName
- Parameter
- Debugging
- sessionId

Im Folgenden sind die Felder angegeben, für die Benutzer den Konfigurationswert überschreiben können, indem sie sie in der eingehenden Nachricht bereitstellen.

- endSession
- relyOnExistingSession

Konfiguration der Laufzeit

Erstellen Sie mithilfe von Affinium|interact|activityOrchestrator|gateways ein Gateway|(InteractGateway)-Vorlage. Der Name des Gateways muss mit dem Gateway-Namen übereinstimmen, der für das generische Eingangs-Gateway angegeben wurde, das zur Designzeit von Interact erstellt wurde. Für den Empfang der eingehenden Nachricht müssen Sie einen Empfänger vom Typ "Kafka" aus "Affinium|interact|activityOrchestrator|receivers" erstellen und die erforderlichen Parameter für die Verbindung zu Kafka als Verbraucher hinzufügen.

Implementierung generischer ausgehender Gateways

Die Konfiguration des ausgehenden Gateways wird verwendet, um die Informationen zu identifizieren, die als Teil der abgehenden Nachricht gesendet werden müssen. Diese out-of-the-box-Implementierung für ein generisches ausgehendes Gateway gilt speziell für die Kafka-Gateways.

Im Folgenden finden Sie eine Beispielnachricht für abgehende Nachrichten, die von der Implementierung des ausgehenden Gateways erstellt wird:

```
{
  "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"
}
```

Vorgegebene Felder

Im Folgenden finden Sie die Felder, die standardmäßig Teil jeder abgehenden Nachricht sind.

- -Gateway
- Kanal
- Interaktiver Kanal
- Verarbeitungszeitmarke
- Zielgruppenebene
- Zielgruppen-ID
- OfferName
- OfferCode
- TreatmentCode

Angebots- und Profilattribute

Die Angebots- und Profileigenschaften, die als Teil der abgehenden Nachricht gesendet werden müssen, können über die Registerkarte "Zuordnung" konfiguriert werden. Die Werte werden bei entsprechender Konfiguration auf Größe und Datumsformat überprüft. Ebenso

werden die als obligatorisch konfigurierten Felder validiert, bevor die Nachrichten gesendet werden können.

Endpunktfelder mit Standardwerten können über die Registerkarte "Kanaleigenschaften" konfiguriert werden. Alle auf dieser Registerkarte konfigurierten Felder werden als Teil jeder abgehenden Nachricht gesendet.

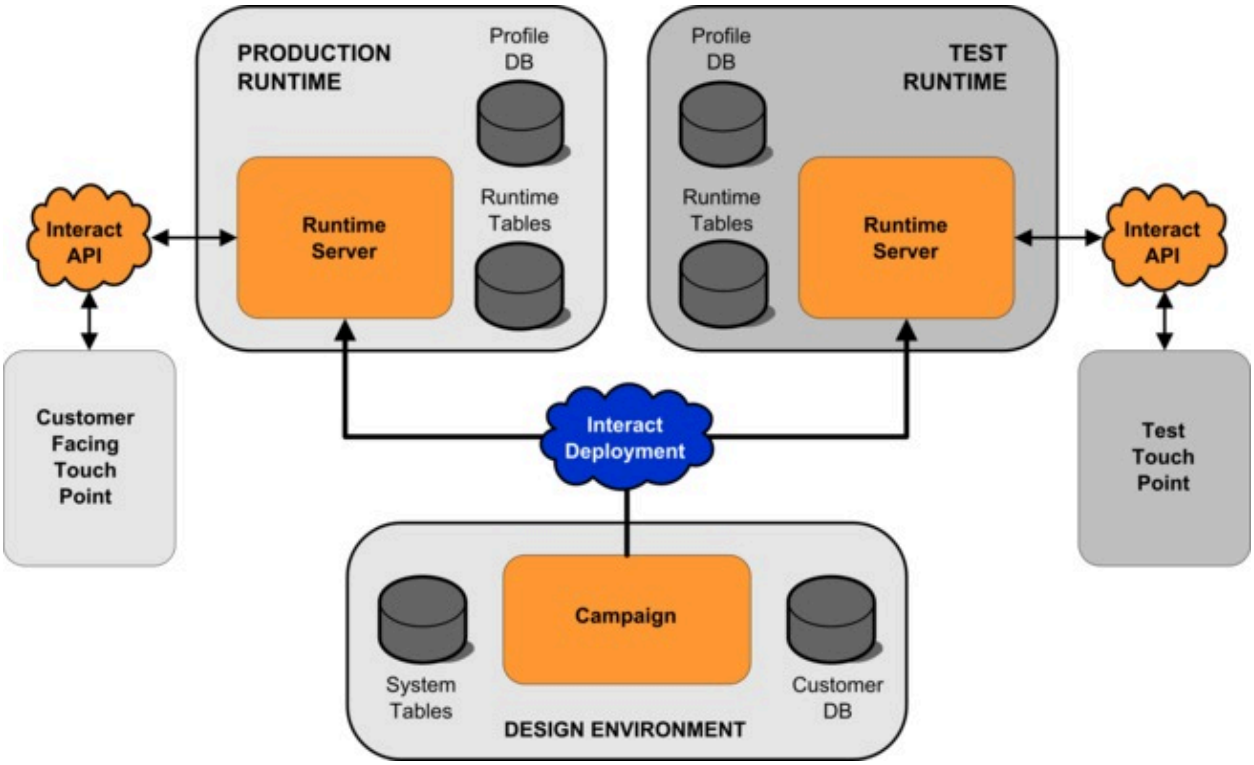
Konfiguration der Laufzeit

Benutzer müssen ein Gateway konfigurieren, indem sie zu Affinium|interact|triggeredMessage|gateways|InteractGateway" navigieren. Der Name dieses Gateways muss mit dem generischen ausgehenden Gateway übereinstimmen, das über die Benutzerschnittstelle erstellt wurde. Die Vorlage erfordert, dass die Benutzer Kafka-Verbindungsdetails zum Senden der Nachricht bereitstellen.

Unica Interact Architektur

Die Unica Interact-Umgebung besteht aus mindestens zwei wichtigen Komponenten, der Designumgebung und der Laufzeitumgebung. Möglicherweise setzen sie außerdem optionale Server zum Testen der Laufzeitumgebung ein.

Die folgende Abbildung zeigt einen allgemeinen Überblick über die Architektur.



In der Designumgebung führen Sie den größten Teil der Unica Interact-Konfiguration aus. Die Designumgebung wird mit derUnica Campaign- Webanwendung installiert und verweist auf die Unica Campaign-Systemtabellen und Ihre Kundendatenbanken. Sie verwenden die Designumgebung, um die Interaktionspunkte und Ereignisse zu definieren, die Sie mit der API verwenden.

Nachdem Sie festgelegt und konfiguriert haben, wie Kundeninteraktionen von der Laufzeitumgebung verarbeitet werden sollen, stellen Sie diese Daten entweder einer Staging-Servergruppe zum Testen oder einer Servergruppe für die Produktionsumgebung für Kundeninteraktionen in Echtzeit bereit.

Die Unica Interact-API stellt die Verbindung zwischen Ihrem Touchpoint und dem Laufzeitserver bereit. Sie referenzieren Objekte (Interaktionspunkte und Ereignisse), die in der Designumgebung erstellt werden, mit der Unica Interact-API und verwenden sie, um Informationen aus dem Laufzeitserver anzufordern.

Überlegungen zum Unica Interact-Netz

Eine Produktionsinstallation von Unica Interact umfasst zumindest zwei Maschinen. In einer hochvolumigen Produktionsumgebung mit mehreren Unica Interact-Laufzeitservern und verteilten Datenbanken kann Ihre Installation Dutzende Maschinen umfassen.

Um die beste Leistung zu erhalten, sind mehrere Netztopologie-Anforderungen zu berücksichtigen.

- Wenn Ihre Implementierung der Unica Interact-API Sitzungen im selben Aufruf startet und beendet, zum Beispiel:

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

  müssen Sie keine Sitzungspersistenz (permanente Sitzungen) zwischen der Lastausgleichsfunktion und den Unica Interact-Laufzeitservern aktivieren. Sie können das Unica Interact-Laufzeitserversitzungsmanagement für den lokalen Cachetyp konfigurieren.
- Wenn Ihre Implementierung der Unica Interact-API mehrere Aufrufe verwendet, um Sitzungen zu starten und zu beenden, zum Beispiel:

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

  und Sie ein Programm für den Lastausgleich für Ihre Unica Interact-Laufzeitserver verwenden, sollten Sie eine Art von Persistenz für die Lastausgleichsfunktion ermöglichen (auch permanente Sitzungen genannt). Wenn das nicht möglich ist oder wenn Sie keine Lastausgleichsfunktion verwenden, konfigurieren Sie das Unica Interact-Serversitzungsmanagement für einen verteilten `cacheType`. Wenn Sie einen verteilten Cache verwenden, müssen alle Unica Interact-Laufzeitserver in der Lage sein, über Multicasting zu kommunizieren. Sie müssen möglicherweise Ihr Netz optimieren, damit die Kommunikation zwischen Unica Interact-Servern, die dieselbe Multicast-IP-Adresse und Port verwenden, nicht die Systemleistung behindert. Eine

Lastausgleichsfunktion mit permanenten Sitzungen bietet eine bessere Leistung als die Verwendung eines verteilten Cache.
- Verteiltes Caching unter mehreren Servergruppen wird nicht unterstützt.
- Die beste Leistung erhalten Sie, wenn sich Ihre Laufzeitumgebung (Unica Interact-Server, Unica Platform-Lastenausgleichsprogramme und Touchpoint) an einem Standort befindet. Die Designumgebung und die Laufzeitumgebung können an verschiedenen Standorten sein, was aber zu einer langsamen Implementierung führen kann.
- Richten Sie eine schnelle Netzverbindung (mindestens 1 Gb) zwischen der Unica Interact-Produktionsservergruppe und ihrem zugehörigen Touchpoint ein.
- Die Designumgebung erfordert http- oder https-Zugriff auf die Laufzeit, um Implementierungsaufgaben durchzuführen. Firewalls oder andere Netzanwendungen müssen so konfiguriert sein, dass sie die Implementierung ermöglichen. Sie müssen möglicherweise die Dauer der HTTP-Zeitlimits zwischen der Designumgebung und den Laufzeitumgebungen verlängern, wenn Sie umfangreiche Implementierungen haben.
- Das Kontakt- und Antwortverlaufsmodul erfordert Zugriff auf die Designzeitdatenbank (Unica Campaign-Systemtabellen) und Zugriff auf die Laufzeitdatenbank (Unica Interact-Laufzeittabellen). Sie müssen Ihre Datenbank und Ihr Netz entsprechend konfigurieren, damit diese Datenübertragung stattfinden kann.

In einer Test- oder Staging-Installation können Sie die Unica Interact-Designumgebung und die Laufzeitumgebung auf derselben Maschine installieren. Dieses Szenario wird nicht für Produktionsumgebungen empfohlen.

Unica Interact Serverports und Netzwerksicherheit

Konfigurieren Sie Unica Interact, um Ihre Serverports zu sichern.

Unica Interact Laufzeitports

Einige dieser Ports können geschlossen werden oder werden abhängig von Ihrer Konfiguration nicht von allen Unica Interact-Installationen benötigt.

Unica Interact-Anwendungsserver-Port für HTTP

Der Standardport, an dem Unica Interact-Anfragen bearbeitet werden.

Unica Interact-Anwendungsserver-Port für HTTPS

Der Standard-SSL-Port, an dem Unica Interact-Anfragen bearbeitet werden.

Unica Interact systemTablesDataSource-Port

Informationen finden Sie in der JDBC-Konfiguration der Datenquelle in Unica Platform.

Unica Interact learningTablesDataSource-Port

Informationen finden Sie in der JDBC-Konfiguration der Datenquelle in Unica Platform.

Unica Interact contactAndResponseHistoryDataSource-Port

Informationen finden Sie in der JDBC-Konfiguration der Datenquelle in Unica Platform.

Unica Interact prodUserDataSource-Port

Informationen finden Sie in der JDBC-Konfiguration der Datenquelle in Unica Platform.

Unica Interact testRunDataSource-Port

Informationen finden Sie in der JDBC-Konfiguration der Datenquelle in Unica Platform.

ETL-Kommunikationsport

Konfigurieren Sie diesen Port in **Unica Interact | ETL | patternStateETL | communicationPort** in den Konfigurationseigenschaften.

EHCache-Multicast-Port

Konfigurieren Sie diesen Port in **Unica Interact | cacheManagement | Cache | Managers | EHCache | Parameter Data | multicastPort** in den Konfigurationseigenschaften, wenn der Cachemodus verteilt wird.

Unica Interact JMX-Überwachungsport

Konfigurieren Sie diesen Port in **Unica Interact | Überwachung | port** in den Konfigurationseigenschaften oder führen Sie `-Dinteract.jmx.monitoring.port=portNumber`.

Unica Interact WebConnector-Port

Dieser Port stimmt normalerweise mit dem Unica Interact-Server-Port überein, er kann jedoch in `jsconnector.xml`geändert werden.

Informationen zu den Ports für integrierte Unica Interact-Produkte finden Sie in der Dokumentation zu diesen Produkten.

JMX-Überwachung ist für die normaleUnica Interact-Funktionalität nicht erforderlich. Sie wird jedoch für die Diagnose und Überwachung verwendet.

JMX-Portzugriff kann in der Unica Interact-Konfiguration inaktiviert oder auf bestimmte IP-Adressen mithilfe der Firewallkonfiguration beschränkt werden. Dies wird wegen der JMX-Anfälligkeit empfohlen, die kürzlich in der Apache Commons Library, einem Drittanbieter, erkannt wurde.

Die JMX-Remoting-Funktion in Apache Geronimo 3.x vor 3.0.1, die in IBM WebSphere Application Server (WAS) Community Edition 3.0.0.3 und anderen Produkten verwendet wird, implementiert das RMI-Klassenladeprogramm nicht ordnungsgemäß, wodurch Remote-Angreifer beliebigen Code ausführen können, um mithilfe des JMX-Connectors ein präpariertes serialisiertes Objekt zu senden.

Unica Interact Designports

Einige dieser Ports können geschlossen werden oder werden abhängig von Ihrer Konfiguration nicht von allen Unica Interact-Installationen benötigt.

Unica Campaign-Anwendungsserver-Port für HTTP

Der Standardport, an dem Unica Interact-Anfragen bearbeitet werden.

Unica Campaign-Anwendungsserver-Port für HTTPS

Der Standard-SSL-Port, an dem Unica Interact-Anfragen bearbeitet werden.

Unica Campaign Listener-Port

Der Port, den Unica Campaign intern verwendet, um Verbindungen vom Web-Client anzunehmen.

Weitere Unica Campaign Designports

Weitere Informationen zu diesen Ports finden Sie in der Campaign-Dokumentation.

Unica Campaign JMX- Connectorport

Konfigurieren Sie diesen Port in **Unica Campaign | Überwachung | port** in den Konfigurationseigenschaften nur für die Kontakt-/Antwortverlaufsüberwachung.

Unica Campaign betriebsbereiter Überwachungsserver-Port

Konfigurieren Sie diesen Port in **Unica Campaign | Überwachung | serverURL** in den Konfigurationseigenschaften.

Anmelden bei Unica

Verwenden Sie diese Prozedur, um sich bei Unica anzumelden.

Sie benötigen das Folgende.

- Eine Intranet-(Netz-)Verbindung, um auf den Unica-Server zuzugreifen.
- Einen auf dem Computer installierten Browser, der auch unterstützt wird.
- Benutzername und Kennwort, damit Sie sich bei Unica anmelden können.
- Die URL, um im Netz auf Unica zuzugreifen.

Die URL ist:

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

Wo

`host` ist das System, auf dem Unica Platform installiert ist.

`domain.com` ist die Domäne, in der sich die Hostmaschine befindet.

`port` ist die Portnummer, auf welcher der Unica Platform-Anwendungsserver empfangsbereit ist.

Anmerkung: Für das folgende Verfahren wird vorausgesetzt, dass Sie mit einem Konto angemeldet sind, das über Administratorzugriff für Unica Platform verfügt.

Greifen Sie über den Browser auf die Unica-URL zu.

- Falls Unica für die Integration mit Windows™ Active Directory oder mit einer Plattform zur Webzugriffssteuerung konfiguriert ist und Sie bei diesem System angemeldet sind, wird die Standarddashboardseite angezeigt. Ihre Anmeldung ist abgeschlossen.
- Wenn die Anmeldeanzeige angezeigt wird, melden Sie sich mit den Standardberechtigungsnachweisen für Administratoren an. Verwenden Sie in einer Umgebung mit nur einer Partition `asm_admin` mit `password` als Kennwort. Verwenden Sie in einer Umgebung mit mehreren Partitionen `platform_admin` mit `password` als Kennwort.

  Sie werden aufgefordert, das Kennwort zu ändern. Sie können das vorhandene Kennwort eingeben. Aus Sicherheitsgründen sollten Sie jedoch ein neues Kennwort verwenden.
- Falls Unica für die Verwendung mit SSL konfiguriert ist, werden Sie bei der erstmaligen Anmeldung eventuell aufgefordert, ein digitales Sicherheitszertifikat anzunehmen. Klicken Sie auf **Ja**, um das Zertifikat anzunehmen.

War die Anmeldung erfolgreich, zeigt Unica die Standarddashboardseite an.

Mit den Unica Platform-Administratorkonten zugeordneten Standardberechtigungen können Sie mithilfe der im Menü **Einstellungen** aufgeführten Optionen Benutzerkonten und Sicherheitsaspekte verwalten. Wenn Sie für Unica-Dashboards Administrationsaufgaben auf der höchsten Ebene ausführen möchten, müssen Sie sich als **platform_admin** anmelden.

Kapitel 2. Sicherheitsmanagement

Der Zugriff auf die Unica Interact Laufzeit-Benutzeroberfläche erfordert eine Authentifizierung. Nur die in Unica Platform definierten Anmelde-IDs, die die Rolle eines Unica Interact Administrators haben, können auf die Seiten zugreifen.

Authentifizierung der Unica Interact JSP-Seiten

Die Platform- oder LDAP-Benutzer mit InteractAdminRole oder InteractUserRole können sich bei Interact Runtime Interface anmelden. Der Benutzername und das Kennwort müssen mit der Unica Platformoder LDAP-Konfiguration angelegt werden. Sie werden beim Schließen des Browsers oder der Registerkarte abgemeldet. Ihr Benutzername wird deaktiviert, wenn Sie dreimal versuchen, sich mit falschen Anmeldedaten anzumelden. Wenn der Benutzername deaktiviert wird, müssen Sie ihn mit Unica Platform admin aktivieren. Diese Authentifizierung gilt nur für JSP-Seiten.

Kapitel 3. Konfigurieren von Benutzern

Für Unica Interact müssen zwei Benutzergruppen eingerichtet werden: Laufzeitumgebungsbenutzer und Designzeitumgebungsbenutzer.

- **Laufzeitbenutzer** werden in Unica Platform erstellt und für das Arbeiten mit den Laufzeitservern konfiguriert.
- **Designumgebungsbenutzer** sind Unica Campaign-Benutzer. Sie konfigurieren die Sicherheit für die verschiedenen Mitglieder Ihres Designteams wie für Unica Campaign.

Den Laufzeitumgebungsbenutzer konfigurieren

Nach der Installation von Unica Interact müssen Sie zumindest einen Unica Interact-Benutzer, den Laufzeitumgebungsbenutzer, konfigurieren. Laufzeitbenutzer werden in Unica Platform erstellt.

Der Laufzeitumgebungsbenutzer bietet Zugriff auf die Laufzeittabellen. Als Laufzeitumgebungsbenutzer werden der Benutzername und das Kennwort angegeben, mit denen Sie interaktive Kanäle bereitstellen. Der Laufzeitserver verwendet die Webanwendungsserver-JDBC-Authentifizierung für die Berechtigungsnachweise für die Datenbank. Sie müssen dem Laufzeitumgebungsbenutzer keine Datenquellen der Laufzeitumgebung hinzufügen.

Ein LDAP-Benutzer und jeder Platform-Benutzer kann einen interaktiven Kanal bereitstellen. Die InteractAdminRole ist zum Bereitstellen des interaktiven Kanals nicht erforderlich.

Beachten Sie bei der Erstellung von Laufzeitbenutzern Folgendes:

- Bei eigenen Unica Platform-Instanzen für jeden Laufzeitserver müssen Sie jeweils denselben Benutzer und dasselbe Kennwort erstellen. Für alle derselben Servergruppe angehörenden Laufzeitserver müssen dieselben Benutzerberechtigungsnachweise angegeben werden.
- Wenn Sie ein Datenbankladedienstprogramm verwenden, müssen Sie die Laufzeittabellen als eine Datenquelle mit Anmeldungsberechtigungsnachweisen für

die Laufzeitumgebung in den Konfigurationseigenschaften unter `Interact > general > systemTablesDataSource` definieren .

- Wenn Sie die Sicherheit für JMX-Überwachung mit dem JMXMP-Protokoll einrichten, ist eventuell ein eigener Benutzer für die Sicherheit der JMX-Überwachung erforderlich.

In der Dokumentation zu Unica Platform finden Sie die Schritte, die zur Erstellung von Laufzeitbenutzern ausgeführt werden müssen.

Interact-Laufzeit unterstützt Rollen und Berechtigungen zur Steuerung des Benutzerzugriffs auf Objekte und Funktionen in Interact-Laufzeit. Diese Rollen und Berechtigungen können in der Platform konfiguriert werden.

Im Folgenden sind die Interact-Laufzeitberechtigungen aufgeführt, die für globale Standardrichtlinien und neue Richtlinien gelten.

Kategorie	Genehmigungen	Syntax
Interact	Interact-Laufzeitstatus anzeigen	Prüft den Initialisierungsstatus, zeigt die Konfigurationsvalidierung und die Info-Seite an.
Interact	Ausführen von Interact-Laufzeit-APIs	Führt Interact-Laufzeit-APIs aus.
Interact	Anzeigen von Interact Admin-Links	Zeigt andere Admin-Seitenlinks wie JMX Sweep, Manager Configuration Sweep, Offer Constraint und Event Pattern-Status an.

Designumgebungsbenutzer konfigurieren

Designumgebungsbenutzer sind Unica Campaign-Benutzer. Sie konfigurieren Ihre Designumgebungsbenutzer auf die gleiche Weise, wie Sie Unica Campaign-Rollenberechtigungen konfigurieren.

Für bestimmte Designumgebungsbenutzer sind auch gewisse Unica Campaign-Berechtigungen wie z. B. benutzerdefinierte Makros erforderlich.

Beachten Sie bei der Erstellung von Designumgebungsbenutzern Folgendes:

- Wenn Sie über Unica Campaign-Benutzer verfügen, die zum Bearbeiten von interaktiven Ablaufdiagrammen berechtigt sind, müssen Sie ihnen den Zugriff auf die Testlauftabellendatenquelle erteilen.
- Wenn Unica Interact installiert und konfiguriert ist, sind die unten genannten Zusatzoptionen für die standardmäßige globale Richtlinie und neue Richtlinien verfügbar.

Kategorie	Genehmigungen
Kampagnen	• Anzeigen von Kampagneninteraktionsstrategien - Der Benutzer kann Registerkarten des Typs "Interaktionsstrategie" einer Kampagne anzeigen, aber nicht bearbeiten. • Bearbeiten von Kampagneninteraktionsstrategien - Der Benutzer kann die Registerkarten des Typs "Interaktionsstrategie" ändern, einschließlich Verfahrensregeln. • Löschen von Kampagneninteraktionsstrategien - Der Benutzer kann Registerkarten des Typs "Interaktionsstrategie" aus Kampagnen löschen. Das Löschen einer Registerkarte des Typs "Interaktionsstrategie" ist eingeschränkt, wenn die Interaktionsstrategie in eine Bereitstellung eines interaktiven Kanals aufgenommen wurde. • Hinzufügen von Kampagneninteraktionsstrategien - Der Benutzer kann neue Registerkarten des Typs "Interaktionsstrategie" für eine Kampagne erstellen.

Kategorie	Genehmigungen
	• Initiate® Bereitstellung von Kampagneninteraktionsstrategien zur Initialisierung - Der Benutzer kann eine Registerkarte des Typs Interaktionsstrategie zur Bereitstellung oder zum Zurücknehmen der Bereitstellung markieren.
Interaktive Kanäle	• Interaktive Kanäle bereitstellen - Der Benutzer kann einen interaktiven Kanal für die Unica Interact-Laufzeitumgebungen bereitstellen. • Interaktive Kanäle bearbeiten - Der Benutzer kann die Übersichtsregisterkarte von interaktiven Kanälen ändern. • Interaktive Kanäle löschen - Der Benutzer kann interaktive Kanäle löschen. Das Löschen von interaktiven Kanälen ist eingeschränkt, wenn der interaktive Kanal bereitgestellt worden ist. • Interaktive Kanäle anzeigen - Der Benutzer kann interaktive Kanäle anzeigen, aber nicht bearbeiten. • Interaktive Kanäle hinzufügen - Der Benutzer kann neue interaktive Kanäle hinzufügen. • Berichte zu interaktiven Kanälen anzeigen - Der Benutzer kann die Registerkarte "Analyse" des interaktiven Kanals anzeigen. • Untergeordnete Objekte zum interaktiven Kanal hinzufügen - Der Benutzer kann Interaktionspunkte, Zonen, Ereignisse und Kategorien hinzufügen. • Interaktionspunkte/Zonen hinzufügen - Der Benutzer kann Interaktionspunkte und Zonen, die unter dem interaktiven Kanal 'Interaktionspunkte' verfügbar sind, hinzufügen/bearbeiten/löschen. • Ereignisse/Muster hinzufügen - Der Benutzer kann Ereignisse/Muster hinzufügen/bearbeiten/löschen, die unter der Registerkarte 'Ereignisse' des interaktiven Kanals verfügbar sind.

Kategorie	Genehmigungen
	• Angebotsbeschränkungen hinzufügen - Der Benutzer kann Angebotsbeschränkungen hinzufügen/bearbeiten/löschen, die unter dem interaktiven Kanal 'Beschränkungen' verfügbar sind. • Selbstlernmodell hinzufügen - Der Benutzer kann ein Selbstlernmodell hinzufügen/bearbeiten/löschen, das unter dem interaktiven Kanal 'Selbstlernen' verfügbar ist. • Getriggerte Nachrichten hinzufügen - der Benutzer kann getriggerte Nachrichten hinzufügen/bearbeiten/speichern/löschen, die unter der Registerkarte 'Getriggerte Nachrichten' des interaktiven Kanals verfügbar sind. • Simulator-Szenarien hinzufügen - der Benutzer kann die Simulator-Szenarien hinzufügen/bearbeiten/ausführen/löschen, die unter der Registerkarte 'Simulator' des interaktiven Kanals vorhanden sind. • Angebotszuordnung anzeigen - Benutzer kann bestehende Angebotszuordnungen anzeigen. • Angebotszuordnung bearbeiten - Benutzer kann Angebotszuordnungen hinzufügen/bearbeiten/löschen. • Bereitstellung der Angebotszuordnung initiieren - der Benutzer kann die Angebotszuordnung für die Bereitstellung markieren.
Sitzungen	• Interaktive Ablaufdiagramme anzeigen - Der Benutzer kann ein interaktives Ablaufdiagramm in einer Sitzung anzeigen. • Interaktive Ablaufdiagramme hinzufügen - Der Benutzer kann neue interaktive Ablaufdiagramme in einer Sitzung erstellen. • Interaktive Ablaufdiagramme bearbeiten - Der Benutzer kann interaktive Ablaufdiagramme ändern.

Kategorie	Genehmigungen
	• Interaktive Ablaufdiagramme löschen - Der Benutzer kann interaktive Ablaufdiagramme löschen. Das Löschen von interaktiven Ablaufdiagrammen ist eingeschränkt, wenn der Kanal, dem dieses interaktive Ablaufdiagramm zugeordnet ist, bereitgestellt worden ist. • Interaktive Ablaufdiagramme kopieren - Der Benutzer kann interaktive Ablaufdiagramme kopieren. • Testlauf für interaktive Ablaufdiagramme ausführen - Der Benutzer kann einen Testlauf eines interaktiven Ablaufdiagramms durchführen. • Interaktive Ablaufdiagramme prüfen - Der Benutzer kann ein interaktives Ablaufdiagramm prüfen und Prozesse zur Ansicht von Einstellungen öffnen, aber nicht ändern. • Interaktive Ablaufdiagramme bereitstellen - Der Benutzer kann ein interaktives Ablaufdiagramm zur Bereitstellung oder Zurücknehmen der Bereitstellung markieren.
Globales Lernen	• Bin-Definitionen anzeigen - der Benutzer kann die unter Globales Lernen verfügbaren Bin-Definitionen anzeigen. • Bin-Definitionen hinzufügen - der Benutzer kann Bin-Definitionen hinzufügen/bearbeiten/löschen, die unter Globales Lernen verfügbar sind.
Globale Definitionen	• Echtzeit-Attribute anzeigen - Benutzer können Echtzeit-Attribute unter Globale Definition anzeigen. • Echtzeit-Attribute hinzufügen - der Benutzer kann Echtzeit-Attribute hinzufügen/bearbeiten/ändern, die unter Globale Definition verfügbar sind.

Beispiel für Designumgebungsberechtigungen

In diesem Beispiel werden die Berechtigungen aufgelistet, die für zwei unterschiedliche Rollen erteilt werden: Hierbei handelt es sich um eine Rolle für die Benutzer, die

interaktive Ablaufdiagramme erstellen, und um eine Rolle für die Benutzer, die die Interaktionsstrategien definieren.

Rolle für interaktive Ablaufdiagramme

Diese Tabelle enthält die Berechtigungen, die der Rolle für interaktive Ablaufdiagramme erteilt werden:

Kategorie	Campaign
Benutzerdefiniertes Makro	Die Benutzerrolle verfügt über folgende Berechtigungen: • Benutzerdefinierte Makros hinzufügen • Benutzerdefinierte Makros bearbeiten • Benutzerdefinierte Makros verwenden
Abgeleitetes Feld	Die Benutzerrolle verfügt über folgende Berechtigungen: • Abgeleitete Felder hinzufügen • Abgeleitete Felder bearbeiten • Abgeleitete Felder verwenden
Ablaufdiagrammvorlage	Die Benutzerrolle verfügt über folgende Berechtigungen: • Vorlagen einfügen
Segmentvorlage	Die Benutzerrolle verfügt über folgende Berechtigungen: • Segmente hinzufügen • Segmente bearbeiten

Kategorie	Campaign
Sitzung	Die Benutzerrolle verfügt über folgende Berechtigungen: • Sitzungsübersicht anzeigen • Interaktive Ablaufdiagramme anzeigen • Interaktive Ablaufdiagramme hinzufügen • Interaktive Ablaufdiagramme bearbeiten • Interaktive Ablaufdiagramme kopieren • Test für interaktive Ablaufdiagramme ausführen • Interaktive Ablaufdiagramme bereitstellen

Rolle für Interaktionsstrategien

Diese Tabelle enthält die Berechtigungen, die der Rolle für Interaktionsstrategien erteilt werden:

Kategorie	Campaign
Campaign	Die Benutzerrolle verfügt über folgende Berechtigungen: • Kampagnenübersicht anzeigen • Zielzellen von Kampagnen verwalten • Kampagneninteraktionsstrategien anzeigen

Kategorie	Campaign
	• Kampagneninteraktionsstrategien bearbeiten • Kampagneninteraktionsstrategien hinzufügen • Initiate® Bereitstellung von Kampagneninteraktionsstrategien initiieren
Angebot	Die Benutzerrolle verfügt über folgende Berechtigungen: • Angebotsübersicht anzeigen
Segmentvorlage	Die Benutzerrolle verfügt über folgende Berechtigungen: • Segmentübersicht anzeigen
Sitzung	Die Benutzerrolle verfügt über folgende Berechtigungen: • Interaktive Ablaufdiagramme prüfen

Kapitel 4. Verwalten von Unica Interact Datenquellen

Unica Interact erfordert mehrere Datenquellen, um ordnungsgemäß zu funktionieren. Einige Datenquellen enthalten die Informationen, die Unica Interact zum Funktionieren benötigt, und andere Datenquellen enthalten Ihre Daten.

Die folgenden Abschnitte beschreiben die Unica Interact-Datenquellen, einschließlich Informationen, die Sie benötigen, um sie ordnungsgemäß zu konfigurieren, sowie einiger Hinweise zu ihrer Wartung.

Unica Interact-Datenquellen

Unica Interact erfordert mehrere Gruppen von Daten, um zu funktionieren. Die Datengruppen werden gespeichert und von Datenquellen abgerufen. Welche Datenquellen Sie einrichten, hängt davon ab, welche Unica Interact-Funktionen Sie aktivieren.

- **Unica Campaign Systemtabellen**. Neben allen Daten für Unica Campaign enthalten die Unica Campaign-Systemtabellen Daten für Unica Interact-Komponenten, die Sie in der Designumgebung erstellen, wie z. B. Verfahrensregeln und interaktive Kanäle. Die Designumgebung und die Unica Campaign-Systemtabellen nutzen dieselbe physische Datenbank und dasselbe Schema.
- **Laufzeittabellen**(`systemTablesDataSource`) Diese Datenquelle enthält die Bereitstellungsdaten aus der Designumgebung, Staging-Tabellen für Kontakt- und Antwortverlauf sowie Laufzeitstatistikdaten.
- **Profiltabellen** (`prodUserDataSource`). Diese Datenquelle enthält - neben den in Echtzeit erfassten Informationen - alle Kundendaten, die von interaktiven Ablaufdiagrammen benötigt werden, um Besucher ordnungsgemäß in Smart Segments zu platzieren. Wenn Sie sich ausschließlich auf Echtzeitdaten verlassen, benötigen Sie keine Profiltabellen. Wenn Sie Profiltabellen verwenden, müssen Sie mindestens eine Profiltabelle pro vom interaktiven Kanal verwendeter Zielgruppenebene haben.

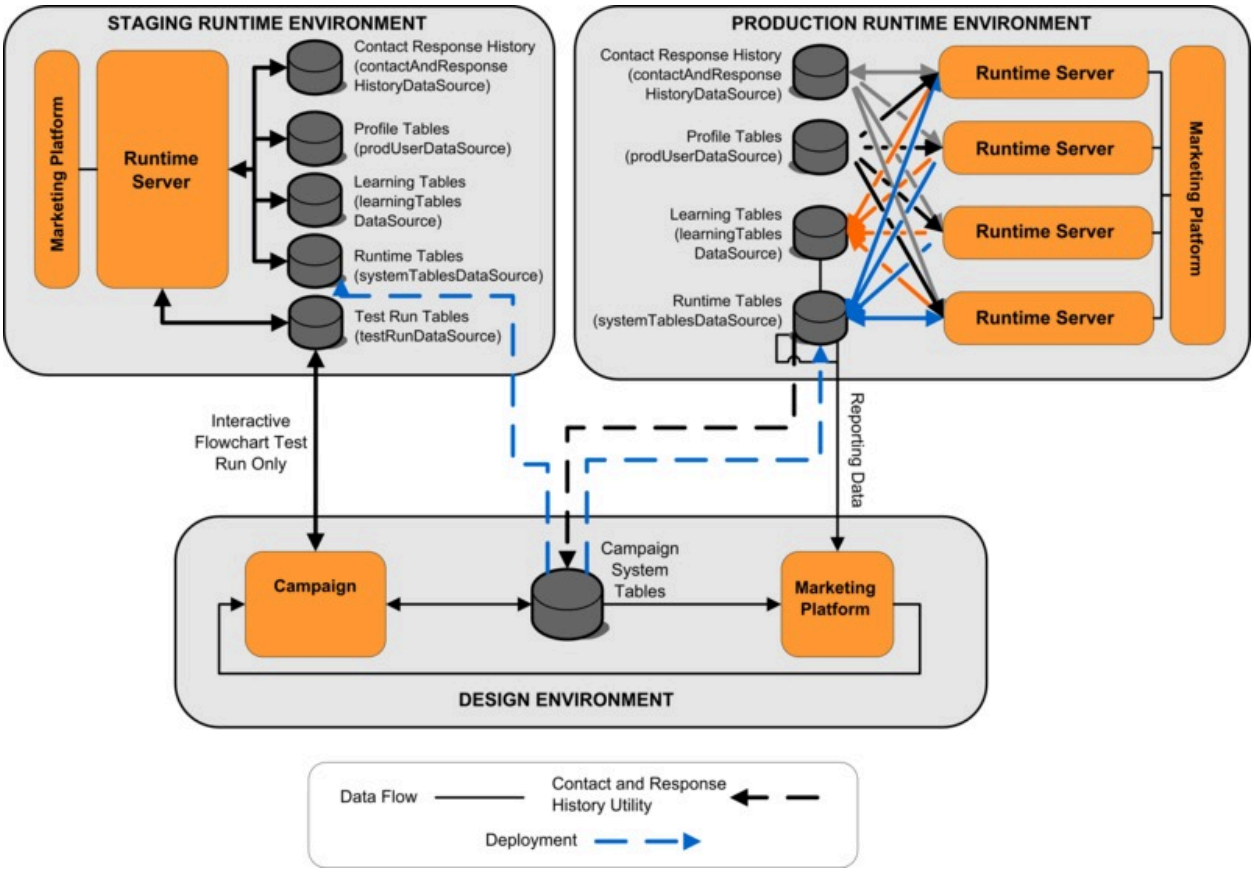
Die Profiltabellen können auch die Tabellen zum Erweitern der Angebotsbereitstellung enthalten, einschließlich der Tabellen für Angebotsunterdrückung, Bewertungsüberschreibung sowie globaler und individueller Angebotszuweisung.

- **Testlauftabellen** (`testRunDataSource`). Diese Datenquelle enthält ein Muster aller Daten, die von Ablaufdiagrammen benötigt werden, um Besucher in Smart Segments zu platzieren, einschließlich Daten, die nachahmen, was in Echtzeit während einer Interaktion erfasst wird. Diese Tabellen sind nur für die Servergruppe erforderlich, die als Testlaufservergruppe für die Designumgebung bestimmt ist.
- **Lerntabellen** (`learningTablesDataSource`). Diese Datenquelle enthält alle Daten, die vom integrierten Lerndienstprogramm gesammelt werden. Zu diesen Tabellen kann auch eine Tabelle gehören, die dynamische Attribute definiert. Wenn Sie die Lernfunktion nicht einsetzen oder ein externes Lerndienstprogramm verwenden, benötigen Sie keine Lerntabellen.
- **Kontakt- und Antwortverlauf für sitzungsübergreifende Antwort** (`contactAndResponseHistoryDataSource`). Diese Datenquelle enthält entweder die Unica Campaign-Kontaktverlaufstabellen oder Kopien davon. Wenn Sie die sitzungsübergreifende Antwortfunktion nicht verwenden, müssen Sie diese Kontaktverlaufstabellen nicht konfigurieren.

Datenbanken und die Anwendungen

Die Datenquellen, die Sie erstellen, damit sie von Unica Interact verwendet werden, können auch dazu eingesetzt werden, Daten mit anderen Unica-Anwendungen auszutauschen oder mit ihnen zu teilen.

Das nachfolgende Diagramm zeigt Unica Interact-Datenquellen und ihre Beziehung zu Unica-Anwendungen.



- Sowohl Unica Campaign als auch die Testlaufservergruppe greifen auf die Testlauftabellen zu.
- Die Testlauftabelle wird nur für Testläufe interaktiver Ablaufdiagramme verwendet.
- Wenn Sie einen Laufzeitserver zum Testen einer Implementierung (einschließlich der Unica Interact-API) verwenden, verwendet der Laufzeitserver die Profiltabellen für die Daten.
- Wenn Sie das Kontakt- und Antwortverlaufsmodul konfigurieren, verwendet das Modul einen ETL-Hintergrundprozess (ETL = Extrahieren, Transformieren, Laden), um Daten aus den Laufzeit-Staging-Tabellen in die Unica Campaign-Kontakt- und -Antwortverlaufstabellen zu verschieben.
- Die Funktion zur Berichterstellung fragt Daten aus den Lerntabellen, den Laufzeittabellen und den Unica Campaign-Systemtabellen ab, um Berichte in Unica Campaign anzuzeigen.

Sie sollten die Testlaufzeitumgebung so konfigurieren, dass ein anderer Tabellensatz als in Ihren Produktionsumgebungen verwendet wird. Mit separaten Tabellen für das Staging und die Produktion können Sie Ihre Testergebnisse von Ihren tatsächlichen Ergebnissen getrennt halten. Bedenken Sie, dass das Kontakt- und Antwortverlaufsmodul immer Daten in die tatsächlichen Unica Campaign-Kontakt- und -Antwortverlaufstabellen einfügt (Unica Campaign hat keine Testkontakt- und Antwortverlaufstabellen). Wenn Sie separate Lerntabellen für die Testlaufzeitumgebung haben und die Ergebnisse in Berichten sehen wollen, benötigen Sie eine separate Instanz von IBM® Cognos® BI, um die Lernberichte für die Testumgebung auszuführen.

Unica Campaign Systemtabellen

Wenn Sie die Unica Interact-Designumgebung installieren, erstellen Sie auch neue, Unica Interact-spezifische Tabellen in den Unica Campaign-Systemtabellen. Welche Tabellen Sie erstellen, hängt davon ab, welche Unica Interact-Funktionen Sie aktivieren.

Wenn Sie das Kontakt- und Antwortverlaufsmodul aktivieren, kopiert das Modul den Kontakt- und Antwortverlauf aus den Staging-Tabellen in den Laufzeittabellen in die Kontakt- und Antwortverlaufstabellen in den Unica Campaign-Systemtabellen. Die Standardtabellen sind `UA_ContactHistory`, `UA_DtlContactHist` und `UA_ResponseHistory`, aber das Kontakt- und Antwortverlaufsmodul verwendet jeweils Tabellen, die in Unica Campaign für die Kontakt- und Antwortverlaufstabellen zugeordnet sind.

Wenn Sie die globalen Angebotstabellen und Bewertungsüberschreibungstabellen verwenden, um Angebote zuzuweisen, müssen Sie möglicherweise die Tabelle `UACI_ICBatchOffers` in den Unica Campaign-Systemtabellen auffüllen, wenn Sie Angebote verwenden, die nicht in den Verfahrensregeln für den interaktiven Kanal enthalten sind.

Laufzeittabellen

Wenn mehr als eine Zielgruppenebene definiert ist, müssen Sie Staging-Tabellen für die Kontakt- und Antwortverlaufsdaten für jede Zielgruppenebene erstellen.

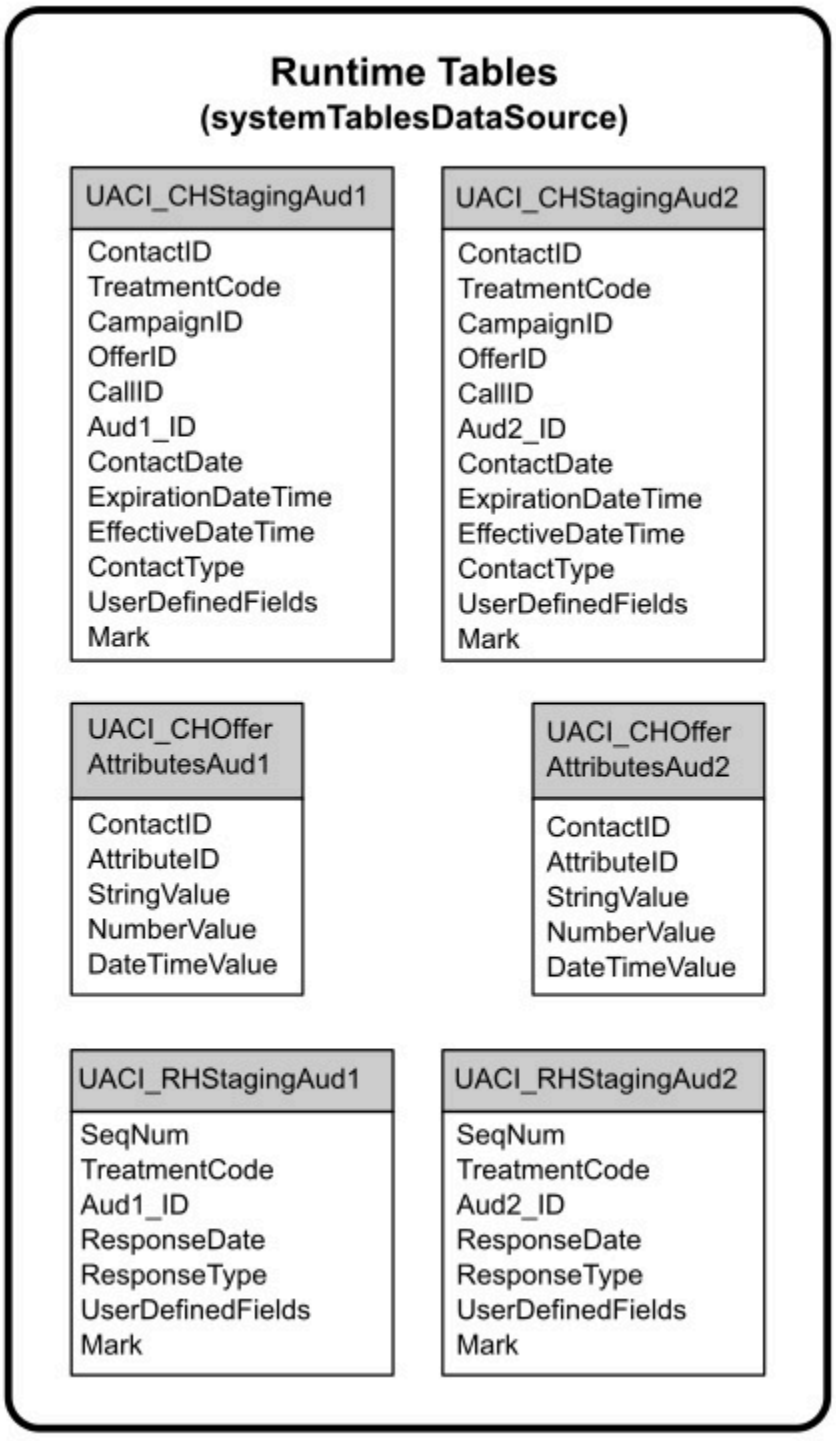
Die SQL-Scripts erstellen die folgenden Standard-Zielgruppenebenen:

- UACI_CHStaging
- UACI_CHOfferAttrib
- UACI_RHStaging

Sie müssen Kopien dieser drei Tabellen für jede Ihrer Zielgruppenebenen in den Laufzeittabellen erstellen.

Wenn Ihre Unica Campaign-Kontakt- und -Antwortverlaufstabellen benutzerdefinierte Felder haben, müssen Sie dieselben Feldnamen und Typen in den Tabellen UACI_CHStaging und UACI_RHStaging erstellen. Sie können diese Felder während der Laufzeit auffüllen, indem Sie Name/Wert-Paare desselben Namens in den Sitzungsdaten erstellen. Beispiel: Ihre Kontakt- und Antwortverlaufstabellen enthalten das Feld catalogID. Sie müssen das Feld catalogID beiden Tabellen UACI_CHStaging und UACI_RHStaging hinzufügen. Später füllt die Unica Interact-API dieses Feld auf, indem ein Ereignisparameter als ein Name/Wert-Paar namens catalogID definiert wird. Sitzungsdaten können durch die Profiltabelle, durch zeitbezogene Daten, durch die Lernfunktion oder die Unica Interact-API bereitgestellt werden.

Das folgende Diagramm zeigt Beispieltabellen für die Zielgruppenebenen Aud1 und Aud2. Dieses Diagramm umfasst nicht alle Tabellen in der Laufzeitdatenbank.


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

Alle Felder in den Tabellen sind erforderlich. Sie können `CustomerID` und `UserDefinedFields` ändern, um Ihren Unica Campaign-Kontakt- und -Antwortverlaufstabellen zu entsprechen.

Testlauftabellen

Die Testlauftabellen werden für Testläufe interaktiver Ablaufdiagramme verwendet. Testläufe interaktiver Ablaufdiagramme sollten Ihre Segmentierungslogik testen. Sie müssen nur eine Testlaufdatenbank für Ihre Unica Interact-Installation konfigurieren. Die Testlauftabellen müssen nicht in einer eigenständigen Datenbank sein. Sie könnten beispielsweise Ihre Kundendatentabellen für Unica Campaign verwenden.

Der den Testlauftabellen zugeordnete Datenbankbenutzer muss CREATE-Berechtigungen haben, um die Testlauf-Ergebnistabellen hinzuzufügen.

Die Testlauf-Datenbank muss alle Tabellen enthalten, die im interaktiven Kanal zugeordnet sind.

Diese Tabellen sollten Daten enthalten, um Szenarien auszuführen, die Sie in Ihren interaktiven Ablaufdiagrammen testen wollen. Beispiel: Wenn Ihre interaktiven Ablaufdiagramme Logik haben, um Personen in Segmente basierend auf der getroffenen Auswahl in einem Voicemail-System zu sortieren, sollten Sie zumindest eine Zeile für jede mögliche Auswahl haben. Wenn Sie eine Interaktion erstellen, die mit einem Formular auf Ihrer Website funktioniert, sollten Sie Zeilen aufnehmen, die fehlende oder fehlerhafte Daten darstellen; verwenden Sie beispielsweise `name@domaincom` für den Wert einer E-Mail-Adresse.

Jede Testlauftabelle muss zumindest eine Liste von IDs für die entsprechende Zielgruppenebene und eine Spalte haben, die die Echtzeitdaten darstellt, die Sie zu verwenden gedenken. Da Testläufe keinen Zugriff auf Echtzeitdaten haben, müssen Sie Beispieldaten für jedes Element der erwarteten Echtzeitdaten bereitstellen. Beispiel: Wenn Sie Daten verwenden möchten, die Sie in Echtzeit erfassen können, wie der Name der zuletzt besuchten Webseite (im Attribut `lastPageVisited` gespeichert) oder die Anzahl der Artikel im Warenkorb (im Attribut `shoppingCartItemCount` gespeichert), müssen Sie Spalten mit denselben Namen erstellen und die Spalten mit Beispieldaten auffüllen.

Dies ermöglicht es Ihnen, Testläufe auf den Verzweigungen Ihrer Ablaufdiagrammlogik durchzuführen, die auf Verhalten oder Kontext basieren.

Testläufe von interaktiven Ablaufdiagrammen sind nicht für große Datenmengen optimiert. Sie können die Anzahl der für den Testlauf verwendeten Zeilen im Interaktionsprozess begrenzen. Die führt allerdings dazu, dass immer der erste Satz von Zeilen ausgewählt wird. Um sicherzustellen, dass andere Zeilensätze ausgewählt werden, verwenden Sie verschiedene Ansichten der Testlauftabellen.

Um die Durchsatzleistung von interaktiven Ablaufdiagrammen in Laufzeit zu testen, müssen Sie eine Testlaufzeitumgebung erstellen, einschließlich einer Profiltabelle für die Testumgebung.

In der Praxis benötigen Sie möglicherweise drei Testtabellengruppen - eine Testlauftabelle für Testläufe von interaktiven Ablaufdiagrammen, Testprofiltabellen für die Testservergruppe und einen Satz von Produktionsprofiltabellen.

Die Standarddatentypen für dynamisch erstellte Tabellen überschreiben

Die Interact-Laufzeitumgebung erstellt dynamisch Tabellen in zwei Szenarien: während des Testlaufs eines Ablaufdiagramms und während der Ausführung eines Prozesses "Momentaufnahme", der in eine Tabelle schreibt, die noch nicht existiert. Um diese Tabellen zu erstellen, verwendet Interact fest codierte Datentypen für jeden unterstützten Datenbanktyp.

Sie können die Standarddatentypen überschreiben, indem Sie eine Tabelle von alternativen Datentypen namens `uaci_column_types` in `testRunDataSource` oder `prodUserDataSource` erstellen. Diese zusätzliche Tabelle ermöglicht es Interact, seltene Fälle zu verarbeiten, die nicht durch die fest codierten Datentypen abgedeckt sind.

Wenn die Tabelle `uaci_column_types` definiert ist, verwendet Interact die Metadaten für die Spalten als die Datentypen, die für Tabellengenerierungen verwendet werden. Wenn die Tabelle `uaci_column_types` nicht definiert ist oder wenn es Ausnahmebedingungen gibt, die beim Versuch auftreten, die Tabelle zu lesen, werden die Standarddatentypen verwendet.

Beim Start überprüft das Laufzeitsystem zuerst `testRunDataSource` auf die Tabelle `UACI_column_types`. Wenn die Tabelle `UACI_column_types` nicht in `testRunDataSource` vorhanden ist oder wenn `prodUserDataSource` von einem anderen Datenbanktyp ist, überprüft Interact anschließend `prodUserDataSource` auf die Tabelle.

Überschreiben der Standarddatentypen

Verwenden Sie dieses Verfahren, um die Standarddatentypen für dynamisch erstellte Tabellen zu überschreiben.

Nach jeder Änderung der Tabelle `uaci_column_types` müssen Sie den Laufzeitserver erneut starten. Planen Sie Ihre Änderungen so ein, dass die Operationen durch den Neustart des Servers möglichst wenig beeinträchtigt werden.

1. Erstellen Sie eine Tabelle in `TestRunDataSource` oder `ProdUserDataSource` mit den folgenden Eigenschaften:

   Tabellenname: `UACI_Column_types`

   Spaltennamen:

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

   Definieren Sie jede Spalte unter Verwendung des entsprechenden Datentyps, der von Ihrer Datenbank unterstützt wird.
2. Starten Sie den Laufzeitserver erneut, um es Interact zu ermöglichen, die neue Tabelle zu erkennen.

Standarddatentypen für dynamisch erstellte Tabellen

Für jede unterstützte, vom Unica Interact-Laufzeitsystem verwendete Datenbank gibt es fest codierte Datentypen, die standardmäßig für die Spalten "Gleitkomma", "Zahl", "Datum/Uhrzeit" und "Zeichenfolge" verwendet werden.

Tabelle 1. Standarddatentypen für dynamisch erstellte Tabellen

Datenbank	Standarddatentypen
DB2®	• `float` • `bigint` • `timestamp` • `varchar`
Oracle	• `float` • `number(19)` • `timestamp` • `varchar2`
SQL Server	• `float` • `bigint` • `datetime` • `nvarchar`

Profildatenbank

Der Inhalt der Profildatenbank hängt vollständig von den Daten ab, die Sie für die Konfiguration Ihrer interaktiven Ablaufdiagramme und der Unica Interact API benötigen. Unica Interact erfordert oder empfiehlt, dass jede Datenbank bestimmte Tabellen oder Daten enthält.

Die Profildatenbank muss Folgendes enthalten:

- Alle Tabellen, die im interaktiven Kanal zugeordnet sind.

  Diese Tabellen müssen alle Daten enthalten, die für das Ausführen Ihrer interaktiven Ablaufdiagramme in der Produktion erforderlich sind. Diese Tabellen sollten abgewickelt, optimiert und richtig indiziert sein. Da es Leistungseinbußen beim Zugriff auf Dimensionsdaten gibt, sollten Sie soweit möglich ein denormalisiertes Schema verwenden. Zumindest sollten Sie die Profiltabelle auf der Zielgruppenebene ID-Felder indizieren. Wenn es weitere aus dimensionalen Tabellen abgerufene

Felder gibt, sollten diese entsprechend indiziert sein, um die Datenbank-Abrufzeit zu verringern. Die Zielgruppen-IDs für die Profiltabellen müssen mit den Zielgruppen-IDs übereinstimmen, die in Unica Campaign definiert sind.

- Wenn Sie die Konfigurationseigenschaft `enableScoreOverrideLookup` auf true festlegen, müssen Sie eine Bewertungsüberschreibungstabelle für zumindest eine Zielgruppenebene aufnehmen. Sie definieren die Namen der Bewertungsüberschreibungstabellen mit der Eigenschaft `scoreOverrideTable`.

  Die Bewertungsüberschreibungstabelle kann einzelne Kunde-zu-Angebot-Paarungen haben. Sie können eine Beispiel-Bewertungsüberschreibungstabelle `UACI_ScoreOverride` erstellen, indem Sie das SQL-Script `aci_usrtab` in Ihrer Profildatenbank ausführen. Sie sollten auch diese Tabelle auf der Zielgruppen-ID-Spalte indizieren.

  Wenn Sie die Konfigurationseigenschaft `enableScoreOverrideLookup` auf false festlegen, müssen Sie keine Bewertungsüberschreibungstabelle aufnehmen.
- Wenn Sie die Konfigurationseigenschaft `enableDefaultOfferLookup` auf true festlegen, müssen Sie die globale Angebotstabelle (`UACI_DefaultOffers`) aufnehmen. Sie können eine globale Angebotstabelle erstellen, indem Sie das SQL-Script `aci_usrtab` in Ihrer Profildatenbank ausführen.

  Die globale Angebotstabelle kann Zielgruppe-zu-Angebot-Paarungen enthalten.
- Wenn Sie die Konfigurationseigenschaft `enableOfferSuppressionLookup` auf true festlegen, müssen Sie eine Angebotsunterdrückungstabelle für zumindest eine Zielgruppenebene aufnehmen. Sie definieren die Namen der Angebotsunterdrückungstabelle mit der Eigenschaft `offerSuppressionTable`.

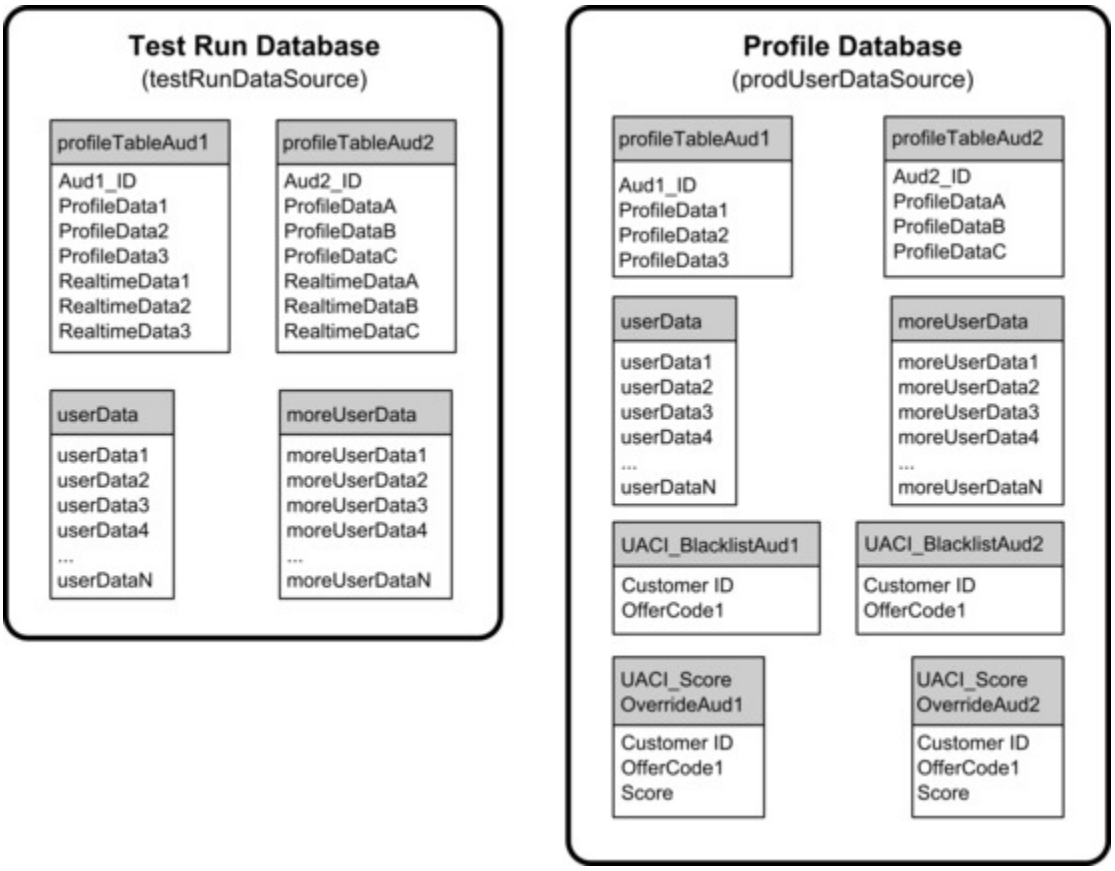
  Die Tabelle für Angebotsunterdrückung kann eine Zeile für jedes für ein Zielgruppenmitglied unterdrücktes Angebot enthalten, auch wenn ein Eintrag nicht für alle Zielgruppenmitglieder erforderlich ist. Sie können eine Beispiel-Angebotsunterdrückungstabelle, `UACI_BlackList`, erstellen, indem Sie das SQL-Script `aci_usrtab` in Ihrer Profildatenbank ausführen.

  Wenn Sie die Konfigurationseigenschaft `enableOfferSuppressionLookup` auf false festlegen, müssen Sie keine Angebotsunterdrückungstabelle aufnehmen.

Ein großes Datenvolumen in einer dieser Tabellen kann möglicherweise die Leistung beeinträchtigen. Um beste Ergebnisse zu erzielen, erstellen Sie entsprechende Indizes auf die Zielgruppenebenenspalten bei Tabellen, die zur Laufzeit verwendet werden und große Datenvolumen haben.

Alle oben genannten Konfigurationseigenschaften befinden sich in der Kategorie **Interact > Profil** oder **Hinteract > Profil > Benutzergruppenebenen > AudienceLevel** Das SQL-Script `aci_usrtab` befindet sich im `ddl`-Verzeichnis in Ihrem Laufzeitumgebungsinstallationsverzeichnis.

Das folgende Diagramm zeigt Beispieltabellen für den Testlauf und Profildatenbanken für die Zielgruppenebenen Aud1 und Aud2.



Lerntabellen

Falls Sie das integrierte Lernen von Unica Interact verwenden, müssen Sie die Lerntabellen konfigurieren. Diese Tabellen enthalten alle Daten, die die integrierte Lernfunktion benötigt.

Wenn Sie dynamische Lernattribute verwenden, müssen Sie die Tabelle `UACI_AttributeList` auffüllen.

Die Lernfunktion umfasst das Schreiben in temporäre Staging-Tabellen und das Aggregieren von Informationen aus Staging-Tabellen in Lerntabellen. Die Konfigurationseigenschaften `insertRawStatsIntervalInMinutes` und `aggregateStatsIntervalInMinutes` in der Kategorie `Interact > offerserving > Built-in Learning Config` legt fest, wie oft die Lerntabellen aufgefüllt werden.

Das Attribut `insertRawStatsIntervalInMinutes` legt fest, wie oft Annahme- und Kontaktinformationen für jeden Kunden und jedes Angebot aus dem Speicher in die Staging-Tabellen `UACI_OfferStatsTX` und `UACI_OfferTxAll` verschoben werden. Die in den Staging-Tabellen gespeicherten Informationen werden aggregiert und in die Tabellen `UACI_OfferStats` und `UACI_OfferStatsAll` in regelmäßigen Intervallen verschoben, die durch die Konfigurationseigenschaft `aggregateStatsIntervalInMinutes` festgelegt werden.

Das integrierte Lernen von Unica Interact verwendet diese Daten, um endgültige Bewertungen für Angebote zu berechnen.

Kontaktverlauf für sitzungsübergreifende Antwortverfolgung

Wenn Sie die sitzungsübergreifende Antwortfunktion aktivieren, benötigt die Laufzeitumgebung Lesezugriff auf die Unica Campaign-Kontaktverlaufstabellen. Sie können die Laufzeitumgebung konfigurieren, die Unica Campaign-Systemtabellen anzuzeigen, oder Sie können die Unica Campaign-Kontaktverlaufstabellen kopieren. Wenn Sie eine Kopie der Tabellen erstellen, müssen Sie den Prozess steuern, um die Kopie auf dem letzten Stand zu halten. Das Kontakt- und Antwortverlaufsmodul aktualisiert nicht die Kopie der Kontaktverlaufstabellen.

Sie müssen das SQL-Script `aci_crhtab` auf diesen Kontaktverlaufstabellen ausführen, um Tabellen hinzuzufügen, die für die sitzungsübergreifende Antwortverfolgungsfunktion erforderlich sind.

Datenbankscripts zur Aktivierung von Unica-Funktionen ausführen

Um die in Unica verfügbaren optionalen Funktionen nutzen zu können, führen Sie Datenbankscripts für die Datenbank aus, um Tabellen zu erstellen oder vorhandene Tabellen zu aktualisieren.

Ihre Interact-Installation mit der Designzeit- als auch der Laufzeitumgebung enthält ddl-Funktionsscripts. Die ddl-Scripts fügen erforderliche Spalten zu Ihren Tabellen hinzu.

Um diese optionalen Funktionen zu aktivieren, führen Sie das entsprechende Script für die angegebene Datenbank oder Tabelle aus.

`dbType` ist der Datenbanktyp, z. B. `sqlsvr` für Microsoft™ SQL Server, `ora` für Oracle, `db2` für DB2®, und MariaDB für MariaDB-Datenbank.

Verwenden Sie die folgende Tabelle, um Datenbankscripts für die Datenbank auszuführen, um Tabellen zu erstellen oder vorhandene Tabellen zu aktualisieren:

Tabelle 2. Datenbankscripts

Diese vierspaltige Tabelle enthält in der ersten Spalte die Funktionsnamen, in der zweiten Spalte die Funktionsscripts, in der dritten Spalte Details zur Ausführung der Datenbankscripts und in der vierten Spalte die Änderungen an den Tabellen.

Funktionsname	Funktionsscript	Ausführen für	Ändern
Globale Angebote, Angebotsunterdrückung und Bewertungsüberschreibung	aci_usrtab_dbType.sql in *`Interact_Home`*`\ddl\acifeatures\` (Installationsverzeichnis der Laufzeitumgebung)	Ihre Profildatenbank (userProdDataSource)	Erstellt die Tabellen UACI_DefaultOffers, UACI_BlackList und UACI_ScoreOverride.

Tabelle 2. Datenbankscripts

Diese vierspaltige Tabelle enthält in der ersten Spalte die Funktionsnamen, in der zweiten Spalte die Funktionsscripts, in der dritten Spalte Details zur Ausführung der Datenbankscripts und in der vierten Spalte die Änderungen an den Tabellen.

(Fortsetzung)

Funktionsname	Funktionsscript	Ausführen für	Ändern
Bewertung	aci_scoringfeature_dbType.sql in *Interact_Home*\ddl\acifeatures\ (Installationsverzeichnis der Laufzeitumgebung)	Bewertungsüberschreibungstabellen in Ihrer Profildatenbank (userProdDataSource)	Fügt die Spalten LikelihoodScore und AdjExploreScore hinzu.
Lernen	aci_lrnfeature_dbType.sql in *Interact_Home*\interactDT\ddl\acifeatures\ (Installationsverzeichnis der Designzeitumgebung)	Unica Campaign-Datenbank mit den Kontaktverlaufstabellen	Fügt die Spalten RTSelectionMethod, RTLearningMode und RTLearningModelID zur Tabelle UA_DtlContactHist hinzu. Fügt außerdem die Spalten RTLearningMode und RTLearningModelID zur Tabelle UA_ResponseHistory hinzu. Dieses Script ist außerdem für die Berichtsfunktionen des optional erhältlichen -Berichtspakets erforderlich.

Zum Thema Sitzungsübergreifende Kontaktverfolgung

Ein Kontaktereignis wird an Interact gepostet, nachdem ein Angebot einem Endbenutzer präsentiert wurde. Derzeit wird davon ausgegangen, dass der Endnutzer handelt, sobald das Angebot präsentiert wird. Daher muss das Kontaktereignis innerhalb derselben Interact-Sitzung gepostet werden wie die getOffers-API-Anforderung, die das ursprüngliche Angebot zurückgibt. Es gibt jedoch viele Szenarien, in denen Kontaktereignisse in einer anderen Sitzung stattfinden.

Da für das Posten eines Kontaktereignisses die Sitzungsdaten und personalisierte Angebotsattribute benötigt werden, müssen die erforderlichen Daten nach getOffers im System aufbewahrt werden. Aufgrund der begrenzten Speicherkapazität können sitzungsübergreifende Kontaktereignisse nur innerhalb einer begrenzten Zeitspanne nach getOffers bearbeitet werden.

So funktioniert ein sitzungsübergreifender Kontakt.

- Wenn ein Kontaktereignis ohne Verfahrens- oder Angebotscode gesendet wird, werden alle Verfahren, die beim vorherigen Aufruf von getOffers zurückgegeben wurden, protokolliert und immer im aktuellen Sitzungsobjekt gespeichert. Interact sucht nach dem passenden Verfahren unter den verfügbaren Verfahren, die beim vorherigen Aufruf von getOffers zurückgegeben wurden.
- Wenn kein Verfahren im Sitzungsobjekt gefunden wird und die sitzungsübergreifende Kontaktverfolgung aktiviert ist, sucht Interact nach dem passenden Verfahren und lädt es in den konfigurierten Datenspeicher.
- Wenn kein Kontakttyp angegeben ist oder der angegebene Kontakttyp als echtes Kontaktereignis definiert ist, wird dieses Ereignis an die Lernmaschine gesendet und die Angebotsunterdrückungsregeln werden gegebenenfalls aktualisiert.
- Ein Fehler wird zurückgegeben, wenn keine Verfahren gefunden werden.

Datenbank- und Schemaänderung

Die folgende Tabelle wird in der Laufzeitdatenbank hinzugefügt. Sie enthält die personalisierten Verfahren, die den Kunden über die getOffers-API-Aufrufe oder ausgehende Anrufe übermittelt werden. Diese Tabelle ist für jede Zielgruppenebene erforderlich.

Tabelle 3. UACI - Verfahren.

Spalte	DataType	Syntax
SeqNum	Zahl	Die interne ID dieses Verfahrens.
TreatmentCode	varchar	Der Verfahrenscode dieses personalisierten Verfahrens.
OfferID	Zahl	Die ID des Angebots, von dem dieses Verfahren abgeleitet ist.
OfferCode	varchar	Der Code des Angebots, von dem dieses Verfahren abgeleitet ist. Wenn der Angebotscode aus mehreren Teilen besteht, werden alle Teile zu einer einzigen, durch Komma getrennten Zeichenfolge verkettet. Wenn die Gesamtlänge der Angebotscodes lang ist, kann es erforderlich sein, die Gesamtlänge dieser Spalte zu erhöhen.

Tabelle 3. UACI - Verfahren. (Fortsetzung)

Spalte	DataType	Syntax
PresentDate	timestamp	Der Zeitstempel, zu dem dieses personalisierte Verfahren bereitgestellt wurde.
IPName	varchar	Der Name des Interaktionspunktes, der für dieses Verfahren erstellt wurde. Wenn es keinen zugehörigen Interaktionspunkt gibt, wie z. B. bei wenigen ausgelösten Nachrichtenfällen, ist der Wert dieses Feldes null.
Kunden-ID	Zahl	Die Zielgruppen-ID, wenn die Zielgruppenebene "Kunde" ist. Wenn die Zielgruppenebene nicht "Kunde" ist, muss diese Spalte durch die entsprechenden Zielgruppen-ID-Felder ersetzt werden.
Kontaktiert	Zahl	Dadurch wird festgelegt, ob ein Kontaktereignis in dieses Verfahren gepostet wird. Wenn dieses Verfahren aufgerufen wird, gibt es den Wert "1" zurück, und wenn es nicht aufgerufen wird, gibt es einen anderen Wert zurück.

Tabelle 3. UACI - Verfahren. (Fortsetzung)

Spalte	DataType	Syntax
Details	varchar	Hier werden die mit diesem personalisierten Verfahren verbundenen Informationen angegeben, wie z. B. die Sitzungsparameter und Profilattribute, die für die entsprechenden CH/RH-Staging-Tabellen und die personalisierten Angebotsattribute erforderlich sind.

Informationen zur Verfolgung von Kontakt- und Antwortverlauf

Sie können die Laufzeitumgebung so konfigurieren, dass der Kontakt- und Antwortverlauf in den Unica Campaign-Kontakt- und Antwortverlaufstabellen erfasst wird. Die Laufzeitserver speichern den Kontakt- und Antwortverlauf in Staging-Tabellen. Das Kontakt- und Antwortverlaufsmodul kopiert diese Daten aus den Staging-Tabellen in die Unica Campaign-Kontakt- und Antwortverlaufstabellen.

Das Kontakt- und Antwortverlaufsmodul funktioniert nur, wenn Sie die Eigenschaften `Campaign > partitions > partition1 > Interact interactInstalled` und `contactAndResponseHistTracking > isEnabled` auf der Seite "Konfiguration" für die Designumgebung auf`yes` festlegen.

Wenn Sie das sitzungsübergreifende Antwortüberwachungsmodul verwenden, ist das Kontakt- und Antwortverlaufsmodul eine separate Entität.

Kontakt- und Antworttypen

Sie können einen Kontakttyp und zwei Antworttypen mit Unica Interact erfassen. Sie können auch zusätzliche benutzerdefinierte Antworttypen mit der Methode `postEvent` aufzeichnen.

Tabelle mit contactAndResponseHistTracking-Eigenschaften

In dieser Tabelle werden die Eigenschaften aufgelistet, die in der Kategorie `contactAndResponseHistTracking` enthalten sind:

Ereignis	Kontakt-/Antworttyp	Konfigurationseigenschaft
Angebotskontakt protokollieren	Kontakt	`kontaktiert`
Angebotsannahme protokollieren	Antwort	`accept (annehmen)`
Angebotsablehnung protokollieren	Antwort	`ablehnen (reject)`

Tabelle mit UA_UsrResponseType-Eigenschaften

Stellen Sie sicher, dass die Spalte `CountsAsResponse` der Tabelle `UA_UsrResponseType` in den Unica Campaign -Systemtabellen ordnungsgemäß konfiguriert ist. Alle diese Antworttypen müssen in der Tabelle `UA_UsrResponseType` vorhanden sein.

Damit ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` vorliegt, müssen Sie für alle Spalten der Tabelle, einschließlich`CountsAsResponse`, einen Wert definieren. Die folgenden Werte sind für `CountsAsResponse` gültig:

- 0 - keine Antwort
- 1 - eine Antwort
- 2 - eine Ablehnung
-

Diese Antworten werden für die Berichterstellung verwendet.

Zusätzliche Kontakttypen

In Interact können Sie die postEvent-Methode in der Interact-API verwenden, um ein Kontaktereignis auszulösen. Sie können das System auch so erweitern, dass der Anruf nach dem Ereignis zusätzliche benutzerdefinierte Kontakttypen aufzeichnen kann. Alle diese

Kontakttypen müssen in der Tabelle UA_ContactStatus in den Campaign-Systemtabellen vorhanden sein. Durch die Verwendung spezifischer Ereignisparameter für die postEvent-Methode können Sie zusätzliche Kontakttypen aufzeichnen und bestimmen, ob es sich um einen echten Kontakt handelt oder nicht. Um zusätzliche Kontakttypen zu protokollieren, müssen Sie die folgenden Ereignisparameter hinzufügen:

UACIContactStatusCode - eine Zeichenfolge, die einen Kontakttypcode darstellt. Der Wert muss ein gültiger Eintrag in der Tabelle UA_ContactStatus sein. Damit ein gültiger Eintrag im UA_ContactStatus vorliegt, müssen Sie alle Spalten in der Tabelle definieren, einschließlich CountsAsContact. Gültige Werte für CountsAsContact sind 0 und 1. 0 bedeutet keinen erfolgreichen Kontakt, 1 bedeutet einen erfolgreichen Kontakt.

Zusätzliche Antworttypen

In Unica Interact können Sie die Methode postEvent in der Unica Interact-API verwenden, um ein Ereignis auszulösen, das eine Aktion "Accept" oder "Reject" für ein Angebot protokolliert. Sie können das System auch erweitern, damit der Aufruf `postEvent` zusätzliche Antworttypen erfasst, wie z. B. Explore, Consider, Commit oder Fulfill.

Alle diese Antworttypen müssen in der Tabelle `UA_UsrResponseType` in den Unica Campaign-Systemtabellen vorhanden sein. Mit spezifischen Ereignisparametern für die Methode `postEvent` können Sie zusätzliche Antworttypen aufzeichnen und festlegen, ob ein Akzeptieren in den Lernprozess einbezogen werden soll, und es wird empfohlen, keine Mehrfachantworten (Akzeptieren/Ablehnen) für einen einzelnen Kontakt zu veröffentlichen, da dies zu falschen Lernergebnissen führen kann.

Um zusätzliche Antworttypen zu protokollieren, müssen Sie folgende Ereignisparameter hinzufügen:

- **UACIResponseTypeCode** - eine Zeichenfolge, die einen Antworttypcode darstellt. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` sein.

  Damit ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` vorliegt, müssen Sie für alle Spalten der Tabelle, einschließlich`CountsAsResponse`, einen Wert definieren. Gültige Werte für `CountsAsResponse` sind 0, 1 oder 2. 0 gibt keine Antwort an, 1

gibt eine Antwort an und 2 gibt eine Ablehnung an. Diese Antworten werden für die Berichterstellung verwendet.

- **UACILogToLearning** -

  Der Parameter 'UACILogToLearning' ist in Version 11.0 veraltet. Stattdessen werden die tatsächlichen Werte, die in den Tabellen 'UA_ContactStatus' und 'UA_UsrResponseType' aus der Campaign-Datenbank definiert sind, zusammen mit den Werten, die in den Parametern 'Affinium|interact|Dienste|contactHist|contactStatusCodes' und 'Affinium|interact|Dienste|responseHist|responseTypeCodes' definiert sind, vom Interact-System berücksichtigt.
  ,

  Wenn Sie 'UACILogToLearning= 1' in einem Post-Event-Aufruf übergeben, dann wird die konfigurierte Aktion, die dem Antworttyp/Kontaktstatus zugeordnet ist, ignoriert, und dieses Ereignis wird immer als echte Antwort/Kontakt behandelt.

Sie können Folgendes erstellen: mehrere Ereignisse mit der Aktion Angebotsannahme protokollieren, ein Ereignis für jeden Antworttyp, den Sie protokollieren möchten, oder ein einzelnes Ereignis mit der Aktion Angebotsannahme protokollieren, das Sie für jeden `postEvent`-Aufruf verwenden, mit dem Sie separate Antworttypen protokollieren.

Sie erstellen beispielsweise ein Ereignis mit der Aktion "Angebotsannahme protokollieren" für jeden Antworttyp. Sie definieren die folgenden benutzerdefinierten Antworten in der Tabelle `UA_UsrResponseType` [als Name (Code)]: Explore (EXP), Consider (CON), und Commit (CMT). Dann erstellen Sie drei Ereignisse und nennen Sie LogAccept_Explore, LogAccept_Consider und LogAccept_Commit. Alle drei Ereignisse sind identisch (weisen die Aktion Angebotsannahme protokollieren auf), haben jedoch unterschiedliche Namen, sodass die Person, die mit der API arbeitet, sie unterscheiden kann.

Sie können auch ein einzelnes Ereignis mit der Aktion "Angebotsannahme protokollieren" erstellen, das Sie für alle benutzerdefinierten Antworttypen verwenden. Nennen Sie es beispielsweise LogCustomResponse.

Beim Arbeiten mit der API besteht kein funktionaler Unterschied zwischen den Ereignissen. Die Namenskonventionen können den Code jedoch verständlicher machen. Wenn Sie

jede benutzerdefinierte Antwort anders benennen, zeigt der Bericht Aktivitätsübersicht Kanalereignisse auch genauere Informationen an.

Zuerst definieren Sie alle Name/Wert-Paare.

```
//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);
```

Dieses erste Beispiel zeigt die Verwendung von einzelnen Ereignissen.

```
//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);
```

Dieses zweite Beispiel zeigt die Verwendung nur eines Ereignisses.

```
//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);
```

Beide Beispiele führen genau dieselben Aktionen durch, aber eine Version ist möglicherweise lesbarer als die andere.

Zuordnung der Staging-Tabellen der Laufzeitumgebung zu den Unica Campaign-Verlaufstabellen

Die Unica Interact-Staging-Tabellen für den Kontaktverlauf werden den Unica Campaign-Verlaufstabellen zugeordnet. Sie müssen für jede Zielgruppenebene über eine der Staging-Tabellen der Laufzeitumgebung verfügen.

Zuordnung der Staging-Tabelle "UACI_CHStaging" für den Kontaktverlauf

Diese Tabelle zeigt die Zuordnung der Staging-Tabelle `UACI_CHStaging` der Laufzeitumgebung zu der Unica Campaign-Kontaktverlaufstabelle. Die angezeigten Tabellennamen sind die Beispieltabellen, die für die Standardzielgruppe in den Laufzeittabellen und in den Unica Campaign-Systemtabellen erstellt werden.

Tabelle 4. Kontaktverlauf

UACI_CHStaging Unica Interact Spaltenname der Staging-Tabelle im Kontaktprotokoll	Unica Campaign Kontaktprotokolltabelle	Name der Tabellenspalte
ContactID	Nicht zutreffend	Nicht zutreffend
TreatmentCode	UA_Treatment	TreatmentCode
CampaignID	UA_Treatment	CampaignID

Tabelle 4. Kontaktverlauf (Fortsetzung)

UACI_CHStaging Unica Interact Spaltenname der Staging-Tabelle im Kontaktprotokoll	Unica Campaign Kontaktprotokolltabelle	Name der Tabellenspalte
OfferID	UA_Treatment	OfferID
CellID	UA_Treatment	CellID
Kunden-ID	UA_DtlContactHist	Kunden-ID
ContactDate	UA_DtlContactHist	KontaktDatumUhrzeit
ExpirationDateTime	UA_Treatment	ExpirationDateTime
EffectiveDateTime	UA_Treatment	EffectiveDateTime
ContactType	UA_DtlContactHist	ContactStatusID
ContactStatusCode	UA_DtlContactHist	ContactStatusId
UserDefinedFields	UA_DtlContactHist	UserDefinedFields

`ContactID` ist ein Schlüssel, mit dem die Tabelle `UACI_CHOfferAtrib` mit der Tabelle `UACI_CHStaging` verknüpft wird. Die Spalte `userDefinedFields` kann beliebige Daten Ihrer Wahl enthalten.

Zuordnung der Staging-Tabelle "UACI_CHOfferAttrib" für den Kontaktverlauf

Diese Tabelle zeigt die Zuordnung der Staging-Tabelle `UACI_CHOfferAttrib` der Laufzeitumgebung zu der Unica Campaign-Kontaktverlaufstabelle. Die angezeigten Tabellennamen sind die Beispieltabellen, die für die Standardzielgruppe in den Laufzeittabellen und in den Unica Campaign-Systemtabellen erstellt werden.

Tabelle 5. Angebotsattribute

UACI_CHOfferAttrib Unica Interact Spaltenname der Staging-Tabelle im Kontaktprotokoll	Unica Campaign Kontaktprotokolltabelle	Name der Tabellenspalte
ContactID	Nicht zutreffend	Nicht zutreffend
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

Zuordnung der Staging-Tabelle "UACI_RHStaging" für den Antwortverlauf

Diese Tabelle zeigt die Zuordnung der Staging-Tabelle `UACI_CHStaging` der Laufzeitumgebung zu der Unica Campaign-Kontaktverlaufstabelle. Die angezeigten Tabellennamen sind die Beispieltabellen, die für die Standardzielgruppe in den Laufzeittabellen und in den Unica Campaign-Systemtabellen erstellt werden.

Tabelle 6. Antwortverlauf

UACI_RHStaging Unica Interact Spaltenname der Staging-Tabelle im Antwortverlauf	Unica Campaign Antwortverlaufstabelle	Name der Tabellenspalte
SeqNum	Nicht zutreffend	Nicht zutreffend
TreatmentCode	UA_ResponseHistory	TreatmentInstID
Kunden-ID	UA_ResponseHistory	Kunden-ID
ResponseDate	UA_ResponseHistory	ResponseDateTime

Tabelle 6. Antwortverlauf (Fortsetzung)

UACI_RHStaging Unica Interact Spaltenname der Staging-Tabelle im Antwortverlauf	Unica Campaign Antwortverlaufstabelle	Name der Tabellenspalte
ResponseType	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

`SeqNum` ist ein Schlüssel, der im Modul für den Kontakt- und Antwortverlauf zum Identifizieren von Daten verwendet, aber nicht in den Unica Campaign-Antworttabellen aufgezeichnet wird. Die Spalte `userDefinedFields` kann beliebige Daten Ihrer Wahl enthalten.

Zusätzliche Spalten in Staging-Tabellen

Wenn Sie den Staging-Tabellen weitere Spalten hinzufügen, werden diese vom Modul für den Kontakt- und Antwortverlauf in den Tabellen `UA_DtlContactHist` oder `UA_ResponseHistory` in die Spalten mit dem gleichen Namen geschrieben.

Beispiel: Wenn Sie der Tabelle `UACI_CHStaging` die Spalte `linkFrom` hinzufügen, kopiert das Modul für den Kontakt- und Antwortverlauf diese Daten in die Spalte `linkFrom` in der Tabelle `UA_DtlContactHist`.

Zusätzliche Spalten in Kontakt- und Antwortverlaufstabellen von Unica Campaign

Wenn Sie über zusätzliche Spalten in Ihren Kontakt- und Antwortverlaufstabellen von Unica Campaign verfügen, müssen Sie den Staging-Tabellen die entsprechenden Spalten hinzufügen, bevor Sie das Modul für den Kontakt- und Antwortverlauf ausführen.

Um die zusätzlichen Spalten in den Staging-Tabellen auszufüllen, erstellen Sie Spalten mit den gleichen Namen wie Ihre Name/Wert-Paare in den Daten der Laufzeitsitzung.

So können Sie beispielsweise die Name/Wert-Paare `NumberItemsInWishList` und `NumberItemsInShoppingCart` erstellen und diese Ihrer `UACI_RHStaging-Tabelle` hinzufügen. Wenn ein Ereignis zum Protokollieren der Annahme oder der Ablehnung eines Angebots eintritt, füllt die Laufzeitumgebung diese Felder aus. Die Laufzeitumgebung füllt die Tabelle `UACI_CHStaging` aus, wenn ein Ereignis zum Protokollieren eines Angebotskontakts auftritt.

Einbeziehen einer Bewertung für ein Angebot mithilfe von Tabellen

Sie können die benutzerdefinierten Felder verwenden, um die Bewertung einzuschließen, die zur Präsentation eines Angebots verwendet wird. Fügen Sie eine Spalte mit dem Namen `FinalScore` sowohl der Tabelle `UACI_CHStaging` in den Laufzeittabellen als auch der Tabelle `UA_DtlContactHist` in den Unica Campaign-Systemtabellen hinzu. Unica Interact füllt die Spalte `FinalScore` automatisch mit dem für das Angebot verwendeten Endergebnis, wenn Sie integriertes Lernen verwenden.

Wenn Sie ein benutzerdefiniertes Lernmodul aufbauen, können Sie die `setActualValueUsed`-Methode der `ITreatment`-Benutzeroberfläche und die `logEvent`-Methode der `ILearning`-Benutzeroberfläche verwenden.

Wenn Sie kein Lernmodul verwenden, fügen Sie sowohl der Tabelle `UACI_CHStaging` in den Laufzeittabellen als auch der Tabelle `UA_DtlContactHist` in den Unica Campaign-Systemtabellen eine Spalte mit dem Namen `Score` hinzu. Unica Interact füllt die Spalte Score automatisch mit dem für das Angebot verwendeten `Score`.

Erstellen von neuen Verlaufstabellen in Unica Campaign und von Staging-Tabellen in Unica Interact

Wenn Sie eine andere Zielgruppenebene als `Kunde` verwenden, müssen Sie in Unica Campaign neue Verlaufstabellen und in Unica Interact neue Staging-Tabellen erstellen.

Beispielsweise wird das unten stehende Beispielscript in der IBM DB2®-Designzeitdatenbank verwendet, um Verlaufstabellen in Unica Campaign für eine Zielgruppenebene des Typs `Konto` zu erstellen.

```
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;
```

Beispielsweise wird das unten stehende Beispielscript in der IBM DB2®-Designzeitdatenbank verwendet, um Verlaufs-Staging-Tabellen in Unica Interact für eine Zielgruppenebene des Typs `Konto` zu erstellen.

```
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);
```

Konfigurieren der JMX-Überwachung für das Kontakt- und Antwortverlaufsmodul

Verwenden Sie dieses Verfahren, um die JMX-Überwachung für das Kontakt- und Antwortverlaufsmodul zu konfigurieren. Die JMXMP- und RMI-Protokolle werden unterstützt. Die Konfiguration der JMX-Überwachung bedeutet nicht, dass die Sicherheit für das Kontakt- und Antwortverlaufsmodul aktiviert wird. Die Konfiguration der JMX-Überwachung erfolgt über Unica Platform für die Designumgebung.

Zur Verwendung Ihres JMX-Überwachungstools für das Kontakt- und Antwortverlaufsmodul wird folgende Standardadresse verwendet:

- JMXMP-Protokoll: `service:jmx:jmxmp://CampaignServer:port/campaign`.
- RMI-Protokoll: `service:jmx:rmi:///jndi/rmi://CampaignServer:port/campaign`.

Wenn Sie die Daten in Ihrem JMX-Überwachungstool anzeigen, werden die Ergebnisattribute zuerst nach Partition und dann nach Zielgruppenebene organisiert.

Bearbeiten Sie in Unica Platform für die Designumgebung folgende Konfigurationseigenschaften in der Kategorie `Campaign > Überwachung`.

Konfigurationseigenschaft	Einstellung
`monitorEnabledForInteract`	**True**
Port	Die Portnummer für den JMX-Service
protocol	Das zu verwendende Protokoll: • **JMXMP** • **RMI** Es ist keine Sicherheit für das Kontakt- und Antwortverlaufsmodul aktiviert, selbst wenn Sie das JMXMP-Protokoll auswählen.

Informationen zur sitzungsübergreifenden Antwortverfolgung

Die Besucher schließen möglicherweise nicht bei jedem einzelnen Touchpoint-Besuch immer eine Transaktion ab. Ein Kunde kann zum Beispiel einen Artikel im Warenkorb hinzufügen, aber den Einkauf erst zwei Tage später abschließen. Dennoch ist es nicht sinnvoll, die Laufzeitsitzung für unbestimmte Zeit aktiv zu lassen. Sie können die sitzungsübergreifende Antwortverfolgung aktivieren, um die Präsentation eines Angebots in einer Sitzung zu verfolgen und mit einer Antwort in einer anderen Sitzung abzugleichen.

Die sitzungsübergreifende Antwortverfolgung in Unica Interact kann standardmäßig nach Verfahrenscodes oder Angebotscodes abgleichen. Sie können die sitzungsübergreifende

Antwortverfolgung jedoch auch so konfigurieren, dass ein beliebiger Alternativcode abgeglichen wird. Dabei werden alle verfügbaren Daten und Antworten sitzungsübergreifend abgeglichen. Beispiel: Ihre Website enthält ein Angebot mit einem Werbecode, der eine Woche lang gültig ist und zu dem Zeitpunkt generiert wird, an dem der Rabattartikel angezeigt wird. Ein Benutzer kann den Artikel im Warenkorb hinzufügen, aber den Kauf erst drei Tage später abschließen. Wenn Sie den `postEvent`-Aufruf verwenden, um ein Annahmeereignis zu protokollieren, können Sie nur den Aktionscode einbeziehen. Da die Laufzeit in der aktuellen Sitzung keinen übereinstimmenden Verfahrens- oder Angebotscode finden kann, wird das Annahmeereignis mit allen verfügbaren Informationen in der Staging-Tabelle für die sitzungsübergreifende Antwort (XSessResponse) gespeichert. Der CrossSessionResponse-Service liest die XSessResponse-Tabelle regelmäßig aus und versucht, die Datensätze mit den verfügbaren Kontaktverlaufsdaten abzugleichen. Der CrossSessionResponse-Service gleicht den Aktionscode mit dem Kontaktverlauf ab und erfasst alle erforderlichen Daten, die zum Protokollieren einer ordnungsgemäßen Antwort benötigt werden. Anschließend schreibt der CrossSessionResponse-Service die Antwort in die Staging-Tabellen und (sofern aktiviert) in die Lerntabellen. Das Modul für den Kontakt- und Antwortverlauf schreibt die Antwort dann in die Unica Campaign Kontakt- und Antwortverlaufstabellen. Die erfolgreiche Verarbeitung der sitzungsübergreifenden Antwort hängt von den Verlaufsdatensätzen des Erstkontakts ab, die durch den ETL-Prozess des Kontaktverlaufs zu Unica Campaign migriert wurden.

Erkennung und Unterdrückung von Duplikaten aktivieren

Fügen Sie der Interact Laufzeit den JVM Parameter hinzu, um die Duplikatsprüfung von Antworten und sitzungsübergreifenden Antworten zu aktivieren.

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

Hier `<max records>` beträgt die maximale Anzahl eindeutiger Datensätze, die zur Überprüfung von Duplikaten berücksichtigt werden. Diese Überprüfung wird deaktiviert, wenn der Wert 0 beträgt, was der Standardwert ist.

Darüber hinaus kann auch der folgende JVM Parameter zur Interact Laufzeit hinzugefügt werden. Wenn der Wert auf "true" gesetzt wird, werden doppelte Antworten und sitzungsübergreifende Antworten unterdrückt. Standardmäßig ist es deaktiviert.

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

Sobald die Duplikatsprüfung aktiviert ist, wird eine Warnmeldung mit den Informationen zu den Duplikaten in interact.log protokolliert. Diese Überprüfung findet in den folgenden zwei Stufen im Code statt.

- Nachdem Interact das Antwortereignis verarbeitet hat, aber bevor es dem Speichercache hinzugefügt wird.
- Wenn Interact das Antwortereignis in der Staging-Tabelle speichert.

Darüber hinaus wird der JMX bean com.unicacorp.interact:type=Services,group=Response History Memory Cache Statistics eine neue Eigenschaft `CacheInfo` hinzugefügt. Wenn die Duplikatsprüfung aktiviert wird, werden Informationen zum Speichercache im folgenden Format zurückgegeben.

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

Zum Beispiel:

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

Nur die duplizierten Einträge werden in die Nachricht aufgenommen.

Dies ist der Aktivierung der Duplikatsprüfung von Kontakten ähnlich; fügen Sie der Interact Laufzeit den folgenden JVM Parameter hinzu. Die folgenden JVM Parameter sind erforderlich.

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

Die betroffene JMX Bean ist com.unicacorp.interact:type=Services,group=Contact History Memory Cache Statistics

Sitzungsübergreifender Antwortprozess

Der sitzungsübergreifende Antwortprozess beginnt mit der Initialisierung von Interact. Es verarbeitet Datensätze in der sitzungsübergreifenden Antwort-Inszenierungstabelle. Der Prozess fragt die Tabelle nach einem konfigurierbaren Zeitintervall immer wieder nach neuen Datensätzen oder Wiederholungsversuchen zur Verarbeitung ab.

`Affinium|interact|services|crossSessionResponse|xsessionProcessIntervalInSecs`. Erfolgreiche Datensätze werden in der Tabelle bereinigt. Nicht erfolgreiche Datensätze werden für einen bestimmten Zeitraum zur Wiederholung markiert. `Affinium|interact|services|crossSessionResponse|purgeOrphanResponseThresholdInMinutes`. Datensätze werden nach mehreren Versuchen nicht erfolgreich verarbeitet.`Affinium|interact|services|crossSessionResponse|purgeOrphanResponseThresholdInMinutes`. Minuten nach der Antwortzeit des Datensatzes werden diese Datensätze als "fehlgeschlagen" markiert.

Zur Unterstützung der mehrfachen Laufzeitinstanzen für die Arbeit an denselben Daten aktualisiert jeder sitzungsübergreifende Antwortprozess die Anzahl der `xsessionResponseBatchSize`-Datensätze auf "In Bearbeitung" mit einem eindeutigen Wert für die Markierungsspalte, der nur diesem Prozess bekannt ist. Der Prozess versucht, diese Datensätze mit den verfügbaren Kontakthistoriendaten abzugleichen, indem er die systemdefinierten SQLs für Übereinstimmung `byTreatmentCode` und `byOfferCode` verwendet. Für die Datensätze, bei denen eine Übereinstimmung gefunden wird, löst das System eine Protokollantwort aus, und die Markierungsspalte wird wieder auf "Erfolg" aktualisiert.

Anmerkung: Die Aktualisierung der Markierungsspalte mit einem prozessspezifischen eindeutigen Wert ist nur dann anwendbar, wenn der Benutzer keine "Override SQL" konfiguriert hat. Wenn die Option "Override SQL" zur Definition der übereinstimmenden Abfrage verwendet wird, nimmt der sitzungsübergreifende Antwortprozess die exklusive Datenbanksperre auf der sitzungsübergreifenden Antwort-Staging-Tabelle, verarbeitet die Datensätze und hebt die Sperre auf.

Wenn eine große Anzahl unverarbeiteter Datensätze in der xSessionResponse-Tabelle verfügbar ist, kann es zu Leistungsproblemen kommen, wenn der CrossSessionResponse-Dienst versucht, alle Datensätze auf einmal zu verarbeiten.

Um die Leistung zu verbessern, können Benutzer `Affinium|interact|services|crossSessionResponse|xsessionResponseBatchSize` auf einen positiven ganzzahligen Wert festlegen. Der CrossSessionResponse-Dienst verarbeitet jeweils `xsessionResponseBatchSize`-Datensätze und durchläuft die xSessionResponse-Tabelle, bis alle neuen oder wiederholten Datensätze verarbeitet sind.

Konfiguration der Quellendaten für die sitzungsübergreifende Antwortverfolgung

Unica Interact Mit der sitzungsübergreifenden Antwortverfolgung können Sie Sitzungsdaten aus der Laufzeitumgebung mit dem Unica Campaign Kontakt- und Antwortverlauf abgleichen. Standardmäßig verwendet die sitzungsübergreifende Antwortverfolgung den Verfahrenscode oder den Angebotscode zum Abgleich. Sie können die Laufzeitumgebung so konfigurieren, dass ein anderer, benutzerdefinierter Code zum Abgleich verwendet wird.

- Wenn Sie einen alternativen Code abgleichen möchten, müssen Sie den alternativen Code in der `UACI_TrackingType`-Tabelle in den Unica Interact-Laufzeittabellen definieren.
- Die Laufzeitumgebung benötigt Zugriff auf die Unica Campaign Kontaktverlaufstabellen. Um dies sicherzustellen, können Sie entweder die Laufzeitumgebung für den Zugriff auf die Unica Campaign Kontaktverlaufstabellen konfigurieren oder eine Kopie der Kontaktverlaufstabellen in der Laufzeitumgebung erstellen.

  Dieser Zugriff erfolgt schreibgeschützt und unabhängig vom Dienstprogramm für den Kontakt- und Antwortverlauf.

  Wenn Sie eine Kopie der Tabellen erstellen, liegt es in Ihrer Verantwortung, sicherzustellen, dass die Daten in der Kopie des Kontaktverlaufs korrekt sind. Mit der `purgeOrphanResponseThresholdInMinutes`-Eigenschaft können Sie die Dauer konfigurieren, wie lange der CrossSessionResponse-Service Antworten beibehalten soll, die noch nicht abgeglichen wurden, und wie oft die Daten in der Kopie der

Kontaktverlaufstabellen aktualisiert werden. Wenn Sie das Modul für den Kontakt- und Antwortverlauf verwenden, sollten Sie die Aktualisierungen des ETL-Prozesses koordinieren, um sicherzustellen, dass die Daten stets auf dem neuesten Stand sind.

Konfigurieren von Kontakt- und Antwortverlaufstabellen für die sitzungsübergreifende Antwortverfolgung

Unabhängig davon, ob Sie eine Kopie der Kontaktverlaufstabellen erstellen oder die tatsächlichen Tabellen in den Unica Campaign-Systemtabellen verwenden, müssen Sie die folgenden Schritte durchführen, um die Kontakt- und Antwortverlaufstabellen zu konfigurieren.

Die Kontakt- und Antwortverlaufstabellen müssen vor Ausführen dieser Schritte in Unica Campaign korrekt zugeordnet werden.

1. Führen Sie das SQL-Script `aci_lrnfeature` im Verzeichnis `interactDT/ddl/acifeatures` im Installationsverzeichnis der Unica Interact-Designumgebung für die Tabellen `UA_DtlContactHist` und `UA_ResponseHistory` in den Unica Campaign-Systemtabellen aus.

   Dadurch wird den Tabellen `UA_DtlContactHist` und `UA_ResponseHistory` die Spalte `RTSelectionMethod` hinzugefügt. Führen Sie das Script `aci_lrnfeature` für diese Tabellen für jede einzelne Zielgruppenebene aus. Bearbeiten Sie das Script, falls erforderlich, um stets mit der korrekten Tabelle für jede Zielgruppenebene zu arbeiten.
2. Wenn Sie die Kontaktverlaufstabellen in die Laufzeitumgebung kopieren möchten, tun Sie das jetzt.

   Wenn Sie eine Kopie der Unica Campaign-Kontaktverlaufstabellen erstellen möchten, die die Laufzeitumgebung aufrufen kann, um die sitzungsübergreifende Antwortverfolgung zu unterstützen, beachten Sie die folgenden Richtlinien:

   - Die sitzungsübergreifende Antwortverfolgung benötigt Lesezugriff auf diese Tabellen.
   - Die sitzungsübergreifende Antwortverfolgung benötigt die folgenden Tabellen aus dem Unica Campaign-Kontaktprotokoll.

- `UA_DtlContactHist` (für jede Zielgruppenebene)
- `UA_Treatment`

Sie müssen die Daten in diesen Tabellen regelmäßig aktualisieren, um die korrekte Antwortverfolgung sicherzustellen.

3. Führen Sie das SQL-Script `aci_crhtab` im Verzeichnis `ddl` im Installationsverzeichnis der Unica Interact-Laufzeitumgebung für die Quellendaten des Kontakt- und Antwortverlaufs aus.

   Dieses Script erstellt die Tabellen `UACI_XsessResponse` und `UACI_CRHTAB_Ver`.
4. Erstellen Sie für jede Zielgruppenebene eine Version der `UACI_XsessResponse`-Tabelle.

Um die Performance der sitzungsübergreifenden Antwortverfolgung zu verbessern, können Sie die Menge der Kontaktverlaufsdaten einschränken, indem Sie entweder die Kontaktverlaufsdaten kopieren oder eine entsprechende Ansicht in den Unica Campaign-Kontaktverlaufstabellen konfigurieren. Beispiel: Wenn in Ihren geschäftsrelevanten Prozessen und Verfahren geregelt ist, dass ein Angebot maximal 30 Tage lang gültig sein kann, sollten Sie die Kontaktverlaufsdaten auf die letzten 30 Tage beschränken. Wenn Sie die Anzahl der Tage der zu verwaltenden Kontaktverlaufsdaten ändern wollen, öffnen Sie die Konfigurationseigenschaft **Campaign | Partitionen | Partition n| Interact | contactAndResponseHistTracking** und setzen Sie den Wert auf **daysBackInHistoryToLookupContact**.

Es werden keine Ergebnisse aus der sitzungsübergreifenden Antwortverfolgung angezeigt, bevor das Modul für den Kontakt- und Antwortverlauf aufgerufen wird. Beispiel: Der Standardwert für `processSleepIntervalInMinutes` beträgt 60 Minuten. Es kann daher mindestens eine Stunde dauern, bis sitzungsübergreifende Antworten im Unica Campaign Antwortverlauf angezeigt werden.

UACI_TrackingType-Tabelle

Die`UACI_TrackingType`-Tabelle ist Teil der Laufzeitumgebungstabellen. Diese Tabelle definiert die Verfolgungscodes, die mit der sitzungsübergreifenden Antwortverfolgung verwendet werden. Der Verfolgungscode definiert, welche Methode die Laufzeitumgebung

verwendet, um das aktuelle Angebot in einer Laufzeitsitzung mit dem Kontakt- und Antwortverlauf abzugleichen.

Spalte	Typ	Syntax
`TrackingCodeType`	int	Eine Zahl, die den Verfolgungscodetyp darstellt. Auf diese Zahl wird von den SQL-Befehlen verwiesen, die verwendet werden, um die Informationen aus den Sitzungsdaten mit den Kontakt- und Antwortverlaufstabellen abzugleichen.
`Name`	varchar(64)	Der Name für den Verfolgungscodetyp. Dieser wird an die Sitzungsdaten übergeben, indem der reservierte Parameter `UACI_TrackingCodeType` mit der Methode `postEvent` verwendet wird.
`Beschreibung`	varchar(512)	Eine Kurzbeschreibung des Verfolgungscodetyps. Dieses Feld ist optional.

Standardmäßig sind für die Laufzeitumgebung zwei Verfolgungscodetypen definiert, wie in der folgenden Tabelle dargestellt. Für jeden anderen Code müssen Sie einen eindeutigen `TrackingCodeType`definieren.

TrackingCodeType	Name	Syntax
1	Verfahrenscode	UACI generierter Verfahrenscode
2	Angebotscode	UAC Kampagnenangebotscode

UACI_XSessResponse

Die `UACI_XSessResponse`-Tabelle ist Teil der Laufzeitumgebungstabellen. Diese Tabelle wird für die sitzungsübergreifende Antwortverfolgung verwendet.

Für jede Zielgruppenebene muss eine Instanz dieser Tabelle in der Datenquelle mit dem Kontakt- und Antwortverlauf vorhanden sein, der für die Unica Interact sitzungsübergreifende Antwortverfolgung verfügbar ist.

Spalte	Typ	Syntax
`SeqNumber`	bigint	Kennung für die Datenzeile. Der CrossSessionResponse-Service verarbeitet alle Datensätze in der `SeqNumber`-Reihenfolge.
`ICID`	bigint	ID des interaktiven Kanals
`AudienceID`	bigint	Die Zielgruppen-ID für diese Zielgruppenebene. Der Name dieser Spalte muss mit der in Unica Campaign definierten Zielgruppen-ID übereinstimmen. Die Beispieltabelle enthält die Spalte `CustomerID`.
`TrackingCode`	varchar(64)	Der Wert, der vom `UACIOfferTrackingCode`-Parameter der `postEvent`-Methode übergeben wird.
`TrackingCodeType`	int	Die numerische Darstellung des Verfahrenscodes. Der Wert muss ein gültiger Eintrag in der Tabelle `UACI_TrackingType` sein.
`OfferID`	bigint	Die in Unica Campaign definierte Angebots-ID.
`ResponseType`	int	Der Antworttyp für diesen Datensatz. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` sein.
`ResponseTypeCode`	varchar(64)	Der Antworttypcode für diesen Datensatz. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` sein.
`ResponseDate`	datetime	Das Datum der Antwort.
`Markierung`	bigint	Der Wert dieses Feldes identifiziert den Status des Datensatzes.

Spalte	Typ	Syntax
		• 1 - In Bearbeitung. Es ist anwendbar, wenn die "Override SQL"-Konfiguration verwendet wird. In diesem Fall wird die vom Benutzer definierte passende Abfrage verwendet und nicht die vom System generierte SQL. • Random Big Integer-Wert - Er ist anwendbar, wenn "Use System generated SQL" eingestellt ist oder die übereinstimmende Standardabfrage verwendet wird. Das System aktualisiert die zu verarbeitenden Datensätze mit einem eindeutigen großen ganzzahligen Wert, um die von diesem Thread zu verarbeitenden Datensätze zu identifizieren. • 2 - Erfolgreich. Es ist anwendbar, wenn eine Übereinstimmung in der Kontakthistorie vorliegt und die Protokollantwort erfolgreich ausgeführt wird. • NULL - Neue Datensätze • 0 - Erneut versuchen. Es ist anwendbar, wenn keine Übereinstimmung in der Kontakthistorie vorliegt und der Datensatz weniger als `purgeOrphanResponseThresholdInMinutes` Minuten in der Datenbank gespeichert ist. • -1 - Datensatz befindet sich seit mehr als `purgeOrphanResponseThresholdInMinutes` Minuten in der Datenbank. Im Rahmen der Wartung dieser Tabelle durch den Datenbankadministrator können Sie mit diesem Feld nach Datensätzen suchen, die nicht abgegli-

Spalte	Typ	Syntax
		chen werden, das sind alle Datensätze mit dem Wert -1. Alle Datensätze mit dem Wert 2 werden automatisch vom CrossSessionResponse-Service entfernt.
`UsrDefinedFields`	char(18)	Alle benutzerdefinierten Felder, die Sie beim Abgleichen von Angebotsantworten im Kontakt- und Antwortverlauf einschließen möchten. Wenn Sie zum Beispiel einen Aktionscode abgleichen möchten, müssen Sie ein benutzerdefiniertes Feld für den Aktionscode einschließen.

Aktivieren der sitzungsübergreifenden Antwortverfolgung

Mit diesem Verfahren können Sie die sitzungsübergreifende Antwortverfolgung aktivieren.

Um die sitzungsübergreifende Antwortverfolgung optimal nutzen zu können, müssen Sie das Modul für den Kontakt- und Antwortverlauf konfigurieren.

Um die sitzungsübergreifende Antwortverfolgung zu verwenden, müssen Sie die Laufzeitumgebung für den Lesezugriff auf die Kontakt- und Antwortverlaufstabellen in Unica Campaign konfigurieren. Sie können entweder die tatsächlichen Unica Campaign Kontakt- und Antwortverlaufstabellen in der Designumgebung oder eine Kopie der Tabellen in den Datenquellen der Laufzeitumgebung lesen. Die Konfiguration der Laufzeitumgebung für den Lesezugriff auf die Kontakt- und Antwortverlaufstabelle erfolgt unabhängig von der Konfiguration eines eventuellen Kontakt- und Antwortverlaufsmoduls.

Wenn Sie etwas anderes als den Verfahrens- oder Angebotscode zum Abgleichen verwenden möchten, müssen Sie der Tabelle `UACI_TrackingType` den entsprechenden Bezug hinzufügen.

1. Erstellen Sie die XSessResponse-Tabellen in den Kontakt- und Antwortverlaufstabellen, die die Laufzeitumgebung aufrufen kann.
2. Definieren Sie die Eigenschaften in der Kategorie `contactAndResponseHistoryDataSource` für die Laufzeitumgebung.
3. Definieren Sie die Eigenschaft `crossSessionResponseTable` für jede Zielgruppenebene.
4. Erstellen Sie für jede Zielgruppenebene eine Kategorie `OverridePerAudience`.

Sitzungsübergreifender Abgleich von Angebot und Antwort

Standardmäßig verwendet die sitzungsübergreifende Antwortverfolgung die Verfahrenscodes oder die Angebotscodes zum Abgleich. Der crossSessionResponse-Service verwendet SQL-Befehle, um Verfahrenscodes, Angebotscodes oder einen benutzerdefinierten Code aus den Sitzungsdaten mit den Unica Campaign Kontakt- und Antwortverlaufstabellen abzugleichen. Sie können diese SQL-Befehle bearbeiten, um alle Anpassungen abzugleichen, die Sie an den Verfolgungscodes, Angebotscodes oder angepassten Codes vorgenommen haben.

Abgleich nach Verfahrenscode

Die SQL zum Abgleich nach Verfahrenscode muss alle Spalten in der XSessResponse-Tabelle für diese Zielgruppenebene plus eine Spalte mit dem Namen `OfferIDMatch` zurückgeben. Der Wert in der Spalte `OfferIDMatch` muss mit der `offerId`-Kennung identisch sein, die dem Verfahrenscode im Datensatz XSessResponse entspricht.

Im Folgenden ist ein Beispiel für den standardmäßig generierten SQL-Befehl, der mit den Verfahrencodes übereinstimmt. Unica Interactgeneriert die SQL, um die korrekten Tabellennamen für die Zielgruppenebene zu verwenden. Diese SQL wird verwendet, wenn die Eigenschaft `Interact > Services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byTreatmentCode > SQL` auf **Systemgenerierte SQL verwenden** gesetzt ist.

```
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
```

Die Werte `UACI_XsessResponse`, `UA_DtlContactHist`, `CustomerID`und `UA_ContactHistory` werden von den Einstellungen in Unica Interact definiert. Beispiel: `UACI_XsessResponse` wird von der Konfigurationseigenschaft `Interact > Profil > Zielgruppenebenen > [AudienceLevelName] > crossSessionResponseTable` definiert.

Wenn Sie die Kontakt- und Antwortverlaufstabellen angepasst haben, müssen Sie diese SQL möglicherweise überarbeiten, um mit den Tabellen arbeiten zu können. SQL-Überschreibungen werden in der Eigenschaft `Interact > Services > crossSessionResponse > OverridePerAudience > (AudienceLevel) TrackingCodes > byTreatmentCode > OverrideSQL` definiert. Wenn Sie die SQL überschreiben, müssen Sie auch die Eigenschaft `SQL` in **SQL überschreiben** ändern.

Abgleich nach Angebotscode

Die SQL zum Abgleich nach Angebotscode muss alle Spalten in der XSessResponse-Tabelle für diese Zielgruppenebene plus eine Spalte mit dem Namen `TreatmentCodeMatch` zurückgeben. Der Wert in der Spalte `TreatmentCodeMatch` ist der Verfahrenscode, der der Angebots-ID (und dem Angebotscode) im Datensatz XSessResponse entspricht.

Im Folgenden ist ein Beispiel für den standardmäßig generierten SQL-Befehl, der mit den Verfahrencodes übereinstimmt. Unica Interactgeneriert die SQL, um die korrekten Tabellennamen für die Zielgruppenebene zu verwenden. Diese SQL wird verwendet, wenn die Eigenschaft `Interact > Services > crossSessionResponse > OverridePerAudience > AudienceLevel > TrackingCodes > byOfferCode > SQL` auf **Systemgenerierte SQL verwenden** gesetzt ist.

```
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
```

Die Werte `UACI_XsessResponse`, `UA_DtlContactHist`, `CustomerID`und `UA_ContactHistory` werden von den Einstellungen in Unica Interact definiert. Beispiel: `UACI_XsessResponse` wird von der Konfigurationseigenschaft `Interact > Profil > Zielgruppenebenen > [AudienceLevelName] > crossSessionResponseTable` definiert.

Wenn Sie die Kontakt- und Antwortverlaufstabellen angepasst haben, müssen Sie diese SQL möglicherweise überarbeiten, um mit den Tabellen arbeiten zu können. SQL-Überschreibungen werden in der Eigenschaft `Interact > Services > crossSessionResponse > OverridePerAudience > (AudienceLevel) TrackingCodes > byOfferCode > OverrideSQL` definiert. Wenn Sie die SQL überschreiben, müssen Sie auch die Eigenschaft `SQL` in **SQL überschreiben** ändern.

Abgleich nach alternativem Code

Sie können einen SQL-Befehl definieren, um einen Abgleich nach einem beliebigen Alternativcode durchzuführen. So können Sie zum Beispiel unabhängig von den Angebots- und Verfahrenscode auch zusätzliche Werbe- und Produktcodes definieren.

Dieser Alternativcode muss in der Tabelle `UACI_TrackingType` in den Unica Interact Tabellen der Laufzeitumgebung definiert werden.

Sie müssen eine SQL oder eine gespeicherte Prozedur in der Eigenschaft `Interact > Services > crossSessionResponse > OverridePerAudience > (AudienceLevel) > TrackingCodes > byAlternateCode > OverrideSQL` angeben, die alle Spalten in der Tabelle XSessResponse für diese Zielgruppenebene plus die Spalten `TreatmentCodeMatch` und `OfferIDMatch` zurückgibt. Optional können Sie auch `offerCode` anstelle von `OfferIDMatch` zurückgeben (in der Form `offerCode1`, `offerCode2`, `offerCode`*N* für N-teilige Angebotscodes). Die Werte in den Spalten `TreatmentCodeMatch` und `OfferIDMatch` (oder in Spalten mit dem Angebotscode) müssen dem `TrackingCode` im Datensatz XSessResponse entsprechen.

Beispiel: Der folgende SQL-Pseudocode verwendet zum Abgleich die Spalte `AlternateCode` in der Tabelle 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>
```

Dabei ist <x> der in der Tabelle `UACI_TrackingType` definierte Verfolgungscode.

Verwenden eines Datenbankladedienstprogramms mit der Laufzeitumgebung

Standardmäßig schreibt die Laufzeitumgebung die Protokolldaten für Kontakte und Antworten aus den Sitzungsdaten in Staging-Tabellen. Auf einem sehr aktiven

Produktionssystem kann die benötigte Speichermenge, die zum Zwischenspeichern aller Daten erforderlich ist, bevor diese in die Staging-Tabellen geschrieben werden können, jedoch ein Problem darstellen. Sie können die Laufzeit konfigurieren und ein Datenbankladedienstprogramm verwenden, um die Leistung zu verbessern.

Wenn Sie ein Datenbankladedienstprogramm aktivieren, fixiert die Laufzeit den gesamten Kontakt- und Antwortverlauf nicht im Speicher, sondern schreibt die Daten stattdessen in eine Staging-Datei, bevor in die Staging-Tabellen geschrieben wird. Verwenden Sie die `externalLoaderStagingDirectory`-Eigenschaft, um die Position für das Verzeichnis zu definieren, das die Staging-Dateien enthält. Dieses Verzeichnis enthält mehrere Unterverzeichnisse. Das erste Unterverzeichnis ist das Verzeichnis der Laufzeitinstanz mit den Verzeichnissen `contactHist` und `respHist`. Die Verzeichnisse `contactHist` und `respHist` enthalten eindeutig benannte Unterverzeichnisse im Format audienceLevelName.uniqueID.currentState, in denen die Staging-Dateien enthalten sind.

Current® Status	Syntax
CACHE	Verzeichnisinhalte werden gerade in eine Datei geschrieben.
READY	Verzeichnisinhalte sind zur Verarbeitung bereit.
RUN	Verzeichnisinhalte werden gerade in die Datenbank geschrieben.
VERARBEITET	Verzeichnisinhalte wurden in die Datenbank geschrieben.
FEHLER	Fehler beim Schreiben der Verzeichnisinhalte in der Datenbank.
ATTN	Verzeichnisinhalte bedürfen Ihrer Aufmerksamkeit. Vermutlich sind manuelle Maßnahmen erforderlich, um den Schreibvorgang in der Datenbank abzuschließen.
RERUN	Verzeichnisinhalte sind zum Schreiben in der Datenbank bereit. Sie sollten die Verzeichnisse `ATTN` oder `ERROR` in `RERUN` umbenennen, nachdem Sie das Problem behoben haben.

Sie können das Verzeichnis der Laufzeitinstanz definieren, indem Sie die JVM-Eigenschaft interact.runtime.instance.name im Startscript des Anwendungsservers

definieren. Sie können dem Startscript Ihres Webanwendungsservers zum Beispiel `-Dinteract.runtime.instance.name=instance2` hinzufügen. Sofern nicht anders angegeben, wird der Standardname `DefaultInteractRuntimeInstance` verwendet.

Das Verzeichnis `samples` enthält Beispieldateien, die Ihnen beim Schreiben Ihrer eigenen Steuerdateien für das Datenbankladedienstprogramm behilflich sind.

Aktivieren eines Datenbankladedienstprogramms mit der Laufzeitumgebung

Verwenden Sie dieses Verfahren, um ein Datenbankladedienstprogramm mit der Laufzeitumgebung zu aktivieren.

Sie müssen Befehls- oder Steuerdateien für Ihr Datenbankladedienstprogramm definieren, bevor Sie die Laufzeitumgebung für ihre Verwendung konfigurieren. Diese Dateien müssen an derselben Position auf allen Laufzeitservern in derselben Servergruppe vorhanden sein.

Unica Interact stellt Beispiel-Befehls- und -Steuerdateien im Verzeichnis `loaderService` in Ihrer Unica Interact-Laufzeitserverinstallation bereit.

1. Überprüfen Sie, ob der Laufzeitumgebungsbenutzer über Anmeldeinformationen für die Laufzeittabellendatenquelle verfügt, die in Ihren Konfigurationseigenschaften unter `Interact > allgemein > systemTablesDataSource` definiert ist.
2. Definieren Sie die Konfigurationseigenschaften von `> allgemein > systemTablesDataSource > loaderProperties`.
3. Definieren Sie die Eigenschaft `Interact > services > externalLoaderStagingDirectory`.
4. Überarbeiten Sie ggf. die Konfigurationseigenschaften `Interact > services > responseHist > fileCache`.
5. Überarbeiten Sie ggf. die Konfigurationseigenschaften `Interact > services > contactHist > fileCache`.
6. Starten Sie den Laufzeitserver erneut.

ETL-Prozess für Ereignismuster

Wenn Sie umfangreiche Unica Interact-Ereignismusterdaten verarbeiten müssen und diese Daten für Abfragen und zu Berichtszwecken zur Verfügung stellen möchten, können Sie zur Erzielung einer optimalen Leistung auf jedem unterstützten Server einen eigenständigen ETL-Prozess (ETL = Extrahieren, Transformieren, Laden) installieren.

In Interact werden alle Ereignismusterdaten für eine bestimmte Zielgruppen-ID als einzelne Sammlung in den Laufzeitdatenbanktabellen gespeichert. Die AudienceID und die Informationen zum Musterzustand werden als großes Binärobjekt (BLOB) gespeichert. Damit auf Basis der Ereignismuster SQL-Abfragen durchgeführt oder Berichte erstellt werden können, ist dieser neue ETL-Prozess erforderlich, um das Objekt in Tabellen in einer Zieldatenbank zu unterteilen. Um dies zu erreichen, nimmt der eigenständige ETL-Prozess Ereignismusterdaten aus den Laufzeitdatenbanktabellen von Unica Interact, verarbeitet diese in dem von Ihnen angegebenen Zeitplan und speichert sie in der Zieldatenbank. Dort stehen die Daten dann für SQL-Abfragen oder zusätzliche Berichterstellungen zur Verfügung.

Neben der Verschiebung von Ereignismusterdaten in die Zieldatenbank und deren dortiger Transformation synchronisiert der eigenständige ETL-Prozess die Daten in der Zieldatenbank mit den aktuellen Informationen in Ihrer Unica Interact-Laufzeitdatenbank. Wenn Sie beispielsweise ein Ereignismuster in der Unica Interact-Laufzeit löschen, werden die verarbeiteten Daten dieses Ereignismusters bei der nächsten Ausführung des ETL-Prozesses aus der Zieldatenbank entfernt. Die Informationen zum Ereignismusterzustand werden ebenfalls auf dem neuesten Stand gehalten. Daher handelt es sich bei den in der Zieldatenbank gespeicherten Informationen zu Ereignismustern nicht um archivierte Daten, sondern ausschließlich um aktuelle Daten.

Ausführen des eigenständigen ETL-Prozesses

Wenn Sie den eigenständigen ETL-Prozess auf einem Server starten, wird er kontinuierlich im Hintergrund ausgeführt, bis er gestoppt wird. Der Prozess befolgt während seines Betriebs die Anweisungen in den Unica Platform-Konfigurationseigenschaften, um die Häufigkeit, Datenbankverbindungen und sonstige Details zu bestimmen.

Stellen Sie sicher, dass Sie die folgenden Tasks abschließen, bevor Sie den eigenständigen ETL-Prozess ausführen:

- Sie müssen über die Rolle eines Interact-Benutzers mit Administratorberechtigung verfügen.
- Sie müssen den Prozess auf einem Server installiert und die Dateien auf dem Server und in Unica Platform für Ihre Konfiguration ordnungsgemäß konfiguriert haben.

Anmerkung:

Wenn Sie den ETL-Prozess unter Microsoft Windows in einer anderen Sprache als in US-amerikanischem Englisch ausführen möchten, legen Sie mithilfe von `chcp` in der Eingabeaufforderung die Codepage für die von Ihnen verwendete Sprache fest. Sie können zum Beispiel einen der folgenden Codes verwenden: ja_jp=932, zh_cn=936, ko_kr=949, ru_ru=1251 und für de_de, fr_fr, it_it, es_es, pt_br, verwenden Sie 1252. Verwenden Sie vor dem Start des ETL-Prozesses den Befehl `chcp` in der Windows-Befehlseingabeaufforderung, um sicherzustellen, dass die Zeichen ordnungsgemäß angezeigt werden.

Sobald Sie den eigenständigen ETL-Prozess installiert und konfiguriert haben, können Sie den Prozess starten.

1. Öffnen Sie eine Eingabeaufforderung auf dem Server, auf dem der ETL-Prozess installiert ist.
2. Navigieren Sie zu dem Verzeichnis `<Interact_home>/PatternStateETL/bin`, das die ausführbaren Dateien für den ETL-Prozess enthält.
3. Fügen Sie zum Ausführen des ETL-Berichts `-Dcom.ibm.interact.logconfiglocation=PatternStateETL/bin/etl_log4j2.xml` hinzu.
4. Führen Sie die Datei `command.bat` (unter Microsoft Windows) bzw. `command.sh` (auf UNIX-ähnlichen Betriebssystemen) mit den folgenden Parametern aus:

- `-u <username>`. Für diesen Wert müssen Sie einen gültigen Unica Platform -Benutzer angeben. Außerdem muss dieser Benutzer mit Zugriff auf die vom ETL-Prozess verwendeten Datenquellen **TargetDS** und **RuntimeDS** konfiguriert worden sein.
- `-p <password>`. Ersetzen Sie <password> durch das Kennwort des von Ihnen angegebenen Benutzers. Wenn das Kennwort für diesen Benutzer leer ist, geben Sie zwei Anführungszeichen an (Beispiel: `-p ""`). Das Kennwort ist bei der Ausführung der Befehlsdatei optional; wenn Sie das Kennwort nicht im Befehl angeben, werden Sie bei der Ausführung des Befehls zu dessen Eingabe aufgefordert.
- `-c <profileName>`. Ersetzen Sie <profileName> durch genau den Namen, den Sie in Unica Platform über die von Ihnen erstellte Konfiguration **Interact | PatternStateETL** angegeben haben.

  Der von Ihnen hier eingegebene Name muss mit dem Wert übereinstimmen, den Sie bei der Erstellung der Konfiguration im Feld **Neuer Kategoriename** angegeben haben.
- `start`. Der Befehl "start" ist zum Starten des Prozesses erforderlich.

Der vollständige Befehl für den Start des Prozesses hat daher also folgendes Format:

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

Der eigenständige ETL-Prozess wird ausgeführt. Er wird im Hintergrund ausgeführt, bis Sie den Prozess stoppen oder der Server erneut gestartet wird.

Anmerkung:

Wenn Sie den Prozess das erste Mal ausführen, kann die Ausführung der kumulierten Ereignismusterdaten eine beträchtliche Zeit in Anspruch nehmen. Die nachfolgenden Ausführungen des Prozesses verwenden nur die neueste Gruppe von Ereignismusterdaten und dauern daher nicht so lange.

Hinweis: Sie können wie im folgenden Beispiel auch das Argument `help` für die Datei `command.bat` bzw. `command.sh` angeben, um alle verfügbaren Optionen anzuzeigen:

```
command.bat help
```

Stoppen des eigenständigen ETL-Prozesses

Wenn Sie den eigenständigen ETL-Prozess auf einem Server starten, wird er kontinuierlich im Hintergrund ausgeführt, bis er gestoppt wird.

1. Öffnen Sie eine Eingabeaufforderung auf dem Server, auf dem der ETL-Prozess installiert ist.
2. Navigieren Sie zu dem Verzeichnis `<Interact_home>/PatternStateETL/bin`, das die ausführbaren Dateien für den ETL-Prozess enthält.
3. Führen Sie die Datei `command.bat` (unter Microsoft Windows) bzw. `command.sh` (auf UNIX-ähnlichen Betriebssystemen) mit den folgenden Parametern aus:
    - `-u <username>`. Für diesen Wert müssen Sie einen gültigen Unica Platform-Benutzer angeben. Außerdem muss dieser Benutzer mit Zugriff auf die vom ETL-Prozess verwendeten Datenquellen **TargetDS** und **RuntimeDS** konfiguriert worden sein.
    - `-p <password>`. Ersetzen Sie <password> durch das Kennwort des von Ihnen angegebenen Benutzers. Wenn das Kennwort für diesen Benutzer leer ist, geben Sie zwei Anführungszeichen an (Beispiel: `-p ""`). Das Kennwort ist bei der Ausführung der Befehlsdatei optional; wenn Sie das Kennwort nicht im Befehl angeben, werden Sie bei der Ausführung des Befehls zu dessen Eingabe aufgefordert.
    - `-c <profileName>`. Ersetzen Sie <profileName> durch genau den Namen, den Sie in Unica Platform über die von Ihnen erstellte Konfiguration **Interact | PatternStateETL** angegeben haben.

      Der von Ihnen hier eingegebene Name muss mit dem Wert übereinstimmen, den Sie bei der Erstellung der Konfiguration im Feld **Neuer Kategoriename** angegeben haben.

- `stop`. Der Befehl "stop" ist zum Stoppen des Prozesses erforderlich. Wenn Sie diesen Befehl verwenden, werden vor der Beendigung des Prozesses alle aktuellen Operationen des ETL-Prozesses abgeschlossen.

  Wenn Sie den ETL-Prozess beenden möchten, ohne auf den Abschluss der aktuellen Operationen zu warten, verwenden Sie den Befehl `forcestop` anstelle von `stop`.

Der vollständige Befehl für den Start des Prozesses hat daher also folgendes Format:

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

Der eigenständige ETL-Prozess wird gestoppt.

Kapitel 5. Services für Angebote

Sie können Unica Interact auf viele Arten konfigurieren, um die Möglichkeiten zu erweitern, wie Angebote zum Präsentieren ausgewählt werden. In den folgenden Abschnitten werden diese optionalen Funktionen im Detail beschrieben.

Angebotsberechtigung

Der Zweck von Unica Interact ist es, auswählbare Angebote zu empfehlen. Einfach gesagt: Unica Interact zeigt die besten Angebote unter den auswählbaren Angeboten an, basierend auf dem Besucher, dem Kanal und der Situation.

Verfahrensregeln sind nur der Anfang davon, wie Unica Interact bestimmt, welche Angebote für einen Kunden in Frage kommen. Unica Interact hat mehrere optionale Funktionen, die Sie implementieren können, um die Art und Weise der Bestimmung der zu präsentierenden Angebote durch die Laufzeitumgebung zu verbessern. Keine dieser Funktionen garantiert, dass einem Kunden ein Angebot präsentiert wird. Diese Funktionen beeinflussen die Wahrscheinlichkeit, dass ein Angebot auswählbar ist, um einem Kunden präsentiert zu werden. Sie können so viele oder so wenige dieser Funktionen verwenden, die Sie benötigen, um die beste Lösung für Ihre Umgebung zu implementieren.

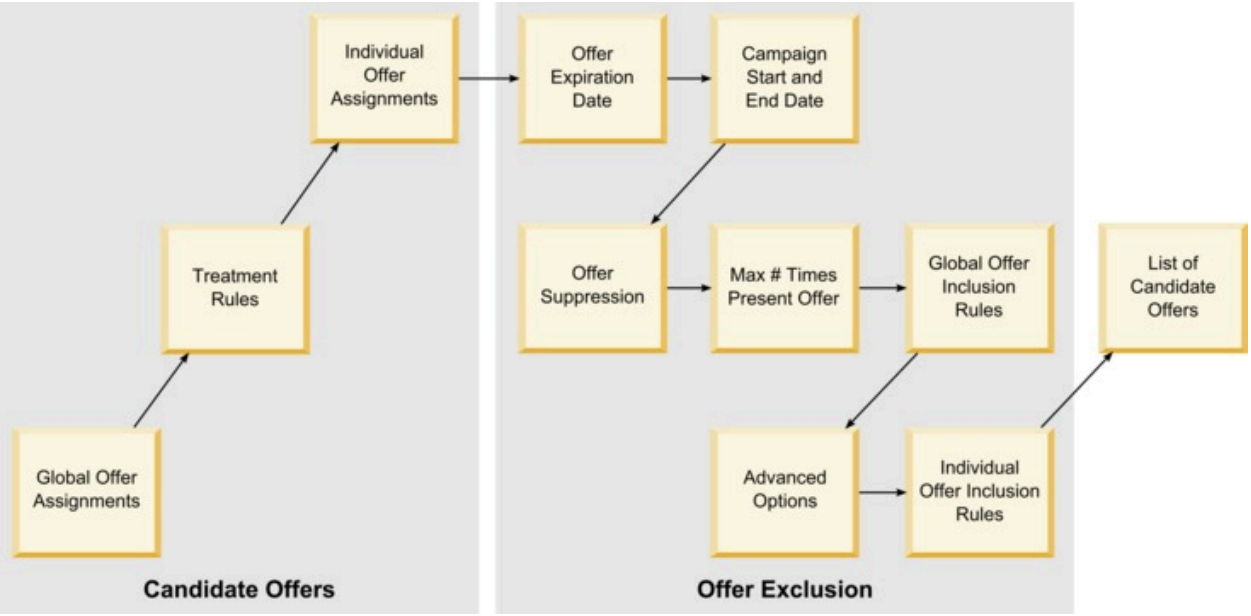
Es gibt drei Hauptbereiche, in denen Sie die Wählbarkeit der Angebote beeinflussen können: die Generierung der Liste von potentiellen Angeboten, die Bestimmung der Marketingbewertung und das Lernen.

Erstellen einer Liste von möglichen Angeboten

Das Erstellen einer Liste von möglichen Angeboten besteht aus zwei Hauptschritten. Der erste Schritt ist die Generierung einer Liste aller möglichen Angebote, für die der Kunde möglicherweise infrage kommt. Der zweite Schritt ist das Herausfiltern von Angeboten, für die der Kunde nicht mehr infrage kommt. Es gibt mehrere Stellen in beiden Schritten, an denen Sie die Generierung der Liste von möglichen Angeboten beeinflussen können.

Dieses Diagramm zeigt die Schritte der Generierung der Liste von möglichen Angeboten. Die Pfeile zeigen den Ausführungsvorrang an. Beispiel: Wenn ein Angebot den Filter

Max. Anzahl Anzeigewiederholungen eines Angebots passiert, aber am Filter **Globale Angebotsaufnahmeregeln** scheitert, schließt die Laufzeitumgebung das Angebot aus.



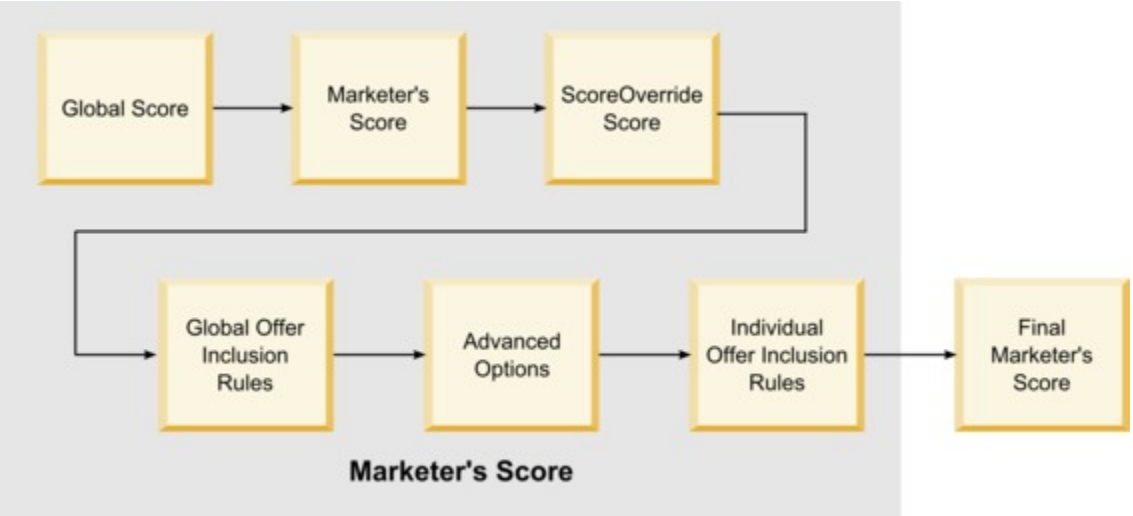
- **Globale Angebotszuweisungen** - Sie können globale Angebote nach Zielgruppenebene mithilfe der globalen Angebotstabelle definieren.
- **Verfahrensregeln** - Die grundlegende Methode, um Angebote nach Segment durch Interaktionspunkt mithilfe der Registerkarte "Interaktionsstrategie" zu definieren.
- **Individuelle Angebotszuweisungen** - Sie können spezielle Angebotszuweisungen nach Kunde mithilfe der Bewertungsüberschreibungstabelle definieren.
- **Angebotsablaufdatum** - Wenn Sie ein Angebot in Unica Campaign erstellen, können Sie ein Ablaufdatum definieren. Wenn das Ablaufdatum für ein Angebot überschritten wurde, schließt die Laufzeitumgebung das Angebot aus.
- **Kampagnenstart- und -enddatum** - Wenn Sie eine Kampagne in Unica Campaign erstellen, können Sie ein Start- und Enddatum für die Kampagne definieren. Wenn das Startdatum für die Kampagne noch nicht eingetreten ist oder das Enddatum für die Kampagne überschritten wurde, schließt die Laufzeitumgebung das Angebot aus.
- **Angebotsunterdrückung** - Sie können eine Angebotsunterdrückung für bestimmte Zielgruppenmitglieder mithilfe der Tabelle für Angebotsunterdrückung definieren.

- **Max. Anzahl Anzeigewiederholungen eines Angebots** - Wenn Sie einen interaktiven Kanal definieren, legen Sie die maximale Häufigkeit fest, mit der ein Angebot einem Kunden pro Sitzung bereitgestellt wird. Wenn ein Angebot bereits mit dieser Häufigkeit bereitgestellt wurde, schließt die Laufzeitumgebung das Angebot aus.
- **Globale Angebotsaufnahmeregeln** - Sie können einen booleschen Ausdruck definieren, um Angebote auf einer Zielgruppenebene mithilfe der globalen Angebotstabelle zu filtern. Wenn das Ergebnis "false" ist, schließt die Laufzeitumgebung das Angebot aus.
- **Erweiterte Optionen** - Sie können die erweiterte Option **Diese Regel als berechtigt betrachten, wenn der folgende Ausdruck "true" ist** in einer Verfahrensregel verwenden, um Angebote auf einer Segmentebene zu filtern. Wenn das Ergebnis "false" ist, schließt die Laufzeitumgebung das Angebot aus.
- **Individuelle Angebotsaufnahmeregeln** - Sie können einen booleschen Ausdruck definieren, um Angebote auf Kundenebene zu filtern, indem Sie die Tabelle zur Überschreibung der Bewertung verwenden. Wenn das Ergebnis "false" ist, schließt die Laufzeitumgebung das Angebot aus.

Berechnen des Marketing-Score

Es gibt viele Möglichkeiten, den Marketing-Score zu beeinflussen (durch Verwendung einer Berechnung) oder sie zu überschreiben.

Dieses Diagramm zeigt die verschiedenen Phasen, in denen Sie den Marketing-Score beeinflussen oder überschreiben können.



Die Pfeile zeigen den Ausführungsvorrang an. Wenn Sie beispielsweise einen Ausdruck definieren, um den Marketing-Score in den erweiterten Optionen für eine Verfahrensregel zu definieren, und einen Ausdruck in der Bewertungsüberschreibungstabelle definieren, hat der Ausdruck in der Bewertungsüberschreibungstabelle Vorrang.

- **Globale Bewertung** - Sie können eine Bewertung pro Zielgruppenebene unter Verwendung der globalen Angebotstabelle definieren.
- **Marketierbewertung** - Sie können eine Bewertung pro Segment unter Verwendung des Schiebereglers in einer Verfahrensregel definieren.
- **Bewertung aus Bewertungsüberschreibung** - Sie können eine Bewertung pro Kunde unter Verwendung der Bewertungsüberschreibungstabelle definieren.
- **Globale Angebotsaufnahmeregeln** - Sie können einen Ausdruck definieren, der eine Bewertung pro Zielgruppenebene unter Verwendung der globalen Angebotstabelle berechnet.
- **Erweiterte Optionen** - Sie können einen Ausdruck definieren, der eine Bewertung pro Segment unter Verwendung der erweiterten Option **Folgenden Ausdruck als Marketing-Score verwenden** in einer Verfahrensregel berechnet.
- **Angebotsaufnahmeregeln für Bewertungsüberschreibung** - Sie können einen Ausdruck definieren, der eine Bewertung pro Kunde unter Verwendung der Bewertungsüberschreibungstabelle berechnet.

Den Lernprozess beeinflussen

Wenn Sie das integrierte Lernmodul von Unica Interact verwenden, können Sie die Lernausgabe über die Standardlernkonfigurationen wie die Liste von Lernattributen oder die Zuverlässigkeitsstufe hinaus beeinflussen. Sie können Komponenten des Lernalgorithmus außer Kraft setzen, während Sie die verbleibenden Komponenten verwenden.

Sie können die Lernfunktion mithilfe der Spalten `LikelihoodScore` und `AdjExploreScore` der Standardangebots- und Bewertungsüberschreibungstabellen überschreiben. Sie können diese Spalten den Standardangebots- und Bewertungsüberschreibungstabellen mithilfe des Funktionsscripts `aci_scoringfeature` hinzufügen. Um diese Überschreibungen ordnungsgemäß verwenden zu können, bedarf es eines umfassenden Verständnisses der integrierten Lernfunktion von Unica Interact.

Das Lernmodul nimmt eine Liste von möglichen Angeboten und die Marketing-Scores pro möglichem Angebot und verwendet sie in den endgültigen Berechnungen. Die Angebotsliste wird mit den Lernattributen verwendet, um die Wahrscheinlichkeit (Annahmewahrscheinlichkeit) zu berechnen, dass der Kunde das Angebot annehmen wird. Unter Verwendung dieser Wahrscheinlichkeiten sowie der Langzeitanzahl von Präsentationen, um zwischen Untersuchung und Nutzung auszugleichen, ermittelt der Lernalgorithmus die Angebotsgewichtung. Schließlich nimmt die integrierte Lernfunktion die Angebotsgewichtung, multipliziert sie mit dem endgültigen Marketing-Score und gibt eine endgültige Bewertung zurück. Die Angebote werden nach dieser endgültigen Bewertung sortiert.

Angebote unterdrücken

Sie können die Laufzeitumgebung so konfigurieren, dass Angebote unterdrückt werden.

Es stehen mehrere Möglichkeiten zur Verfügung, wie die Laufzeitumgebung ein Angebot unterdrückt:

- Geben Sie eine Menge für das Feld **Max. Anzahl Anzeigewiederholungen eines Angebots während eines Besuchs** ein.

  Sie definieren den Wert für **Max. Anzahl Anzeigewiederholungen eines Angebots während eines Besuchs** beim Erstellen oder Bearbeiten eines interaktiven Kanals.

- Mithilfe einer Tabelle für Angebotsunterdrückung.

  Sie erstellen eine Tabelle für Angebotsunterdrückung in Ihrer Profildatenbank.
- Mithilfe von Angeboten, deren Ablaufdatum überschritten wurde.
- Mithilfe von Angeboten aus abgelaufenen Kampagnen.
- Mithilfe von Angeboten, die ausgeschlossen wurden, weil sie einer Angebotsaufnahmeregel nicht entsprechen (erweiterte Option der Verfahrensregel).
- Angebote werden in einer Unica Interact-Sitzung explizit angenommen oder abgelehnt. Wenn ein Kunde ein Angebot explizit annimmt oder ablehnt, wird dieses Angebot für die Dauer der Sitzung unterdrückt.

Aktivieren der Angebotsunterdrückung

Mit diesem Verfahren können Sie die Angebotsunterdrückung aktivieren.

Sie können Unica Interact so konfigurieren, dass es auf eine Liste unterdrückter Angebote verweist.

1. Erstellen Sie eine Tabelle des Typs `offerSuppressionTable`. Dabei handelt es sich um eine neue Tabelle für jede Zielgruppe, in der die Zielgruppen-ID und die Angebots-ID enthalten sind.
2. Legen Sie die Eigenschaft `enableDefaultOfferLookup` auf **true** fest.
3. Legen Sie für die Eigenschaft `Interact > profile > offerSuppressionTable` den Namen der Tabelle für Angebotsunterdrückung für die entsprechende Zielgruppe fest.

Angebotsunterdrückungstabelle

Die Tabelle für Angebotsunterdrückung ermöglicht es Ihnen, ein Angebot für eine bestimmte Zielgruppen-ID zu unterdrücken. Beispiel: Wenn Ihre Zielgruppe Kunde ist, können Sie ein Angebot für Kunden John Smith unterdrücken. Eine Version dieser Tabelle für zumindest eine Zielgruppenebene muss in Ihrer Produktionsprofildatenbank vorhanden sein. Sie können eine Beispiel-Tabelle für Angebotsunterdrückung `UACI_BlackList` erstellen, indem Sie das SQL-Script `aci_usertab` in Ihrer Profildatenbank

ausführen. Das SQL-Script `aci_usrtab` befindet sich im `ddl`-Verzeichnis in Ihrem Laufzeitumgebungsinstallationsverzeichnis.

Sie müssen die Felder `AudienceID` und `OfferCode1` für jede Zeile definieren. Sie können zusätzliche Spalten hinzufügen, wenn Ihre Zielgruppen-ID oder Ihr Angebotscode aus mehreren Spalten besteht. Diese Spalten müssen mit den in Unica Campaign definierten Spaltennamen übereinstimmen. Beispiel: Wenn Sie die Zielgruppe `Customer` anhand der Felder `HHold_ID` und `MemberNum` definieren, müssen Sie `HHold_ID` und `MemberNum` der Tabelle für Angebotsunterdrückung hinzufügen.

Name	Syntax
AudienceID	(Erforderlich) Der Name dieser Spalte muss mit dem Namen der Spalte übereinstimmen, die die Zielgruppen-ID in Unica Campaign definiert. Wenn Ihre Zielgruppen-ID mehrere Spalten umfasst, können Sie sie dieser Tabelle hinzufügen. Jede Zeile muss die Zielgruppen-ID enthalten, die Sie dem Standardangebot zuweisen, z. B. `customer1`.
OfferCode1	(Erforderlich) Der Angebotscode für das Angebot, das Sie überschreiben. Wenn Ihre Angebotscodes aus mehreren Feldern bestehen, können Sie die zusätzlichen Spalten hinzufügen, z. B. `OfferCode2` etc.

Angebotsunterdrückung ignorieren

OfferSuppression für eine Sitzung kann mittels der folgenden Parameter ignoriert werden:

1. UACIIgnoreBlackList

TRUE - Wenn wir diesen Parameter als "true" übergeben, dann werden alle in der Blacklist-Tabelle verfügbaren Angebote dem Benutzer angezeigt/zurückgegeben.

FALSE - Wenn dies als "false" übergeben wird, werden alle Angebote, die in der Blacklist-Tabelle verfügbar sind, nicht angezeigt/an den Benutzer zurückgegeben.

2. UACIIgnoreSuppressionRules

TRUE - Wenn wir diesen Parameter als "true" übergeben, dann werden alle unterdrückten Angebote in Echtzeit angezeigt bzw. an den Benutzer zurückgeschickt.

FALSE - Wenn wir dies als "false" übergeben, werden alle in Echtzeit unterdrückten Angebote nicht angezeigt/zurückgegeben, wie es die Regel vorsieht.

Wenn diese Parameter nicht übergeben werden, gilt sie standardmäßig als false.

Die Parameter betreffen nur getOffers-Aufrufe in dieser Sitzung. Nachdem der Parameter gesetzt wurde, prüft Interact während der Verarbeitung von Kontakt- und Antwortereignissen keine Unterdrückungsregeln, sodass ein Kontakt-/Antwortereignis auf ein unterdrücktes Angebot gepostet werden kann.

Die auf den Angeboten definierten Unterdrückungsregeln werden durch die Echtzeitaktivitäten der Benutzer ausgelöst.

Die Blacklist-Tabelle steht für UACI_BlackList oder das Äquivalent, je nachdem, wie die Kunden sie nennen, und ILPB kann ihren Inhalt füllen.

Globale Angebote und individuelle Zuweisungen

Sie können die Laufzeitumgebung konfigurieren, um bestimmte Angebote außerhalb der Verfahrensregeln zuzuweisen, die auf der Registerkarte "Interaktionsstrategie" konfiguriert sind. Sie können globale Angebote für jedes Mitglied der Zielgruppenebene und individuelle Zuweisungen für bestimmte Zielgruppenmitglieder definieren. Beispiel: Sie können ein globales Angebot für alle Haushalte definieren, die angezeigt werden, wenn keine anderen verfügbar sind, und dann eine individuelle Angebotszuweisung für den bestimmten Smith-Haushalt erstellen.

Sie können sowohl die globalen Angebote als auch individuelle Zuweisungen durch Zone, Zelle und andere Angebotsaufnahmeregeln beschränken. Globale Angebote und individuelle Zuweisungen werden beide konfiguriert, indem Daten bestimmten Tabellen in Ihrer Produktionsprofildatenbank hinzugefügt werden.

Damit globale Angebote und individuelle Zuweisungen ordnungsgemäß funktionieren, müssen alle referenzierten Zell- und Angebotscodes in der Implementierung vorhanden sein. Um sicherzustellen, dass erforderliche Daten verfügbar sind, müssen Sie Standardzellencodes und die Tabelle `UACI_ICBatchOffers` konfigurieren.

Definieren der Standardzellencodes

Wenn Sie die Standardangebots- oder Bewertungsüberschreibungstabellen für globale oder individuelle Angebotszuweisungen verwenden, müssen Sie Standardzellencodes definieren. Der Standardzellencode (`DefaultCellCode`) wird verwendet, wenn kein Zellencode in einer bestimmten Zeile in den Standardangebots- oder Bewertungsüberschreibungstabellen definiert wurde. Die Berichterstellung verwendet diesen Standardzellencode.

Der Wert von `DefaultCellCode` muss dem in Unica Campaign definierten Zellencodeformat entsprechen. Dieser Zellencode wird bei allen Angebotszuweisungen verwendet, die in der Berichterstellung angezeigt werden.

Wenn Sie eindeutige Standardzellencodes definieren, können Sie ohne großen Aufwand Angebote identifizieren, die durch die Standardangebots- oder Bewertungsüberschreibungstabellen zugewiesen wurden.

Definieren Sie die Eigenschaft `DefaultCellCode` für jede Zielgruppenebene und jeden Tabellentyp in der Kategorie `IndividualTreatment`.

Definieren von Angeboten, die nicht in einer Verfahrensregel verwendet werden

Wenn Sie die Standardangebots- oder Bewertungsüberschreibungstabellen verwenden, müssen Sie sicherstellen, dass alle Angebotscodes in der Bereitstellung vorhanden sind. Wenn Sie wissen, dass alle Angebote, die Sie in den Standardangebots- oder Bewertungsüberschreibungstabellen verwenden, in Ihren Verfahrensregeln verwendet werden, sind die Angebote in der Implementierung vorhanden. Alle Angebote, die nicht in einer Verfahrensregel verwendet werden, müssen jedoch in der Tabelle `UACI_ICBatchOffers` definiert werden.

Die Tabelle `UACI_ICBatchOffers` existiert in den Unica Campaign-Systemtabellen.

Füllen Sie die Tabelle `UACI_ICBatchOffers` mit Angebotscodes aus, die in den Standardangebots- oder Bewertungsüberschreibungstabellen verwendet werden. Die Tabelle hat das folgende Format:

Spaltenname	Typ	Syntax
ICName	varchar(64)	Der Name des interaktiven Kanals, dem das Angebot zugeordnet ist. Wenn Sie dasselbe Angebot mit zwei verschiedenen interaktiven Kanälen verwenden, müssen Sie eine Zeile für jeden interaktiven Kanal bereitstellen.
OfferCode1	varchar(64)	Der erste Teil des Angebotscodes.
OfferCode2	varchar(64)	Der zweite Teil des Angebotscodes.
OfferCode3	varchar(64)	Der dritte Teil des Angebotscodes.
OfferCode4	varchar(64)	Der vierte Teil des Angebotscodes.
OfferCode5	varchar(64)	Der fünfte Teil des Angebotscodes.

Informationen zur globalen Angebotstabelle

Die globale Angebotstabelle ermöglicht es Ihnen, Verfahren auf der Zielgruppenebene zu definieren. Sie können beispielsweise ein globales Angebot für jedes Mitglied der Zielgruppe "Haushalt" definieren.

Sie können globale Einstellungen für die folgenden Elemente der Unica Interact-Angebotsbereitstellung definieren.

- Globale Angebotszuweisungen
- Globale Marketierbewertung, durch eine Zahl oder durch einen Ausdruck
- Boolescher Ausdruck, um Angebote zu filtern
- Lernwahrscheinlichkeit und Gewichtung, wenn Sie die integrierte Unica Interact-Lernfunktion verwenden
- Globale Lernfunktion-Überschreibung

Zuweisen von globalen Angeboten

Mit diesem Verfahren können Sie die Laufzeitumgebung konfigurieren, um globale Angebote für eine Zielgruppenebene außerhalb der Definitionen in den Verfahrensregeln zuzuweisen.

1. Erstellen Sie eine Tabelle namens `UACI_DefaultOffers` in Ihrer Profildatenbank.

   Verwenden Sie die DDL-Datei `aci_usrtab`, damit die Tabelle `UACI_DefaultOffers` mit den korrekten Spalten erstellt wird.
2. Legen Sie die Eigenschaft `Interact > profile > enableDefaultOfferLookup` auf **true** fest.

Globale Angebotstabelle

Die globale Angebotstabelle muss in Ihrer Profildatenbank existieren. Sie können die globale Angebotstabelle `UACI_DefaultOffers` erstellen, indem Sie das SQL-Script `aci_usertab` in Ihrer Profildatenbank ausführen.

Das SQL-Script `aci_usertab` befindet sich im `ddl`-Verzeichnis in Ihrem Laufzeitumgebungsinstallationsverzeichnis.

Sie müssen die Felder `AudienceLevel` und `OfferCode1` für jede Zeile definieren. Die anderen Felder sind optional, um Ihre Angebotszuweisungen weiter zu beschränken oder um die integrierte Lernfunktion auf der Zielgruppenebene zu beeinflussen.

Zur Erzielung einer optimalen Leistung sollten Sie einen Index auf der Zielgruppenebenenspalte dieser Tabelle erstellen.

Name	Typ	Syntax
AudienceLevel	varchar(64)	(Erforderlich) Der Name der Zielgruppenebene, der Sie das Standardangebot zuweisen, z. B. `Kunde` oder `Haushalt`. Dieser Name muss mit der in Unica Campaign definierten Zielgruppenebene übereinstimmen.

Name	Typ	Syntax
OfferCode1	varchar(64)	(Erforderlich) Der Angebotscode des Standardangebots. Wenn Ihre Angebotscodes aus mehreren Feldern bestehen, können Sie die zusätzlichen Spalten hinzufügen, z. B. `OfferCode2` etc. Wenn Sie dieses Angebot hinzufügen, um eine globale Angebotszuweisung bereitzustellen, müssen Sie dieses Angebot der Tabelle `UACI_ICBatchOffers` hinzufügen.
Bewertung	float	Eine Zahl, um den Marketing-Score für diese Angebotszuweisung zu definieren.
OverrideTypeID	int	Wenn auf `1` gesetzt und das Angebot nicht in der Liste von möglichen Angeboten aufscheint, dieses Angebot der Liste hinzufügen sowie Bewertungsdaten für das Angebot verwenden. Verwenden Sie im Allgemeinen `1`, um einzelne Angebotszuweisungen bereitzustellen. Wenn auf `0`, null oder auf eine andere Zahl als 1 festgelegt, Daten für das Angebot nur verwenden, wenn das Angebot in der Liste von möglichen Angeboten aufscheint. In den meisten Fällen wird eine Verfahrensregel oder eine individuelle Zuweisung diese Einstellung überschreiben.
Predicate	varchar(4000)	Sie können einen Ausdruck in diese Spalte als erweiterte Optionen für Verfahrensregeln eingeben. Sie können dieselben Variablen und Makros verwenden, die Ihnen beim Schreiben von erweiterten Optionen für Verfahrensregeln verfügbar sind. Das Verhalten dieser Spalte hängt vom Wert der Spalte `EnableStateID` ab.

Name	Typ	Syntax
		• Wenn `EnableStateID 2` ist, funktioniert diese Spalte wie die Option **Diese Regel als berechtigt betrachten, wenn der folgende Ausdruck "true" ist** in den erweiterten Optionen für Verfahrensregeln, um diese Angebotszuweisung zu beschränken. Diese Spalte muss einen booleschen Ausdruck enthalten und zu "true" auflösen, um dieses Angebot aufzunehmen. Wenn Sie versehentlich einen Ausdruck definieren, der in eine Zahl aufgelöst wird, wird jede Zahl ungleich null als "true" und null als "false" angesehen. • Wenn `EnableStateID 3` ist, funktioniert diese Spalte wie die Option **Folgenden Ausdruck als Marketing-Score verwenden** in den erweiterten Optionen für Verfahrensregeln, um dieses Angebot zu beschränken. Diese Spalte muss einen Ausdruck enthalten, der in eine Zahl aufgelöst wird. • Wenn `EnableStateID 1` ist, ignoriert Unica Interact jeden Wert in dieser Spalte. **Anmerkung:** Um einem Angebot eine Punktzahl zuzuweisen, wird die folgende Reihenfolge berücksichtigt. • Endergebnisfeld • Punktzahl [Spalte] • Prädikatspalte

Name	Typ	Syntax
		Wenn Sie der Prädikatspalte eine Punktzahl zuweisen möchten, müssen Sie die Punktespalte als Null belassen.
FinalScore	float	Eine Zahl zum Überschreiben der endgültigen Bewertung, die zum Sortieren der endgültigen Liste der zurückgegebenen Angebote verwendet wird. Diese Spalte wird verwendet, wenn Sie das integrierte Lernmodul aktiviert haben. Sie können Ihre eigene Lernfunktion implementieren, um diese Spalte zu verwenden.
CellCode	varchar(64)	Der Zellencode für ein bereitgestelltes interaktives Segment, dem Sie dieses Standardangebot zuweisen wollen. Wenn Ihre Zellencodes aus mehreren Feldern bestehen, können Sie die zusätzlichen Spalten hinzufügen. Sie müssen einen Zellencode bereitstellen, wenn `OverrideTypeID` 0 oder null ist. Wenn Sie keinen Zellencode aufnehmen, ignoriert die Laufzeitumgebung diese Datenzeile. Wenn `OverrideTypeID` 1 ist, müssen Sie in dieser Spalte keinen Zellencode bereitstellen. Wenn Sie keinen Zellencode bereitstellen, verwendet die Laufzeitumgebung den Zellencode, der in der Eigenschaft `DefaultCellCode` definiert ist, für diese Zielgruppenebene und Tabelle für Berichterstellungszwecke.

Name	Typ	Syntax
Zone	varchar(64)	Der Name der Zone, für die diese Angebotszuweisung gelten soll. Wenn NULL gilt dies für alle Zonen.
EnableStateID	int	Der Wert in dieser Spalte definiert das Verhalten der Spalte `Predicate`. • **1** - Die Spalte `Predicate` nicht verwenden. • **2** - `Predicate` als einen booleschen Ausdruck verwenden, um das Angebot zu filtern. Dies befolgt dieselben Regeln wie die erweiterte Option **Diese Regel als berechtigt betrachten, wenn der folgende Ausdruck wahr ist** in einer Verfahrensregel. • **3** - `Predicate` verwenden, um die Marketierbewertung zu definieren. Dies befolgt dieselben Regeln wie die erweiterte Option **Der folgenden Ausdruck als Marketing-Score verwenden** in einer Verfahrensregel. Zeilen, bei denen diese Spalte Null oder ein anderer Wert als 2 oder 3 ist, ignorieren die Spalte `Predicate`.
LikelihoodScore	float	Diese Spalte wird nur verwendet, um die integrierte Lernfunktion zu beeinflussen. Sie können diese Spalte mit der DDL `aci_scoringfeature` hinzufügen.
AdjExploreScore	float	Diese Spalte wird nur verwendet, um die integrierte Lernfunktion zu beeinflussen. Sie können diese Spalte mit der DDL `aci_scoringfeature` hinzufügen.

Name	Typ	Syntax
Unterdrückungszahl	int	Dieses Feld ist für die Unterdrückung von Exklusivangeboten vorgesehen. Das Feld befindet sich auf der Seite "Strategie". Sobald Sie die Suppression Count in der Erweiterten Strategieoption für die Regel speichern, wird der Wert 'Suppression Count' in dieser Spalte aktualisiert. Standardmäßig ist der Wert 0.n.
Max Score	int	Standardmäßig ist der Wert "false"(0), und in der Strategie wird der Wert "true"(1), sobald Sie Max score für die Regel wählen und die Strategie speichern.

Informationen zur Bewertungsüberschreibungstabelle

Die Bewertungsüberschreibungstabelle ermöglicht es Ihnen, Verfahren auf einer Zielgruppen-ID- oder einer individuellen Ebene zu definieren. Beispiel: Wenn Ihre Zielgruppenebene Besucher ist, können Sie Überschreibung für bestimmte Besucher erstellen.

Sie können Überschreibungen für die folgenden Elemente der Unica Interact-Angebotsbereitstellung definieren.

- Individuelle Angebotszuweisung
- Individuelle Marketierbewertung, durch eine Zahl oder durch einen Ausdruck
- Boolescher Ausdruck, um Angebote zu filtern
- Lernwahrscheinlichkeit und Gewichtung, wenn Sie die integrierte Lernfunktion verwenden
- Individuelle Lernfunktion-Überschreibung

Konfigurieren von Bewertungsüberschreibungen

Sie können in der Konfiguration von Unica Interact festlegen, dass anstelle des Marketing-Score eine Bewertung verwendet wird, die auf Basis einer Modellierungsanwendung generiert wird.

1. Erstellen Sie eine Bewertungsüberschreibungstabelle für jede Zielgruppenebene, der Sie Überschreibungen bereitstellen möchten.

   Mithilfe der DDL-Datei `aci_usrtab` können Sie eine Beispiel-Bewertungsüberschreibungstabelle mit den korrekten Spalten erstellen.
2. Legen Sie die Eigenschaft `Interact > profile > enableScoreOverrideLookup` auf **true** fest.
3. Legen Sie die Eigenschaft `scoreOverrideTable` auf den Namen der Bewertungsüberschreibungstabelle für jede Zielgruppenebene fest, der Sie Überschreibungen bereitstellen möchten.

   Sie müssen nicht eine Bewertungsüberschreibungstabelle für jede Zielgruppenebene bereitstellen.

Bewertungsüberschreibungstabelle

Die Bewertungsüberschreibungstabelle muss in Ihrer Produktionsprofildatenbank existieren. Sie können eine Beispiel-Bewertungsüberschreibungstabelle `UACI_ScoreOverride` erstellen, indem Sie das SQL-Script `aci_usertab` in Ihrer Profildatenbank ausführen.

Das SQL-Script `aci_usrtab` befindet sich im `ddl`-Verzeichnis in Ihrem Laufzeitumgebungsinstallationsverzeichnis.

Sie müssen die Felder *`AudienceID`*, `OfferCode1` und `Score` für jede Zeile definieren. Die Werte in den anderen Feldern sind optional, um Ihre einzelnen Angebotszuweisungen weiter zu beschränken oder um Bewertungsüberschreibungsinformationen für die integrierte Lernfunktion bereitzustellen.

Name	Typ	Syntax
AudienceID	varchar(64)	(Erforderlich) Der Name dieser Spalte muss mit dem Namen der Spalte übereinstimmen, die die Zielgruppen-ID in Unica Campaign definiert. Die Beispieltabelle, die durch die DDL-Datei `aci_usrtab` erstellt wird, erstellt diese Spalte als die Spalte `CustomerID`. Wenn Ihre Zielgruppen-ID mehrere Spalten umfasst, können Sie sie dieser Tabelle hinzufügen. Jede Zeile muss die Zielgruppen-ID enthalten, die Sie dem einzelnen Angebot zuweisen, z. B.`customer1`. Zur Erzielung einer optimalen Leistung sollten Sie einen Index auf dieser Spalte erstellen.
OfferCode1	varchar(64)	(Erforderlich) Der Angebotscode für das Angebot. Wenn Ihre Angebotscodes aus mehreren Feldern bestehen, können Sie die zusätzlichen Spalten hinzufügen, z. B. `OfferCode2` etc. Wenn Sie dieses Angebot hinzufügen, um eine einzelne Angebotszuweisung bereitzustellen, müssen Sie dieses Angebot der Tabelle `UACI_ICBatchOffers` hinzufügen.
Bewertung	float	Eine Zahl, um den Marketing-Score für diese Angebotszuweisung zu definieren.
OverrideTypeID	int	Wenn auf `0` oder null (oder auf eine andere Zahl als 1) festgelegt, Daten für das Angebot nur verwenden, wenn das Angebot in der Liste von möglichen Angeboten aufscheint. Verwenden Sie im Allgemeinen `0`, um Bewertungsüberschreibungen bereitzustellen. Sie müssen einen Zellknoten bereitstellen.

Name	Typ	Syntax
		Wenn auf `1` gesetzt und das Angebot nicht in der Liste von möglichen Angeboten aufscheint, dieses Angebot der Liste hinzufügen sowie Bewertungsdaten für das Angebot verwenden. Verwenden Sie im Allgemeinen `1`, um einzelne Angebotszuweisungen bereitzustellen.
Predicate	varchar(4000)	Sie können einen Ausdruck in diese Spalte als erweiterte Optionen für Verfahrensregeln eingeben. Sie können dieselben Variablen und Makros verwenden, die Ihnen beim Schreiben von erweiterten Optionen für Verfahrensregeln verfügbar sind. Das Verhalten dieser Spalte hängt vom Wert der Spalte `EnableStateID` ab. • Wenn `EnableStateID` `2` ist, funktioniert diese Spalte wie die Option **Diese Regel als berechtigt betrachten, wenn der folgende Ausdruck "true" ist** in den erweiterten Optionen für Verfahrensregeln, um diese Angebotszuweisung zu beschränken. Diese Spalte muss einen booleschen Ausdruck enthalten und zu "true" auflösen, um dieses Angebot aufzunehmen. Wenn Sie versehentlich einen Ausdruck definieren, der in eine Zahl aufgelöst wird, wird jede Zahl ungleich null als "true" und null als "false" angesehen. • Wenn `EnableStateID` `3` ist, funktioniert diese Spalte wie die Option **Folgenden Ausdruck als Marketing-Score verwenden** in den erweiterten Optionen für Verfahrensre-

Name	Typ	Syntax
		geln, um dieses Angebot zu beschränken. Diese Spalte muss einen Ausdruck enthalten, der in eine Zahl aufgelöst wird. • Wenn `EnableStateID` `1` ist, ignoriert Unica Interact jeden Wert in dieser Spalte. **Anmerkung:** Um einem Angebot eine Punktzahl zuzuweisen, wird die folgende Reihenfolge berücksichtigt. • Endergebnisfeld • Punktzahl [Spalte] • Prädikatspalte Wenn Sie der Prädikatspalte eine Punktzahl zuweisen möchten, müssen Sie die Punktespalte als Null belassen.
FinalScore	float	Eine Zahl zum Überschreiben der endgültigen Bewertung, die zum Sortieren der endgültigen Liste der zurückgegebenen Angebote verwendet wird. Diese Spalte wird verwendet, wenn Sie das integrierte Lernmodul aktiviert haben. Sie können Ihre eigene Lernfunktion implementieren, um diese Spalte zu verwenden.
CellCode	varchar(64)	Der Zellencode für ein interaktives Segment, dem Sie dieses Angebot zuweisen wollen. Wenn Ihre Zellencodes aus mehreren Feldern bestehen, können Sie die zusätzlichen Spalten hinzufügen.

Name	Typ	Syntax
		Sie müssen einen Zellencode bereitstellen, wenn `OverrideTypeID` 0 oder null ist. Wenn Sie keinen Zellencode aufnehmen, ignoriert die Laufzeitumgebung diese Datenzeile. Wenn `OverrideTypeID` 1 ist, müssen Sie in dieser Spalte keinen Zellencode bereitstellen. Wenn Sie keinen Zellencode bereitstellen, verwendet die Laufzeitumgebung den Zellencode, der in der Eigenschaft `DefaultCellCode` definiert ist, für diese Zielgruppenebene und Tabelle für Berichterstellungszwecke.
Zone	varchar(64)	Der Name der Zone, für die diese Angebotszuweisung gelten soll. Wenn NULL gilt dies für alle Zonen.
EnableStateID	int	Der Wert in dieser Spalte definiert das Verhalten der Spalte `Predicate`. • **1** - Die Spalte `Predicate` nicht verwenden. • **2** - `Predicate` als einen booleschen Ausdruck verwenden, um das Angebot zu filtern. Dies befolgt dieselben Regeln wie die erweiterte Option **Diese Regel als berechtigt betrachten, wenn der folgende Ausdruck wahr ist** in einer Verfahrensregel. • **3** - `Predicate` verwenden, um die Marketierbewertung zu definieren. Dies befolgt dieselben Regeln wie die erweiterte Option **Der folgenden Ausdruck als Marketing-Score verwenden** in einer Verfahrensregel.

Name	Typ	Syntax
		Zeilen, bei denen diese Spalte Null oder ein anderer Wert als 2 oder 3 ist, ignorieren die Spalte `Predicate`.
LikelihoodScore	float	Diese Spalte wird nur verwendet, um die integrierte Lernfunktion zu beeinflussen. Sie können diese Spalte mit der DDL `aci_scoringfeature` hinzufügen.
AdjExploreScore	float	Diese Spalte wird nur verwendet, um die integrierte Lernfunktion zu beeinflussen. Sie können diese Spalte mit der DDL `aci_scoringfeature` hinzufügen.
Unterdrückungszahl	int	Dieses Feld ist für die Unterdrückung von Exklusivangeboten vorgesehen. Das Feld befindet sich auf der Seite "Strategie". Sobald Sie die Suppression Count in der Erweiterten Strategieoption für die Regel speichern, wird der Wert 'Suppression Count' in dieser Spalte aktualisiert. Standardmäßig ist der Wert 0.n.
Max Score	int	Standardmäßig ist der Wert "false"(0), und in der Strategie wird der Wert "true"(1), sobald Sie Max score für die Regel wählen und die Strategie speichern.

Übersicht über das integrierte Lernen von Unica Interact

Während Sie alle Anstrengungen unternehmen, um sicherzustellen, dass den richtigen Segmenten die richtigen Angebote vorgeschlagen werden, können Sie jederzeit für Sie Wichtiges aus der tatsächlichen Auswahl Ihrer Besucher lernen. Das tatsächliche Verhalten

der Besucher sollte Ihre Strategie beeinflussen. Sie können den Antwortverlauf verwenden und ihn einige Modellierungstools durchlaufen lassen, um eine Bewertung zu erhalten. Diesen können Sie in die interaktiven Ablaufdiagramme einbeziehen.

Dabei handelt es sich jedoch um keine Echtzeitdaten.

Unica Interact bietet Ihnen zwei Optionen, damit Sie aus den Aktionen der Besucher in Echtzeit lernen können.

- Integriertes Lernmodul - Die Laufzeitumgebung verfügt über ein naives bayessches Lernmodul. Dieses Modul überwacht die Kundenattribute Ihrer Wahl und verwendet diese Daten als Hilfe beim Auswählen der Angebote, die angezeigt werden sollen.
- Lern-API - Die Laufzeitumgebung verfügt auch über eine Lern-API, über die Sie Ihr eigenes Lernmodul schreiben können.

Sie müssen die Lernfeatures nicht verwenden. Diese Features sind standardmäßig inaktiviert.

Unica Interact-Lernmodul

Das Unica Interact-Lernmodul überwacht die Antworten des Besuchers auf Angebote und Besucherattribute.

Modi des Lernmoduls

Das Lernmodul weist zwei allgemeine Modi auf:

- Untersuchung - das Lernmodul verarbeitet Angebote, um genügend Antwortdaten zusammenzustellen, damit die im Nutzungsmodus verwendete Einschätzung optimiert werden kann. Die in der Untersuchungsphase angezeigten Angebote spiegeln nicht unbedingt die optimale Auswahl wider.
- Nutzung - Nachdem genügend Daten in der Untersuchungsphase gesammelt wurden, wählt das Lernmodul die anzuzeigenden Angebote auf Basis der Wahrscheinlichkeiten aus.

Das Lernmodul verwendet für den alternativen Einsatz des Untersuchungs- und Nutzungsmodus jeweils zwei Eigenschaften. Dabei handelt es sich um die folgenden beiden Eigenschaften:

- eine Zuverlässigkeitsstufe, die Sie mit der Eigenschaft `confidenceLevel` konfigurieren.
- eine Wahrscheinlichkeit, dass das Lernmodul ein Zufallsangebot anzeigt. Diesen Wert konfigurieren Sie mit der Eigenschaft `percentRandomSelection`.

Eigenschaft für die Zuverlässigkeitsstufe

Für die Eigenschaft `confidenceLevel` legen Sie einen Prozentsatz fest, der angibt, wie sicher sich das Lernmodul sein muss, bevor seine Bewertungen für ein Angebot im Auswahlverfahren verwendet werden. Wenn dem Lernmodul anfangs keine Daten zur Verarbeitung vorliegen, ist es ausschließlich auf den Marketing-Score angewiesen. Nachdem jedes Angebot mit einer durch `minPresentCountThreshold` definierten Häufigkeit angezeigt wurde, wechselt das Lernmodul in den Untersuchungsmodus. Das Lernmodul benötigt eine große Datenmenge zum Arbeiten, um überzeugt davon zu sein, dass die von ihm berechneten Prozentsätze korrekt sind. Deshalb bleibt es im Untersuchungsmodus.

Das Lernmodul ordnet jedem Angebot Gewichte zu. Zum Berechnen der Gewichtungen verwendet das Lernmodul eine Formel, die auf der konfigurierten Zuverlässigkeitsstufe sowie auf historischen Annahmedaten und den aktuellen Sitzungsdaten basiert. Die Formel gleicht von vornherein zwischen Untersuchung und Nutzung aus und gibt das entsprechende Gewicht zurück.

Eigenschaft für die Zufallsauswahl

Um sicherzustellen, dass das System nicht Angebote bevorzugt, die in den frühen Stadien am besten abschneiden, zeigt Unica Interact ein Zufallsangebot über die mit `percentRandomSelection` festgelegte prozentuale Zeit an. Durch diesen Prozentsatz für Zufallsangebote wird das Lernmodul gezwungen, andere Angebote als die erfolgreichsten zu empfehlen, und kann so ermitteln, ob andere Angebote erfolgreicher wären, wenn sie prominenter präsentiert würden. Wenn Sie `percentRandomSelection` z.B. mit dem Wert5

konfigurieren, bedeutet dies, dass das Lernmodul während 5% der Zeit ein Zufallsangebot anzeigt und die Antwortdaten seinen Berechnungen hinzufügt.

Mit dem Wert für **% Zufallswert** können Sie im Fenster "Interaktiver Kanal" in der Registerkarte "Interaktionspunkte" für jede Zone die Wahrscheinlichkeit angeben, mit der das zurückgegebene Angebot zufällig ausgewählt wird.

Vorgehensweise des Lernmoduls beim Bestimmen von Angeboten

Das Lernmodul bestimmt auf folgende Art, welche Angebote präsentiert werden.

1. Es berechnet die Wahrscheinlichkeit, mit der ein Besucher ein Angebot auswählt.
2. Es berechnet die Angebotsgewichtung mithilfe der Wahrscheinlichkeit aus Schritt 1 und ermittelt, ob es sich im Untersuchungs- oder im Nutzungsmodus befinden sollte.
3. Es berechnet die endgültige Bewertung für jedes Angebot unter Verwendung des Marketing-Score und der Angebotsgewichtung aus Schritt 2.
4. Es sortiert die Angebote nach den in Schritt 3 ermittelten Bewertungen und gibt die angeforderte Anzahl von Spitzenangeboten zurück.

Beispielsweise ermittelt das Lernmodul, dass ein Besucher Angebot A wahrscheinlich zu 30 % und Angebot B wahrscheinlich zu 70 % annimmt und dass es diese Informationen nutzen sollte. Anhand der Verfahrensregeln liegt der Marketing-Score für Angebot A bei 75 und für Angebot B bei 55. Da die Bewertung aufgrund der Berechnungen in Schritt 3 jedoch für Angebot B höher als für Angebot A liegt, empfiehlt die Laufzeitumgebung Angebot B.

Anmerkung: Mehrere Antwortereignisse auf ein einzelnes Kontaktereignis führen zu einer Verzerrung der Lernpunktzahl.

Eigenschaften für den Gewichtungsfaktor

Der Lernprozess basiert auch auf der Eigenschaft `recencyWeightingFactor` und der Eigenschaft `recencyWeightingPeriod`. Diese Eigenschaften ermöglichen es Ihnen, aktuelleren Daten gegenüber älteren Daten mehr Gewicht zu verleihen. Der Wert von `recencyWeightingFactor` ist der Prozentsatz der Gewichtung, der den aktuellen Daten zugewiesen werden soll. Der Wert von `recencyWeightingPeriod` ist die aktuelle

Dauer. So konfigurieren Sie beispielsweise `recencyWeightingFactor` mit 0,30 und `recencyWeightingPeriod` mit 24. Diese Einstellungen bedeuten, dass es sich bei den Daten der vergangenen 24 Stunden um 30 % aller berücksichtigten Daten handelt. Wenn Ihnen die Daten für eine Woche vorliegen, machen alle über die ersten sechs Tage gemittelten Daten 70 % der Daten und der letzte Tag 30 % aus.

In die Staging-Tabelle geschriebene Daten

Bei jeder Sitzung werden die folgenden Daten in eine Lern-Staging-Tabelle geschrieben:

- Angebotskontakt
- Angebotsannahme
- Lernattribute

Ein Aggregator liest die Daten in einem konfigurierbaren Intervall aus der Staging-Tabelle, kompiliert sie und schreibt sie in eine Tabelle. Das Lernmodul liest diese Aggregatdaten und verwendet sie in Berechnungen.

Aktivieren des Lernmoduls

Alle Laufzeitserver haben ein integriertes Lernmodul. Dieses Modul ist standardmäßig inaktiviert. Sie können das Lernmodul aktivieren, indem Sie eine Konfigurationseigenschaft ändern.

Bearbeiten Sie in Unica Platform für die Laufzeitumgebung folgende Konfigurationseigenschaften in der Kategorie `Interact > offerserving`.

Konfigurationseigenschaft	Einstellung
`optimizationType`	**BuiltInLearning**

Lernattribute

Das Lernmodul lernt anhand von Besucherattributen, den Zuständen von Ereignismustern und Angebotsannahmedaten. Sie können auswählen, welche Besucherattribute überwacht

werden sollen. Diese Besucherattribute können ein beliebiges Element in einem Kundenprofil sein, einschließlich eines Ereignisparameters, den Sie in Echtzeit erfassen.

Attribute aus dimensionalen Tabellen werden beim Lernen nicht unterstützt.

Auch wenn Sie eine beliebige Anzahl von Attributen zum Überwachen konfigurieren können, so empfiehlt HCL dennoch, dass Sie nicht mehr als zehn Lernattribute zwischen den statischen und dynamischen Lernattributen konfigurieren. Außerdem sollten Sie die folgenden Leitlinien berücksichtigen.

- Wählen Sie unabhängige Attribute aus.

  Wählen Sie keine ähnlichen Attribute aus. Wenn Sie beispielsweise ein Attribut namens HighValue erstellen und dieses Attribut durch eine Berechnung auf der Grundlage des Gehalts definiert wird, wählen Sie weder HighValue noch Gehalt aus. Ähnliche Attribute sind für den Lernalgorithmus hinderlich.
- Wählen Sie Attribute mit eigenständigen Werten aus.

  Wenn ein Attribut Wertspannen aufweist, müssen Sie einen exakten Wert auswählen. Wenn Sie z. B. Gehalt als Attribut verwenden möchten, sollten Sie jedem Gehaltsbereich einen bestimmten Wert zuweisen, der Bereich 20.000-30.000 sollte A sein, 30.001-40.000 sollte B sein usw. Sie können auch Bins im interaktiven System definieren, und das Lernsystem führt die Zuordnung automatisch durch.
- Begrenzen Sie die Anzahl der Attribute, um die Leistung nicht zu behindern.

  Die Anzahl der Attribute, die Sie verfolgen können, hängt von Ihren Leistungsanforderungen und Ihrer Unica Interact-Installation ab. Sofern möglich verwenden Sie ein anderes Modellierungstool (wie z. B. PredictiveInsight), um die zehn wichtigsten prognostizierbaren Attribute zu bestimmen. Sie können das Lernmodul konfigurieren, um automatisch Attribute zu entfernen, die nicht prognostizierbar sind, aber auch Leistungsbeeinträchtigung bewirken.

Sie können die Leistung verwalten, indem Sie sowohl die Anzahl der zu überwachenden Attribute als auch die Anzahl der Werte pro zu überwachendem Attribut definieren. Mit der Eigenschaft `Campaign > partitions > partition1 > Interact > learning > maxAttributeNames` wird die maximale Anzahl der zu verfolgenden Besucherattribute

definiert. Mit der Eigenschaft `maxAttributeNames` wird die maximale Anzahl der zu verfolgenden Besucherattribute definiert. Alle anderen Werte werden einer Kategorie zugeordnet, die durch den Eigenschaftswert `otherAttributeValue` definiert wird. Das Lernmodul verfolgt jedoch nur die ersten gefundenen Werte. So möchten Sie beispielsweise das Benutzerattribut Augenfarbe verfolgen. Da Sie nur an den Werten Blau, Braun und Grün interessiert sind, legen Sie den Wert für `maxAttributeValues` auf 3 fest. Die ersten drei Besucher weisen jedoch die Werte Blau, Braun und Haselnussbraun auf. Dies bedeutet, dass allen Besuchern mit grünen Augen das Attribut `otherAttributeValue` zugeordnet wird.

Sie können auch dynamische Lernattribute verwenden, mit denen Sie Ihre Lernkriterien spezifischer definieren können. Dynamische Lernattribute ermöglichen es Ihnen, anhand der Kombination von zwei Attributen wie von einem einzigen Eintrag zu lernen. Betrachten Sie als Beispiel die folgenden Profilinformationen:

Besucher-ID	Kartentyp	Kartensaldo
1	Gold Card	$1.000
2	Gold Card	$9.000
3	Bronze Card	$1.000
4	Bronze Card	$9.000

Wenn Sie Standardlernattribute verwenden, können Sie nur jeweils anhand des Kartentyps und des Saldos lernen. Besucher 1 und 2 werden, basierend auf demselben Kartentyp, gemeinsam gruppiert, und Besucher 2 und 4 werden basierend auf dem Kartensaldo gruppiert. Dies ist möglicherweise kein korrekter Prädiktor von Angebotsannahmeverhalten. Wenn Goldkarteninhaber dazu tendieren, einen höheren Saldo zu haben, könnte sich das Verhalten von Besucher 2 radikal von Besucher 4 unterscheiden, was die Standardlernattribute verfälschen würde. Wenn Sie allerdings dynamische Lernattribute verwenden, wird individuell anhand der jeweiligen Besucher gelernt, was die Vorhersagen genauer macht.

Wenn Sie dynamische Lernattribute verwenden und der Besucher zwei gültige Werte für ein Attribut hat, wählt das Lernmodul den zuerst gefundenen Wert aus.

Wenn Sie die Eigenschaft `enablePruning` auf `yes` festlegen, ermittelt das Lernmodul algorithmisch, welche Attribute nicht prädiktiv sind, und hört auf, diese Attribute bei der Berechnung von Gewichtungen zu berücksichtigen. Wenn Sie beispielsweise ein Attribut verfolgen, das die Haarfarbe darstellt, und das Lernmodul bestimmt, dass es kein Muster für das Annehmen eines Angebots basierend auf der Haarfarbe des Besuchers gibt, hört das Lernmodul auf, das Attribut Haarfarbe zu berücksichtigen. Attribute werden jedes Mal neu bewertet, wenn der Lernaggregationsprozess ausgeführt wird (durch die Eigenschaft `aggregateStatsIntervalInMinutes` definiert). Dynamische Lernattribute werden auch entfernt.

Die **Ereignismusterzustände** können nun in Learning verwendet werden. Der Name von Ereignismustern mit dem in der Konfigurationseinstellung Affinium|Campaign|Partitionen|Partition1|Interact|Ablaufdiagramm:eventPatternPrefix angegebenen Präfixwert kann in ein Lernmodell und globale Lernattribute eingefügt werden.

Sie werden wie Profilattribute behandelt.

Die Werte eines Ereignismusters können einer der folgenden sein:

0 - Bedingung nicht erfüllt

1 - Bedingung erfüllt

-1- Abgelaufen

-2- Deaktiviert

-3 - noch nicht aktiviert

Definieren eines Lernattributs

Mit diesem Verfahren können Sie ein Lernattribut definieren.

Sie können die Anzahl der Besucherattribute bis zum maximalen Wert `maxAttributeNames` konfigurieren.

`(learningAttributes)` ist eine Vorlage, um neue Lernattribute zu erstellen. Sie müssen für jedes Attribut einen neuen Namen eingeben. Es ist nicht möglich, zwei gleichnamige Kategorien zu erstellen.

Bearbeiten Sie in Unica Platform für die Designumgebung folgende Konfigurationseigenschaften in der Kategorie `Campaign > Partitionen > Partition`*n* `> Interact > Lernen`.

Konfigurationseigenschaft	Einstellung
`attributeName`	Der Wert für `attributeName` muss mit dem Namen eines Name/Wert-Paars in den Profildaten übereinstimmen. Bei diesem Namen braucht die Groß-/Kleinschreibung nicht berücksichtigt zu werden.

Definieren von dynamischen Lernattributen

Um dynamische Lernattribute zu definieren, müssen Sie die `UACI_AttributeList`-Tabelle in der Lerndatenquelle ausfüllen.

Alle Spalten in dieser Tabelle haben den Typ varchar(64).

Spalte	Syntax
`AttributeName`	Der Name des dynamischen Attributs, über das Sie etwas lernen möchten. Dieser Wert muss ein tatsächlich möglicher Wert in `AttributeNameCol` sein.
`AttributeNameCol`	Der vollständig qualifizierte Name der Spalte (hierarchische Struktur, mit der Profiltabelle beginnend), in der `AttributeName` gefunden werden kann. Dieser Spaltenname muss nicht notwendigerweise ein Standardlernattribut sein.
`AttributeValueCol`	Der vollständig qualifizierte Name der Spalte (hierarchische Struktur, mit der Profiltabelle beginnend), in der `AttributeName` gefunden werden kann.

Betrachten Sie zum Beispiel die folgende Profiltabelle und die zugehörige Dimensionstabelle.

Tabelle 7. MyProfileTable

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

Tabelle 8. MyDimensionTable

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

Das folgende Beispiel zeigt eine `UACI_AttributeList`-Tabelle zum Abgleich von Kartentyp und Saldo.

Tabelle 9. UACI_AttributeList

AttributeName	AttributeNameCol	AttributeValueCol
Gold Card	MyProfileTable.MyDimensionTable. CardType	MyProfileTable.MyDimensionTable. CardBalance
Bronze Card	MyProfileTable.MyDimensionTable. CardType	MyProfileTable.MyDimensionTable. CardBalance

Unica Interact AutoBinning

In Interact funktioniert der integrierte Lernalgorithmus teilweise durch Speichern und Analysieren der Werte von Profilattributen zum Zeitpunkt der Kontaktaufnahme und

Beantwortung von Angeboten. Einige Attribute können eine praktisch unbegrenzte Anzahl von eindeutigen Werten haben. Aufgrund der begrenzten Ressourcen in einem Interact-System können Sie jedoch nur eine kleine Anzahl von ihnen speichern. Darüber hinaus ist es oft sinnvoller, die Analyse auf der Grundlage der Wertebereiche durchzuführen. Sie können diese Funktion verwenden, um solche Bins in Interact zu erstellen, und das lernende Subsystem wird die Zuordnung automatisch vornehmen.

Sie können die Bin-Definitionen von der Seite **Interact -> Globales Lernen -> Alle Bin-Definitionen** erstellen. Während des Hinzufügens oder der Bearbeitung einer Bin-Definition können Sie Profilattribute aus der Liste ALLER Attribute aus allen abgebildeten Profiltabellen auswählen. Die Typen einer Bin-Definition können entweder als Bereich oder als Liste definiert werden. Der Typ "Bereich" kann nur mathematische Operatoren haben, der Typ "Liste" kann nur den Operator "enthält" haben und besteht aus einer Liste von Werten.

Beispiel für einen Bin vom Typ " Bereich":

low income < =30000

30000 < medium income < =60000

high income > 60000

Beispiel für einen Bin vom Typ "Liste":

New England: MA, NH, CT

North West: MI, IL

Eine Bin-Definition umfasst globale Daten über alle interaktiven Kanäle und über alle Lernmodelle hinweg.

Alle Bin-Definitionen werden als Teil von Global Deployment Data bereitgestellt. Sie können in jedem interaktiven Kanal bereitgestellt werden, und zwar einmal und für ALLE. Danach werden die neuen Bin-Definitionen in einem Speichercache gespeichert, der nur für das eingebaute Lernsubsystem sichtbar ist.

Wenn ein Kontakt- oder Antwortereignis gepostet wird, wird der Wert eines Profilattributs einem Bin zugeordnet, falls ein solcher existiert. Die "bin"-Werte werden beim Einloggen in die Lerntabellen verwendet. Wenn Bins für das Attribut definiert sind und der Attributwert

nicht Teil einer Bin-Definition ist, dann wird der Attributwert in Lerntabellen als SONSTIGES protokolliert.

Konfigurieren der Laufzeitumgebung für die Erkennung von externen Lernmodulen

Sie können die Lern-Java™-API verwenden, um Ihr eigenes Lernmodul zu schreiben. Sie müssen die Laufzeitumgebung konfigurieren, um Ihr Lerndienstprogramm in Unica Platform zu erkennen.

Sie müssen den Unica Interact-Laufzeitserver erneut starten, damit diese Änderungen wirksam werden.

1. Bearbeiten Sie in Unica Platform für die Laufzeitumgebung folgende Konfigurationseigenschaften in der Kategorie `Interact > offerserving`. Die Konfigurationseigenschaften für die Lernoptimierungsprogramm-API befinden sich in der Kategorie `Interact > offerserving > External Learning Config`.

Konfigurationseigenschaft	Einstellung
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Klassenname für das externe Lernen
`externalLearningClassPath`	Der Pfad zu den Klassen- oder JAR-Dateien auf dem Laufzeitserver für das externe Lernen. Wenn Sie eine Servergruppe verwenden und alle Laufzeitserver dieselbe Instanz von Unica Platform referenzieren, muss jeder Server über eine Kopie der Klassen- oder JAR-Dateien an derselben Position verfügen.

2. Starten Sie den Unica Interact-Laufzeitserver erneut, damit diese Änderungen wirksam werden.

Kapitel 6. Informationen zur Unica Interact-API

Unica Interact stellt Angebote für eine große Vielfalt von Touchpoints bereit. Sie können beispielsweise die Laufzeitumgebung und Ihren Touchpoint konfigurieren, Nachrichten an Ihre Call-Center-Mitarbeiter zu senden, die sie über die besten Up-Selling- oder Cross-Selling-Möglichkeiten bei einem Kunden informieren, der mit einer bestimmten Art von Serviceanfrage anruft. Sie können auch die Laufzeitumgebung und Ihren Touchpoint konfigurieren, maßgeschneiderte Angebote einem Kunden (Besucher) bereitzustellen, der zu einem bestimmten Bereich Ihrer Website navigiert ist.

Die Unica Interact-Anwendungsprogrammierschnittstelle (API) ermöglicht es Ihnen, Ihren Touchpoint und einen Laufzeitserver so zu konfigurieren, dass sie zusammenarbeiten, um die bestmöglichen Angebote bereitzustellen. Mithilfe der API kann der Touchpoint Informationen vom Laufzeitserver anfordern, um den Besucher einer Gruppe (Segment) zuzuweisen und um auf diesem Segment basierende Angebote bereitzustellen. Sie können auch Daten für eine spätere Analyse protokollieren, um Ihre Angebotspräsentationsstrategien zu optimieren.

Die Unica Interact-API ermöglicht außerdem über JavaScript die Kommunikation zwischen dem Endbenutzerclient und dem Server.

Um Ihnen die größtmögliche Flexibilität bei der Integration von Unica Interact in Ihre Umgebungen zu bieten, stellt HCL einen Webservice bereit, der über die Unica Interact-API zugänglich ist.

Standardmäßig werden alle Parameter in der aktuellen Sitzung gespeichert und wirken sich somit auf alle nachfolgenden APIs aus. Die Parameter `UACIPreRemoveParameter` und `UACIPostRemoveParameter` können verwendet werden, um unerwünschte Parameter aus der Sitzung zu löschen.

Allen API-Parametern wird ein Flaggentransient hinzugefügt. Wenn der Wert eines Parameters `INVOCATION (1)` ist, dann ist dieser Parameter nur während des Prozesses dieses API-Aufrufs wirksam. Standardwert ist `SESSION (0)`.

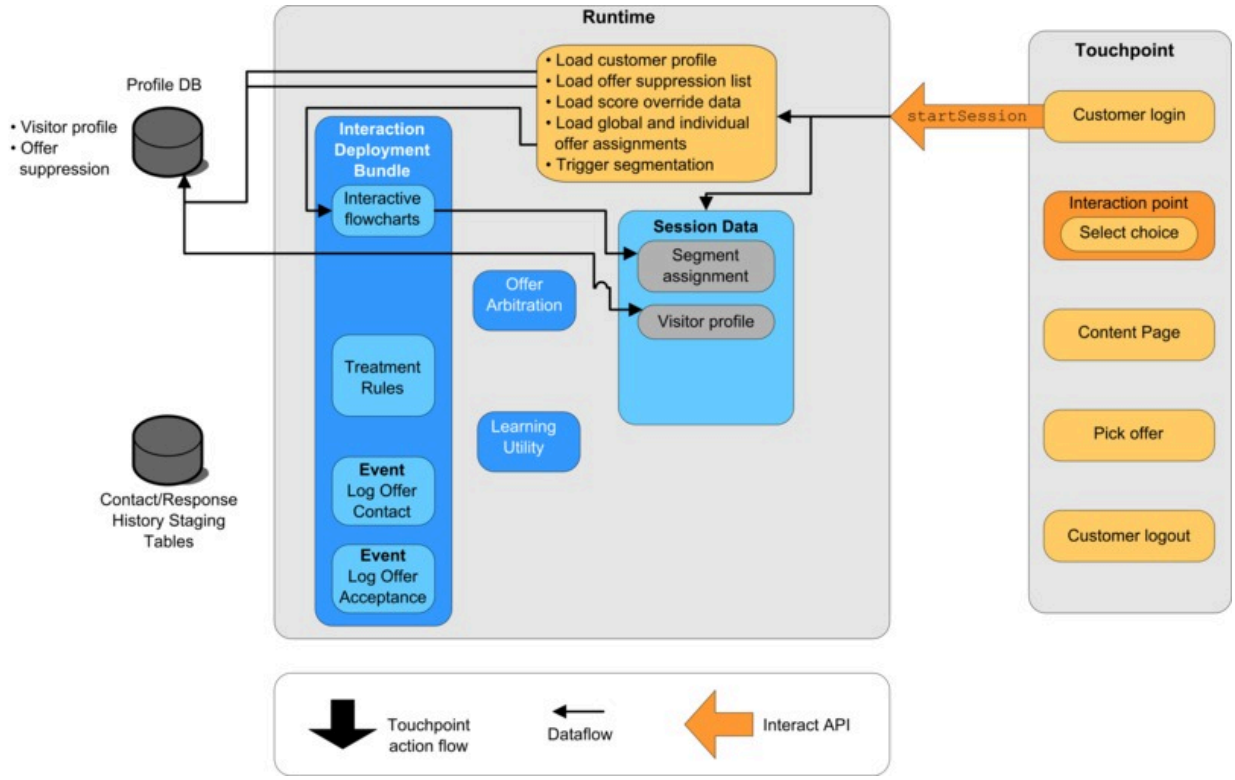
Unica Interact-API-Datenfluss

Dieses Beispiel veranschaulicht, wie die API zwischen Ihrem Touchpoint und der Laufzeitumgebung funktioniert. Der Besucher führt nur vier Aktionen aus - Anmelden, zur Seite mit den Angeboten navigieren, ein Angebot auswählen und Abmelden. Sie können Ihre Integration so kompliziert wie nötig gestalten (innerhalb der Begrenzungen Ihrer Leistungsanforderungen).

Dieses Diagramm zeigt eine einfache Implementierung der Unica Interact-API.

Ein Besucher meldet sich bei einer Website an und navigiert zu einer Seite, auf der Angebote angezeigt werden. Der Besucher wählt ein Angebot aus und meldet sich ab. Auch wenn die Interaktion einfach ist, so gibt es doch mehrere Ereignisse, die im Touchpoint und auf dem Laufzeitserver auftreten:

1. Sitzung starten
2. Zu einer Seite navigieren
3. Angebot auswählen
4. Sitzung schließen



Sitzung starten

Wenn sich ein Besucher anmeldet, löst dies die Methode `startSession` aus.

Die Methode `startSession` führt vier Schritte aus:

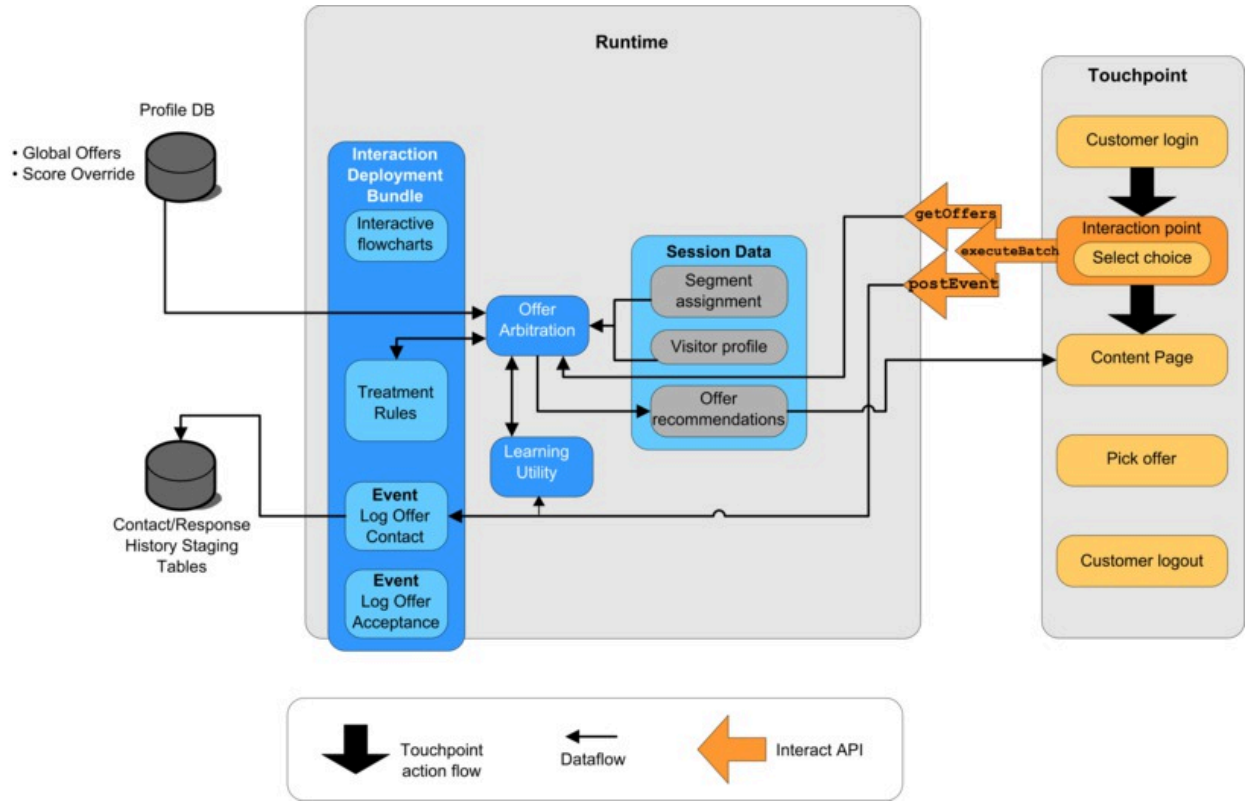
1. Sie erstellt eine neue Laufzeitsitzung.
2. Sie sendet eine Anforderung, die Kundenprofildaten in die Sitzung zu laden.
3. Sie sendet eine Anforderung, die Profildaten zu verwenden und ein interaktives Ablaufdiagramm zu starten, um den Kunden in Segmente zu platzieren. Dieses Ablaufdiagramm wird asynchron ausgeführt.
4. Der Laufzeitserver lädt etwaige Informationen zur Angebotsunterdrückung und globale bzw. individuelle Angebotsverfahrensinformationen in die Sitzung. Die Sitzungsdaten werden für die Dauer der Sitzung im Speicher gehalten.

Zu einer Seite navigieren

Der Besucher navigiert durch die Site, bis er einen vordefinierten Interaktionspunkt erreicht. In der Abbildung ist der zweite Interaktionspunkt (Auswahl der Auswahloption) eine Stelle, an der der Besucher auf einen Link klickt, der eine Gruppe von Angeboten darstellt. Der Touchpoint-Manager hat den Link so konfiguriert, dass die Methode `executeBatch` für die Auswahl eines Angebots ausgelöst wird.

Angebot auswählen

Dieses Diagramm stellt den API-Aufruf dar, der die Methode `executeBatch` auslöst.



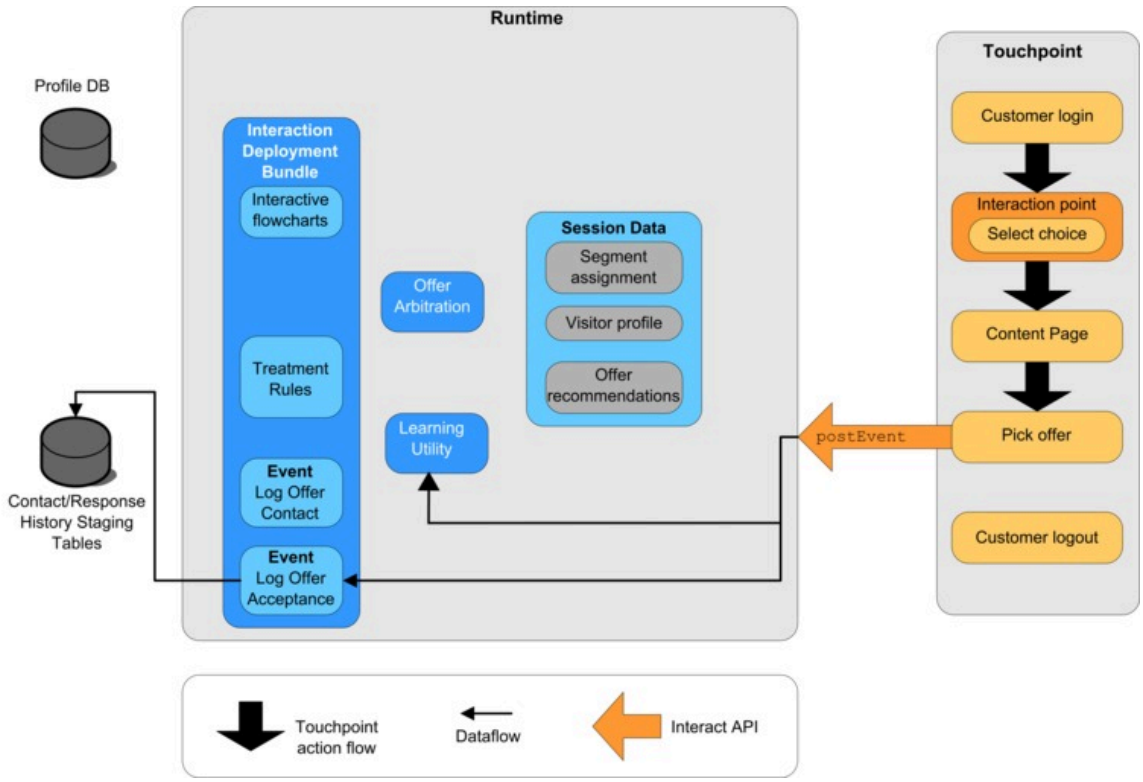
Die Methode `executeBatch` ermöglicht es Ihnen, mehr als eine Methode in einem einzelnen Aufruf an den Laufzeitserver aufzurufen. Diese bestimmte executeBatch-Methode ruft zwei andere Methoden auf, `getOffers` und `postEvent`. Die Methode `getOffers` fordert eine Liste von Angeboten an. Der Laufzeitserver verwendet die Segmentierungsdaten, die Angebotsunterdrückungsliste, die Verfahrensregeln und das Lernmodul, um einen Satz von

Angeboten vorzuschlagen. Der Laufzeitserver gibt einen Satz von Angeboten zurück, der auf der Inhaltsseite angezeigt wird.

Die Methode `postEvent` löst eines der Ereignisse aus, die in der Designumgebung definiert wurden. In diesem bestimmten Fall sendet das Ereignis eine Anforderung, die angezeigten Angebote im Kontaktverlauf zu protokollieren.

Der Besucher wählt eines der Angebote aus (Angebot auswählen).

Dieses Diagramm stellt die Methode `postEvent` dar.

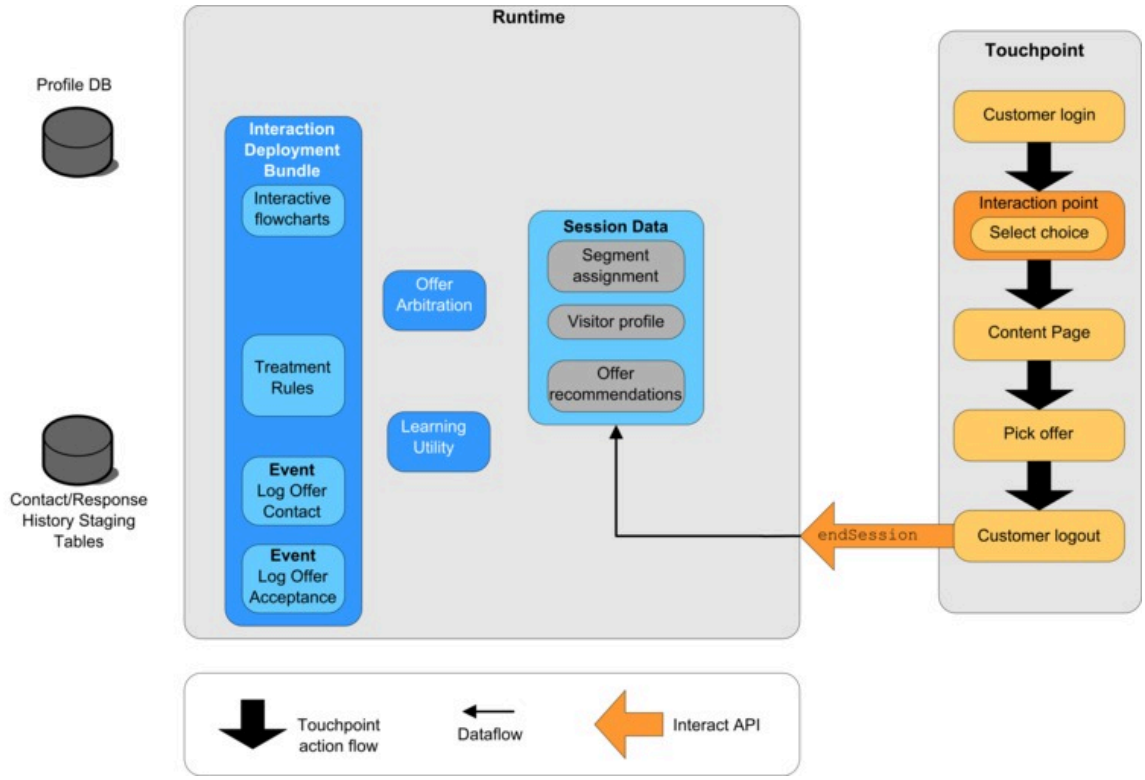


Das der Auswahl des Angebots zugeordnete Steuerelement der Benutzerschnittstelle ist so konfiguriert, dass eine weitere `postEvent`-Methode gesendet wird. Dieses Ereignis sendet eine Anforderung, die Angebotsannahme im Antwortverlauf zu protokollieren.

Sitzung schließen

Nachdem der Besucher das Angebot ausgewählt hat, muss er keine weiteren Aktionen auf der Website ausführen und kann sich abmelden. Der Abmeldebefehl ist mit der Methode `endSession` verknüpft.

Dieses Diagramm stellt die Methode `endSession` dar.



Die Methode `endSession` schließt die Sitzung. Wenn der Besucher vergisst sich abzumelden, gibt es ein konfigurierbares Sitzungszeitlimit um sicherzustellen, dass jede Sitzung einmal endet. Wenn Sie Daten bewahren wollen, die an die Sitzung übergeben werden, wie in Parametern aufgenommene Informationen in der Methode `startSession` oder `setAudience` , wenden Sie sich an die Person, die interaktive Ablaufdiagramme erstellt. Die Person, die ein interaktives Ablaufdiagramm erstellt, kann den Prozess "Momentaufnahme" verwenden, um diese Daten in eine Datenbank zu schreiben, bevor die Sitzung endet und diese Daten verloren gehen. Sie können dann die Methode

`postEvent` verwenden, um das interaktive Ablaufdiagramm aufzurufen, das den Prozess "Momentaufnahme" enthält.

Einfaches Beispiel für Interaktionsplanung

In diesem Beispiel entwerfen Sie eine Interaktion für die Website einer Mobiltelefonfirma. Sie erstellen drei unterschiedliche Angebote, richten eine Protokollierung für die Angebote ein, weisen dem Angebot Verfahrenscodes zu und zeigen mehrere Bilder an, die mit den Angeboten verknüpft werden.

Entwurfsprozess

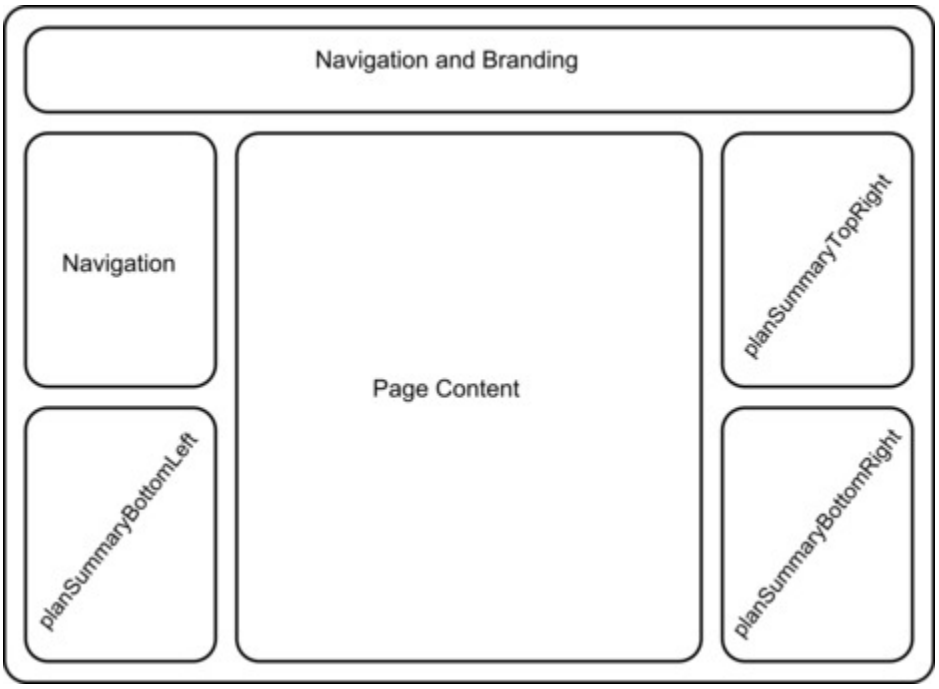
Im Rahmen des Entwurfs einer Interaktion für diesen Kunden können Sie folgende Aktionen ausführen:

1. Identifizieren Sie die Anforderungen für die Übersichtsseite des Kunden.
2. Erstellen Sie Interaktionspunkte für die Angebotsanforderungen.
3. Konfigurieren Sie eine Protokollierung für die Angebote.
4. Erstellen von Verfahrenscodes
5. Verknüpfen Sie eine Reihe von wechselnden Bildern mit den Angeboten

Dies ist ein grundlegendes Beispiel und zeigt nicht die beste Art, die Integration zu schreiben. Beispielsweise umfasst keines dieser Beispiele eine Fehlerprüfung mithilfe der Antwortklasse.

Identifizieren der Anforderungen für die Übersichtsseite des Mobilfunkvertrags

Das folgende Diagramm zeigt das Layout für die Übersichtsseite des Mobilfunkvertrags.



Sie definieren die folgenden Elemente, damit die Anforderungen der Übersichtsseite des Mobilfunkvertrags erfüllt werden:

Anforderung	Implementierung
Ein Angebot, das in einer Zone für Upgradeangebote angezeigt wird Der Bereich auf der Seite, der das Upgradeangebot anzeigt, muss definiert werden. Auch müssen, nachdem Unica Interact ein Angebot zur Anzeige ausgewählt hat, die Informationen protokolliert werden.	• Interaktionspunkt: `ip_planSummaryBottomRight` • Ereignis: `evt_logOffer`
Zwei Angebote für Telefonupgrades Jeder Bereich auf der Seite, der die Telefonupgrades anzeigt, muss definiert werden.	• Interaktionspunkt: `ip_planSummaryTopRight` • Interaktionspunkt: `ip_planSummaryBottomLeft`

Anforderung	Implementierung
Für Analysezwecke müssen Sie protokollieren, welche Angebote angenommen und welche abgelehnt werden.	• Ereignis: `evt_offerAccept` • Ereignis: `evt_offerReject`
Sie wissen auch, dass Sie den Verfahrenscode eines Angebots übergeben müssen, wenn Sie einen Angebotskontakt, ein Annehmen oder eine Ablehnung protokollieren.	`NameValuePair`
Zeigen Sie drei wechselnde Bilder auf der Seite an. Verknüpfen Sie die Bilder mit den Angeboten.	

Erstellen von Interaktionspunkten

Nun können Sie den Designumgebungsbenutzer bitten, für Sie Interaktionspunkte und Ereignisse zu erstellen, während Sie beginnen, die Integration mit Ihrem Touchpoint zu codieren.

Für jeden Interaktionspunkt, der ein Angebot anzeigen wird, müssen Sie zuerst ein Angebot abrufen, um dann die Informationen zu extrahieren, die für die Anzeige des Angebots erforderlich sind. Beispiel: Fordern Sie ein Angebot für den rechten unteren Bereich Ihrer Webseite (`planSummaryBottomRight`) an.

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

Dieser Antwortaufruf gibt ein Antwortobjekt zurück, das eine Antwort des Typs `OfferList` enthält. Ihre Webseite kann jedoch kein Objekt des Typs `OfferList` verwenden. Sie brauchen eine Bilddatei für das Angebot, die bekanntlich eines der Angebotsattribute (`offerImg`) ist. Sie müssen das benötigte Angebotsattribut aus `OfferList` extrahieren.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
```

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

Konfigurieren der Protokollierung

Jetzt, da Sie das Angebot anzeigen, möchten Sie es als einen Kontakt protokollieren.

```
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)
```

Anstatt jede dieser Methoden einzeln aufzurufen, können Sie die Methode `executeBatch` verwenden, wie im folgenden Beispiel für den Abschnitt `planSummaryBottomLeft` der Webseite gezeigt.

```
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);
```

In diesem Beispiel brauchen Sie `UACIOfferTrackingCode` nicht zu definieren. Der Unica Interact-Laufzeitserver protokolliert die letzte empfohlene Liste von Verfahren automatisch als Kontakte, wenn Sie `UACIOfferTrackingCode` nicht bereitstellen.

Erstellen von Verfahrenscodes

Wo notwendig erstellen Sie ein `NameValuePair`, um den Verfahrenscode aufzunehmen, wie im folgenden Beispiel gezeigt.

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

Verknüpfen von Bildern mit Angeboten

Für den zweiten Bereich auf der Seite, der ein Telefonupgrade anzeigt, haben Sie eine Anweisung geschrieben, durch die das angezeigte Bild alle 30 Sekunden wechselt. Da Sie beschließen, zwischen drei Bildern zu wechseln, verwenden Sie folgenden Text, um den Satz von Angeboten abzurufen und ihn in den Cache zu stellen, damit er in Ihrem Code für den Bildwechsel verwendet werden kann.

```
Response response=getOffers(sessionID, ip_planSummaryBottomLeft, 3)
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
```

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

Sie müssen Ihren Kundencode (Client-Code) zum Abrufen aus dem lokalen Cache und zum Protokollieren als Kontakt nur einmal für jedes Angebot schreiben, nachdem das zugehörige Bild angezeigt wird. Um den Kontakt zu protokollieren, muss der Parameter `UACITrackingCode` wie vorhin bereitgestellt werden. Jedes Angebot hat einen anderen Verfolgungscode.

```
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)
;
  }
 }
}
```

Bei jedem Angebot protokollieren Sie, wenn darauf geklickt wird, das angenommene Angebot und die abgelehnten Angebote. (In diesem Szenario werden Angebote, die nicht

explizit ausgewählt werden, als abgelehnt bewertet.) Nachfolgend ein Beispiel, wenn das Angebot `ip_planSummaryTopRight` ausgewählt wird:

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

In der Praxis wäre es das Beste, diese drei `postEvent`-Aufrufe mit der Methode `executeBatch` zu senden.

Entwerfen der Unica Interact-API-Integration

Der Aufbau Ihrer Unica Interact-API-Integration mit Ihrem Touchpoint erfordert einige Arbeitsschritte, bevor Sie mit der Implementierung beginnen können. Sie müssen mit Ihrem Marketingteam zusammenarbeiten um zu entscheiden, wo in Ihrem Touchpoint die Laufzeitumgebung Angebote bereitstellen soll (Definition der Interaktionspunkte) und welche Art von Protokollierung oder interaktive Funktionalität Sie verwenden wollen (Definition Ihrer Ereignisse).

In der Entwurfsphase können dies einfache Grundelemente sein. Bei einer Telekommunikationswebsite beispielsweise sollte die Planübersichtsseite des Kunden ein Angebot in Bezug auf ein Planupgrade und zwei Angebot für Telefonupgrades anzeigen.

Nachdem Ihr Unternehmen entschieden hat, wann und wie es mit Kunden interagieren möchte, müssen Sie Unica Interact verwenden, um die Details zu definieren. Ein Ablaufdiagrammverfasser muss die interaktiven Ablaufdiagramme entwerfen, die verwendet werden, wenn Neusegmentierungsereignisse auftreten. Sie müssen die Anzahl und die Namen der Interaktionspunkte und Ereignisse festlegen sowie entscheiden, welche Daten für ordnungsgemäße Segmentierung, Ereignisbereitstellung und Angebotsabruf übergeben werden sollen. Der Designumgebungsbenutzer definiert die Interaktionspunkte und Ereignisse für den interaktiven Kanal. Sie verwenden dann diese Namen, wenn Sie die Integration mit Ihrem Touchpoint in der Laufzeitumgebung codieren. Sie sollten auch festlegen, welche Metrikinformationen erforderlich sind um zu bestimmen, wann Sie Angebotskontakte und Antworten protokollieren müssen.

Zu berücksichtigende Punkte

Berücksichtigen Sie beim Entwerfen einer Interaktion die Auswirkungen, die nicht auswählbare Angebote, ein nicht erreichbarer Laufzeitserver und das Prozesstiming auf die Interaktion haben. Seien Sie bei der Definition von Angebotsablehnungen möglichst präzise. Berücksichtigen Sie die optionalen Produktfunktionen, die die Interaktion verbessern können.

Führen Sie beim Entwerfen Ihrer Interaktion folgende Schritte aus:

Erstellen Sie einen Standardfüllinhalt

Erstellen Sie einen Standardfüllinhalt (üblicherweise eine freundliche Brandingnachricht oder einen leeren Inhalt) für jeden Interaktionspunkt, an dem Angebote angezeigt werden können. Dieser Füllinhalt wird verwendet, wenn keine Angebote auswählbar sind, die dem aktuellen Besucher in der aktuellen Situation bereitgestellt werden können. Sie weisen diesen Standardfüllinhalt als Standardzeichenfolge für den Interaktionspunkt zu.

Schließen Sie ein Alternativverfahren für die Darstellung von Inhalten ein

Nehmen Sie eine Methode zur Darstellung von Inhalt auf, falls Ihr Touchpoint die Laufzeitservergruppe aus irgendwelchen Gründen nicht erreichen kann.

Berücksichtigen Sie die Dauer einer Ablaufdiagrammausführung

Beim Auslösen von Ereignissen, die Ihren Besucher neu segmentieren (einschließlich `postEvent` und `setAudience`), müssen Sie beachten, dass das Ausführen von Ablaufdiagrammen etwas Zeit beansprucht. Die Methode `getOffers` wartet mit der Ausführung, bis die Segmentierung abgeschlossen ist. Erst dann wird die Methode `getOffers` ausgeführt. Eine allzu häufige Neusegmentierung könnte die Antwortleistung des Aufrufs `getOffers` beeinträchtigen.

Legen Sie fest, was genau eine "Angebotsablehnung" bedeutet

Mehrere Berichte, wie z. B. der Bericht "Übersicht über Kanäle zum Angebotserfolg", geben die Häufigkeit an, mit der ein Angebot abgelehnt wurde. Dieser Bericht gibt Aufschluss darüber, wie oft die Aktion

"Angebotsablehnung protokollieren" von `postEvent` ausgelöst wurde. Sie müssen festlegen, ob die Aktion "Angebotsablehnung protokollieren" für eine tatsächliche Ablehnung gilt. Dies wäre der Fall, wenn beispielsweise auf einen Link mit der Beschriftung **Nein danke** geklickt wurde. Sie kann auch für ein Angebot gelten, das ignoriert wird, z. B. bei einer Seite mit drei Bannerwerbungen, von denen keine ausgewählt wird.

Entscheiden Sie, welche Angebotsauswahlfunktionen verwendet werden sollen

Es gibt mehrere optionale Funktionen, mit denen Sie die Unica Interact-Angebotsauswahl verbessern können. Unter anderem sind folgende Funktionen verfügbar:

- Lernen
- Angebotsunterdrückung
- Individuelle Angebotszuweisungen
- Sonstige Elemente der Angebotsbereitstellung

Sie müssen festlegen, ob eine dieser optionalen Funktionen Ihre Interaktionen bereichern würden.

API-Authentifizierung

Diese Funktion bietet Ihnen die Möglichkeit, die Authentifizierung bei Unica Interact API-Aufrufen zu aktivieren. Wenn sie aktiviert ist, prüft Unica Interact RT, ob eine eingehende API-Anforderung über ein gültiges Authentifizierungstoken verfügt. Ist dies nicht der Fall, wird sie mit dem angegebenen Benutzernamen und Kennwort mit Unica Platform oder LDAP authentifiziert. Der Antrag wird abgelehnt, wenn das Token nicht gültig ist und der Benutzername / das Kennwort nicht gültig ist. Jedes Token verfügt über eine nicht verlängerbare Lebensdauer des Unica Interact Sitzungscaches der eine Zeitüberschreitung aufweist. Das Token ist an die Sitzungs-Id gebunden, die während des startSession()-API-Aufrufs bereitgestellt wird. Daher validieren alle anderen API-Aufrufe dieses Token, vor dem Fortfahren.Diese Funktion ist standardmäßig inaktiviert.

Anmerkung:

Anwendungen von Drittanbietern, die Netzwerkanforderungen abfangen oder modifizieren (z. B. Load Balancer, Firewalls, Proxies, Anforderungsfilter, Antivirus usw.), können Interact API / Bereitstellung / Authentifizierungsanforderungen beeinträchtigen.

Die Interaktionsauthentifizierung kann während der Bereitstellung trotz korrekter Anmeldedaten fehlschlagen, wenn Anwendungen von Drittanbietern (Load Balancer, Firewalls, Proxies, Anforderungsfilter, Antivirus usw.) so konfiguriert sind, dass sie Nachrichten im Netzwerk oder auf dem Anwendungsserver abfangen oder neu schreiben. Dies kann dazu führen, dass Interact-Bereitstellungsnachrichten von einer dritten Partei auf 0 gesetzt werden, obwohl der Inhalt der Authentifizierungsantwort korrekt ist. Campaign hat eine Umgehungslösung bereitgestellt, die den Inhalt unabhängig von der Manipulation des Inhalts liest und so die erfolgreichen Authentifizierungsantworten zur Bereitstellung wiederherstellt, bis der Grund für die Manipulation durch Dritte gefunden ist. Dieses Problem geht nicht auf Interact zurück, sodass Sie sich zur Fehlerbehebung an den Campaign-Support wenden müssen.

Kapitel 7. Verwalten der Unica Interact-API

Immer wenn Sie die Methode `startSession` verwenden, erstellen Sie eine Unica Interact-Laufzeitsitzung auf dem Laufzeitserver. Sie können Konfigurationseigenschaften verwenden, um die Sitzungen auf dem Laufzeitserver zu verwalten.

Sie müssen diese Einstellungen möglicherweise konfigurieren, wenn Sie Ihre Unica Interact-Integration mit Ihrem Touchpoint implementieren.

Diese Konfigurationseigenschaften befinden sich in der Kategorie `sessionManagement`.

Ländereinstellungen und die Unica Interact-API

Sie können Unica Interact für nicht-englische Touchpoints verwenden. Der Touchpoint und alle Zeichenfolgen in der API verwenden die Ländereinstellung, die für den Laufzeitumgebungsbenutzer definiert ist.

Sie können nur eine Ländereinstellung pro Servergruppe auswählen.

Beispiel: In der Laufzeitumgebung erstellen Sie zwei Benutzer, asm_admin_en mit der Benutzerländereinstellung Englisch und asm_admin_fr mit der Benutzerländereinstellung Französisch. Wenn Ihr Touchpoint für französisch Sprechende entwickelt wurde, legen Sie die Eigenschaft `asmUserForDefaultLocale` für die Laufzeitumgebung als asm_admin_fr fest. Wenn die Clientbibliothek (interact_client.jar) verwendet wird, um die Clientanwendung mit Interact-Laufzeitservern zu verbinden, kann optional ein HTTP-Proxy mit Authentifizierung zwischen der Clientanwendung und der Interact-Laufzeit konfiguriert werden. Um den Proxy für Interact-APIs zu aktivieren, fügen Sie unten JVM-Parameter hinzu und starten Sie den Anwendungsserver, auf dem die Clientanwendung bereitgestellt wird, neu.

-Dcom.hcl.interact.http.proxyHost=<IP-Adresse des Proxy-Servers>

-Dcom.hcl.interact.http.proxyPort=<Listener Port des Proxy-Servers>

Geben Sie die unten aufgeführten Parameter an, wenn eine Authentifizierung für den Proxyserver erforderlich ist.

-Dcom.hcl.interact.http.proxyUsername= <Benutzername für die Verbindung zum Proxy-Server. nicht angeben, wenn keine Authentifizierung erforderlich ist>

-Dcom.hcl.interact.http.proxyPassword=<Passwort für die Verbindung zum Proxy-Server. nicht angeben, wenn keine Authentifizierung erforderlich ist>

Informationen zur JMX-Überwachung

Unica Interact stellt den Java™-Management Extensions bereit, auf den Sie mit einer JMX-Überwachungsanwendung zugreifen können. Diese JMX-Überwachung ermöglicht es Ihnen, Ihre Laufzeitserver zu überwachen und zu verwalten.

Die JMX-Attribute stellen eine Menge von Detailinformationen über den Laufzeitserver zur Verfügung. Das JMX-Attribut `ErrorCount` beispielsweise gibt Anzahl von Fehlernachrichten an, die seit der letzten Zurücksetzung oder dem Systemstart protokolliert wurden. Sie können mithilfe dieser Informationen feststellen, wie häufig es Fehler in Ihrem System gibt. Wenn Sie Ihre Website so codiert haben, nur ein End Session aufzurufen, wenn jemand eine Transaktion abschließt, können Sie auch `startSessionCount` mit `endSessionCount` vergleichen um herauszufinden, wie viele Transaktionen unvollständig sind.

Unica Interact unterstützt die RMI- und JMXMP-Protokolle, wie durch JSR 160 definiert. Sie können mit dem JMX-Überwachungsservice über jeden JSR160-kompatiblen JMX-Client verbinden.

Interaktive Ablaufdiagramme können nur mit der JMX-Überwachung überwacht werden. Informationen über interaktive Ablaufdiagramme erscheinen nicht in der Unica Campaign-Überwachung.

Anmerkung: Wenn Sie IBM® WebSphere®mit einem Knotenmanager verwenden, müssen Sie das generische JVM-Argument definieren, um die JMX-Überwachung zu aktivieren.

Konfigurieren von Unica Interact zur Verwendung der JMX-Überwachung mit dem RMI-Protokoll

Verwenden Sie dieses Verfahren, um Unica Interact zur Verwendung der JMX-Überwachung mit dem RMI-Protokoll zu konfigurieren.

Die Standardadresse für die Überwachung mit dem RMI-Protokoll lautet: `service:jmx:rmi:///jndi/rmi://`*`RuntimeServer`*`:`*`port`*`/interact`.

Bearbeiten Sie in Unica Platform für die Laufzeitumgebung folgende Konfigurationseigenschaften in der Kategorie `Interact > Überwachung`.

Konfigurationseigenschaft	Einstellung
`protocol`	**RMI**
`Port`	Die Portnummer für den JMX-Service
`enableSecurity`	**False** Die Unica Interact-Implementierung des RMI-Protokolls unterstützt keine Sicherheit.

Konfigurieren von Unica Interact zur Verwendung der JMX-Überwachung mit dem JMXMP-Protokoll

Verwenden Sie dieses Verfahren, um Unica Interact zur Verwendung der JMX-Überwachung mit dem JMXMP-Protokoll zu konfigurieren.

Das JMXMP-Protokoll erfordert zwei zusätzliche Bibliotheken in der folgenden Reihenfolge im Klassenpfad: `InteractJMX.jar` und `jmxremote_optional.jar`. Beide Dateien befinden sich im `lib`-Verzeichnis Ihrer Laufzeitumgebungsinstallation.

Wenn Sie Sicherheit aktivieren, muss der Benutzername und das Kennwort mit einem Benutzer in Unica Platform für die Laufzeitumgebung übereinstimmen. Sie können kein leeres Kennwort verwenden.

Die Standardadresse für die Überwachung mit dem JMXMP-Protokoll lautet: `service:jmx:jmxmp://RuntimeServer:port`.

1. Überprüfen Sie, ob die Bibliotheken `InteractJMX.jar` und `jmxremote_optional.jar` in der richtigen Reihenfolge im Klassenpfad enthalten sind. Falls sie sich nicht im Klassenpfad befinden, fügen Sie diese dort hinzu.
2. Bearbeiten Sie in Unica Platform für die Laufzeitumgebung folgende Konfigurationseigenschaften in der Kategorie `Interact > Überwachung`.

Konfigurati-onseigenschaft	Einstellung
`protocol`	**JMXMP**
`Port`	Die Portnummer für den JMX-Service
`enableSecurity`	**False**, um die Sicherheit zu inaktivieren, oder **True**, um die Sicherheit zu aktivieren

Konfigurieren von Unica Interact für die Verwendung der jconsole-Scripts zur JMX-Überwachung

Wenn Sie keine separate JMX-Überwachungsanwendung haben, können Sie die mit der JVM installierte jconsole verwenden. Sie können die jconsole mithilfe der Startscripts im Verzeichnis `Interact/tools` starten.

Das jconsole-Script verwendet standardmäßig das JMXMP-Protokoll für die Überwachung. Die Standardeinstellungen für `jconsole.bat` lauten wie folgt:

JMXMP-Verbindung

```
%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%
```

RMI-Verbindung

```
%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. Öffnen Sie `Interact\tools\jconsole.bat` (Windows™) oder `Interact/tools/jconsole.sh` (UNIX) in einem Texteditor.
2. Legen Sie für `INTERACT_LIB` den vollständigen Pfad zum Verzeichnis *`InteractInstallationDirectory`*`/lib` fest.
3. Legen Sie für `HOST` den Hostnamen des Laufzeitservers fest, den Sie überwachen möchten.
4. Legen Sie für `PORT` den Port fest, den Sie für die JMX-Überwachung mit der Eigenschaft `Interact > Überwachung > Port` konfiguriert haben.
5. **Optional:** Wenn Sie das RMI-Protokoll für die Überwachung verwenden, setzen Sie ein Kommentarzeichen vor die JMXMP-Verbindung und entfernen Sie das Kommentarzeichen vor der RMI-Verbindung.

JMX-Attribute

Für die JMX-Überwachung sind mehrere Attribute verfügbar. Zu den Attributen der Designumgebung zählt die Überwachung des Kontakt-/Antwortverlaufs im Zusammenhang mit dem ETL-Prozess. Zu den Attributen der Laufzeitumgebung zählen Ausnahmebedingungen, verschiedene Ablaufdiagrammattribute, Ländereinstellung, Protokollprozess und Statistikdaten zum Thread-Pool. Darüber hinaus sind einige Attribute für Servicestatistiken verfügbar. Alle von der JMX-Überwachung bereitgestellten Daten gelten ab der letzten Zurücksetzung oder ab dem Systemstart. Eine Zählung beispielsweise gilt für die Anzahl der Elemente seit der letzten Zurücksetzung oder dem Systemstart, und nicht seit der Installation.

Attribute des ETL-Monitors für den Kontakt-/Antwortverlauf

Die Attribute des ETL-Monitors für den Kontakt-/Antwortverlauf sind Teil der Designumgebung. Alle folgenden Attribute sind Teil der Laufzeitumgebung.

Tabelle 10. ETL-Monitor für den Kontakt-/Antwortverlauf

Attribut	Syntax
AvgCHExecutionTime	Die durchschnittliche Zahl von Millisekunden, die das Kontakt- und Antwortverlaufsmodul benötigt, um in die Kontaktverlaufstabelle zu schreiben. Dieser Durchschnitt wird nur für Operationen berechnet, die erfolgreich waren und für die zumindest ein Datensatz in die Kontaktverlaufstabelle geschrieben wurde.
AvgETLExecutionTime	Die durchschnittliche Zahl von Millisekunden, die das Kontakt- und Antwortverlaufsmodul benötigt, um Daten aus der Laufzeitumgebung zu lesen. Der Durchschnitt umfasst die Zeit für sowohl erfolgreiche als auch fehlgeschlagene Operationen.
AvgRHExecutionTime	Die durchschnittliche Zahl von Millisekunden, die das Kontakt- und Antwortverlaufsmodul benötigt, um in die Antwortverlaufstabelle zu schreiben. Dieser Durchschnitt wird nur für Operationen berechnet, die erfolgreich waren und für die zumindest ein Datensatz in die Antwortverlaufstabelle geschrieben wurde.
ErrorCount	Die Anzahl von Fehlernachrichten, die seit der letzten Zurücksetzung

Tabelle 10. ETL-Monitor für den Kontakt-/Antwortverlauf (Fortsetzung)

Attribut	Syntax
	oder dem Systemstart protokolliert wurden, falls vorhanden.
HighWaterMarkCHExecutionTime	Die maximale Zahl von Millisekunden, die das Kontakt- und Antwortverlaufsmodul benötigte, um in die Kontaktverlaufstabelle zu schreiben. Dieser Wert wird nur für Operationen berechnet, die erfolgreich waren und für die zumindest ein Datensatz in die Kontaktverlaufstabelle geschrieben wurde.
HighWaterMarkETLExecutionTime	Die maximale Zahl von Millisekunden, die das Kontakt- und Antwortverlaufsmodul benötigte, um Daten aus der Laufzeitumgebung zu lesen. Die Berechnung umfasst sowohl erfolgreiche als auch fehlgeschlagene Operationen.
HighWaterMarkRHExecutionTime	Die maximale Zahl von Millisekunden, die das Kontakt- und Antwortverlaufsmodul benötigte, um in die Antwortverlaufstabelle zu schreiben. Dieser Wert wird nur für Operationen berechnet, die erfolgreich waren und für die zumindest ein Datensatz in die Antwortverlaufstabelle geschrieben wurde.

Tabelle 10. ETL-Monitor für den Kontakt-/Antwortverlauf (Fortsetzung)

Attribut	Syntax
LastExecutionAverage	Die Anzahl der Millisekunden, die das Kontakt- und Antwortverlaufsmodul benötigte, um jeden Kopiervorgang durchzuführen.
NumberOfExecutions	Die Häufigkeit, mit der das Kontakt- und Antwortverlaufsmodul seit der Initialisierung ausgeführt wurde.
LastExecutionStart	Die Uhrzeit, zu der die letzte Ausführung des Kontakt- und Antwortverlaufsmoduls gestartet wurde.
LastExecutionSuccessful	Bei true: Die letzte Ausführung des Kontakt- und Antwortverlaufsmoduls war erfolgreich. Bei false: ein Fehler trat auf.
NumberOfContactHistoryRecordsMarked	Die Anzahl von Kontaktverlaufsdatensätzen in der Tabelle `UACI_CHStaging`, die während der letzten Ausführung des Kontakt- und Antwortverlaufsmoduls verschoben wurden. Dieser Wert ist nur größer als null, wenn das Kontakt- und Antwortverlaufsmodul aktiv ist.
NumberOfResponseHistoryRecordsMarked	Die Anzahl von Antwortverlaufsdatensätzen in der Tabelle `UACI_RHStaging`, die während der letzten Ausführung des Kontakt- und Antwortverlaufsmoduls verschoben wurden. Dieser Wert ist nur größer als null,

Tabelle 10. ETL-Monitor für den Kontakt-/Antwortverlauf (Fortsetzung)

Attribut	Syntax
	wenn das Kontakt- und Antwortverlaufsmodul aktiv ist.

Ausnahmebedingungsattribute

Die Ausnahmebedingungsattribute sind Bestandteil der Laufzeitumgebung.

Tabelle 11. Ausnahmebedingungen

Attribut	Syntax
errorCount	Die Anzahl von Fehlernachrichten, die seit der letzten Zurücksetzung oder dem Systemstart protokolliert wurden.
warningCount	Die Anzahl von Warnhinweisen, die seit der letzten Zurücksetzung oder dem Systemstart protokolliert wurden.

Attribute für Ablaufdiagramm-Engine-Statistikdaten

Die Attribute für Ablaufdiagramm-Engine-Statistikdaten sind Bestandteil der Laufzeitumgebung.

Tabelle 12. Ablaufdiagramm-Engine-Statistikdaten

Attribut	Syntax
activeProcessBoxThreads	Aktive Zählung von Ablaufdiagrammverarbeitungsthreads (von allen Ausführungen gemeinsam genutzt), die derzeit aktiv sind.

Tabelle 12. Ablaufdiagramm-Engine-Statistikdaten (Fortsetzung)

Attribut	Syntax
activeSchedulerThreads	Aktive Zählung von Ablaufdiagramm-Scheduler-Threads, die derzeit aktiv sind.
avgExecutionTimeMillis	Durchschnittliche Ablaufdiagrammlaufzeit in Millisekunden.
CurrentJobsInProcessBoxQueue	Die Anzahl von Jobs, die auf die Ausführung durch Ablaufdiagrammverarbeitungsthreads warten.
CurrentJobsInSchedulerQueue	Die Anzahl von Jobs, die auf die Ausführung durch Ablaufdiagramm-Scheduler-Threads warten.
maximumProcessBoxThreads	Maximale Anzahl von Ablaufdiagrammverarbeitungsthreads (von allen Ausführungen gemeinsam genutzt), die ausgeführt werden können.
maximumSchedulerThreads	Maximale Anzahl von Ablaufdiagramm-Scheduler-Threads (ein Thread pro Ausführung), die ausgeführt werden können.
numExecutionsCompleted	Die Gesamtzahl der Ablaufdiagrammausführungen, die abgeschlossen wurden.
numExecutionsStarted	Die Gesamtzahl der Ablaufdiagrammausführungen, die gestartet wurden.

Attribute für spezielle Ablaufdiagramme nach interaktivem Kanal

Die Attribute für spezielle Ablaufdiagramme nach interaktivem Kanal sind Bestandteil der Laufzeitumgebung.

Tabelle 13. Spezielle Ablaufdiagramme nach interaktivem Kanal

Attribut	Syntax
AvgExecutionTimeMillis	Durchschnittliche Laufzeit in Millisekunden für dieses Ablaufdiagramm in diesem interaktiven Kanal.
HighWaterMarkForExecutionTime	Maximale Laufzeit in Millisekunden für dieses Ablaufdiagramm in diesem interaktiven Kanal.
LastCompletedExecutionTimeMillis	Laufzeit in Millisekunden für die letzte Ausführung dieses Ablaufdiagramms in diesem interaktiven Kanal.
NumExecutionsCompleted	Gesamtzahl von Ausführungen, die für dieses Ablaufdiagramm in diesem interaktiven Kanal abgeschlossen wurden.
NumExecutionsStarted	Gesamtzahl von Ausführungen, die für dieses Ablaufdiagramm in diesem interaktiven Kanal gestartet wurden.

Ländereinstellungsattribute

Die Ländereinstellungsattribute sind Bestandteil der Laufzeitumgebung.

Tabelle 14. Ländereinstellung

Attribut	Syntax
Ländereinstellung	Ländereinstellung für den JMX-Client.

Attribute für die Protokollprozesskonfiguration

Die Attribute für die Protokollprozesskonfiguration sind Bestandteil der Laufzeitumgebung.

Tabelle 15. Protokollprozesskonfiguration

Attribut	Syntax
Kategorie	Ändern der Protokollkategorie, in der die Protokollebene manipuliert werden kann.

Attribute für Services-Thread-Pool-Statistikdaten

Die Attribute für Services-Thread-Pool-Statistikdaten sind Bestandteil der Laufzeitumgebung.

Tabelle 16. Services-Thread-Pool-Statistikdaten

Attribut	Syntax
activeContactHistThreads	Die näherungsweise berechnete Anzahl von Threads, die aktiv Aufgaben für den Kontakt- und Antwortverlauf ausführen.
activeFlushCacheToDBThreads	Die näherungsweise berechnete Anzahl von Threads, die aktiv Aufgaben ausführen, um Cache-Statistikdaten in den Datenspeicher zu schreiben.
activeOtherStatsThreads	Die näherungsweise berechnete Anzahl von Threads, die aktiv Aufga-

Tabelle 16. Services-Thread-Pool-Statistikdaten (Fortsetzung)

Attribut	Syntax
	ben für die Services zur Erstellung einer Berechtigungsstatistik (Eligibility Statistics), zur Erstellung einer Ereignisaktivitätsstatistik (Event Activity Statistics) und zur Erstellung einer Statistik über die Verwendung von Standardzeichenfolgen (Default Statistics) ausführen.
CurrentHighWaterMarkInContactHistQueue	Größte Anzahl von Einträgen, die in der Warteschlange eingereiht sind, um von dem Service protokolliert zu werden, der die Kontakt- und Antwortverlaufsdaten erfasst.
CurrentHighWaterMark InFlushCachetoDBQueue	Größte Anzahl von Einträgen, die in der Warteschlange eingereiht sind, um von dem Service protokolliert zu werden, der die Daten im Cache in die Datenbanktabellen schreibt.
CurrentHighWaterMarkInOtherStatsQueue	Größte Anzahl von Einträgen, die in der Warteschlange eingereiht sind, um von dem Service protokolliert zu werden, der die Berechtigungsstatistiken für Angebote, Statistiken zur Verwendung von Standardzeichenfolgen, Ereignisaktivitätsstatistiken und die aus einem benutzerdefinierten Protokoll in eine Tabelle konvertierten Daten erfasst.

Tabelle 16. Services-Thread-Pool-Statistikdaten (Fortsetzung)

Attribut	Syntax
currentMsgsInContactHistQueue	Die Anzahl von Jobs in der Warteschlange für den Thread-Pool, der für den Kontakt- und Antwortverlauf verwendet wird.
currentMsgsInFlushCacheToDBQueue	Die Anzahl von Jobs in der Warteschlange für den Thread-Pool, der zum Schreiben von Cache-Statistikdaten in den Datenspeicher verwendet wird.
currentMsgsInOtherStatsQueue	Die Anzahl von Jobs in der Warteschlange für den Thread-Pool, der für die Services zur Erstellung einer Berechtigungsstatistik (Eligibility Statistics), zur Erstellung einer Ereignisaktivitätsstatistik (Event Activity Statistics) und zur Erstellung einer Statistik über die Verwendung von Standardzeichenfolgen (Default Statistics) verwendet wird.
maximumContactHistThreads	Die größte Anzahl von Threads, die sich jemals gleichzeitig in dem Pool befanden, der für den Kontakt- und Antwortverlauf verwendet wird.
maximumFlushCacheToDBThreads	Die größte Anzahl von Threads, die sich jemals gleichzeitig in dem Pool befanden, der für das Schreiben von Cache-Statistikdaten in den Daten-

Tabelle 16. Services-Thread-Pool-Statistikdaten (Fortsetzung)

Attribut	Syntax
	speicher (Flushoperation) verwendet wird.
maximumOtherStatsThreads	Die größte Anzahl von Threads, die sich jemals gleichzeitig in dem Pool befanden, der für die Services zur Erstellung einer Berechtigungsstatistik (Eligibility Statistics), zur Erstellung einer Ereignisaktivitätsstatistik (Event Activity Statistics) und zur Erstellung einer Statistik über die Verwendung von Standardzeichenfolgen (Default Statistics) verwendet wird.

Attribute für Servicestatistiken

Die Servicestatistiken bestehen aus einem Satz von Attributen für jeden Service.

- ContactHistoryMemoryCacheStatistics - Der Service, der Daten für die Kontaktverlaufs-Staging-Tabellen erfasst.
- CustomLoggerStatistics - Der Service, der benutzerdefinierte Daten erfasst, um sie in eine Tabelle zu schreiben (ein Ereignis, das den Ereignisparameter`UACICustomLoggerTableName` verwendet).
- Default Statistics - Der Service, der Statistiken dazu erfasst, wie oft die Standardzeichenfolge für den Interaktionspunkt verwendet wurde.
- Eligibility Statistics - Der Service, der Statistiken über berechtigte Angebote schreibt.
- Event Activity Statistics - Der Service, der die Ereignisstatistiken erfasst, sowohl für Systemereignisse wie `getOffer` oder `startSession` als auch für Benutzerereignisse, die durch `postEvent` ausgelöst werden.

- Response History Memory Cache Statistics - Der Service, der Daten in die Antwortverlaufs-Staging-Tabellen schreibt.
- Cross-session Response Statistics - Der Service, der sitzungsübergreifende Antwortverfolgungsdaten erfasst.

Tabelle 17. Servicestatistiken

Attribut	Syntax
Anzahl	Die Anzahl der verarbeiteten Nachrichten.
ExecTimeInsideMutex	Die Zeitspanne in Millisekunden, die für die Verarbeitung von Nachrichten für diesen Service benötigt wurde, außer Wartezeiten auf andere Threads. Wenn es zwischen ExecTimeInsidMutex und ExecTimeMillis eine große Differenz gibt, müssen Sie möglicherweise die Thread-Pool-Größe für diesen Service ändern.
ExecTimeMillis	Die Zeitspanne in Millisekunden, die für die Verarbeitung von Nachrichten für diesen Service benötigt wurde, einschließlich Wartezeiten auf andere Threads.
ExecTimeOfDBInsertOnly	Die Zeitspanne in Millisekunden, die nur für die Verarbeitung des Stapeleinfügungsschritts benötigt wurde.
HighWaterMark	Die maximale Anzahl von Nachrichten, die für diesen Service verarbeitet wurden.

Tabelle 17. Servicestatistiken (Fortsetzung)

Attribut	Syntax
NumberOfDBInserts	Die Gesamtzahl der ausgeführten Stapeleinfügungen.
TotalRowsInserted	Die Gesamtzahl von Zeilen, die in die Datenbank eingefügt wurden.

Attribute für Servicestatistiken - Datenbankladedienstprogramm

Die Attribute für "Servicestatistiken - Datenbankladedienstprogramm" sind Bestandteil der Laufzeitumgebung.

Tabelle 18. Servicestatistiken - Datenbankladedienstprogramm

Attribut	Syntax
ExecTimeOfWriteToCache	Die Zeitspanne in Millisekunden, die zum Schreiben in den Dateicache benötigt wurde, einschließlich des Schreibens in Dateien und des Abrufens des Primärschlüssels aus der Datenbank, falls erforderlich.
ExecTimeOfLoaderDBAccessOnly	Die Zeitspanne in Millisekunden, die nur für das Ausführen des Datenbankladeprogrammschritts benötigt wurde.
ExecTimeOfLoaderThreads	Die Zeitspanne in Millisekunden, die von den Datenbankladeprogrammthreads benötigt wurde.
ExecTimeOfFlushCacheFiles	Die Zeitspanne in Millisekunden, die zum Leeren des Cache und zur Neuerstellung von neuen Caches benötigt wurde.

Tabelle 18. Servicestatistiken - Datenbankladedienstprogramm (Fortsetzung)

Attribut	Syntax
ExecTimeOfRetrievePKDBAccess	Die Zeitspanne in Millisekunden, die für das Abrufen des Primärschlüsseldatenbankzugriffs benötigt wurde.
NumberOfDBLoaderRuns	Die Gesamtzahl der Datenbankladeprogrammausführungen.
NumberOfLoaderStagingDirCreated	Die Gesamtzahl der erstellten Staging-Verzeichnisse.
NumberOfLoaderStagingDirRemoved	Die Gesamtzahl der entfernten Staging-Verzeichnisse.
NumberOfLoaderStaging DirMovedToAttention	Die Gesamtzahl der Staging-Verzeichnisse, die in den Warnungszustand versetzt wurden.
NumberOfLoaderStagingDirMovedToError	Die Gesamtzahl der Staging-Verzeichnisse, die in den Fehlerzustand versetzt wurden.
NumberOfLoaderStagingDirRecovered	Die Gesamtzahl von wiederhergestellten Staging-Verzeichnissen, einschließlich zur Startzeit und erneute Ausführung durch Hintergrundthreads.
NumberOfTimesRetrievePKFromDB	Die Gesamtzahl von Vorgängen zum Abrufen des Primärschlüssels aus der Datenbank.
NumberOfLoaderThreadsRuns	Die Gesamtzahl der Datenbankladeprogrammthreadausführungen.
NumberOfFlushCacheFiles	Die Gesamtzahl von Leerungen des Dateicache.

Attribute für API-Statistiken

Die Attribute für API-Statistiken sind Bestandteil der Laufzeitumgebung.

Tabelle 19. API-Statistiken

Attribut	Syntax
endSessionCount	Die Anzahl von `endSession`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
endSessionAverage	Die beim letzten `endSession`-API-Aufruf verstrichene Zeit in Millisekunden.
executeBatchCount	Die Anzahl von `executeBatch`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
executeBatchAverage	Die beim letzten `executeBatch`-API-Aufruf verstrichene Zeit in Millisekunden.
getOffersCount	Die Anzahl von `getOffers`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
getOffersAverage	Die beim letzten `getOffer`-API-Aufruf verstrichene Zeit in Millisekunden.
getProfileCount	Die Anzahl von `getProfile`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
getProfileAverage	Die beim letzten `getProfileAverage`-API-Aufruf verstrichene Zeit in Millisekunden.
getVersionCount	Die Anzahl von `getVersion`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.

Tabelle 19. API-Statistiken (Fortsetzung)

Attribut	Syntax
getVersionAverage	Die beim letzten `getVersion`-API-Aufruf verstrichene Zeit in Millisekunden.
loadOfferSuppressionAverage	Die beim letzten `loadOfferSuppression`-API-Aufruf verstrichene Zeit.
LoadOffersBySQLCount	Die Anzahl von `LoadOffersBySQL`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
LoadOffersBySQLAverage	Die beim letzten `LoadOffersBySQL`-API-Aufruf verstrichene Zeit in Millisekunden.
loadProfileAverage	Die beim letzten `loadProfile`-API-Aufruf verstrichene Zeit in Millisekunden.
loadScoreOverrideAverage	Die beim letzten `loadScoreOverride`-API-Aufruf verstrichene Zeit in Millisekunden.
postEventCount	Die Anzahl von `postEvent`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
postEventAverage	Die beim letzten `postEvent`-API-Aufruf verstrichene Zeit in Millisekunden.
runSegmentationAverage	Die beim letzten `runSegmentation`-API-Aufruf verstrichene Zeit in Millisekunden.
setAudienceCount	Die Anzahl von `setAudience`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
setAudienceAverage	Die beim letzten `setAudience`-API-Aufruf verstrichene Zeit in Millisekunden.

Tabelle 19. API-Statistiken (Fortsetzung)

Attribut	Syntax
setDebugCount	Die Anzahl von `setDebug`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
setDebugAverage	Die beim letzten `setDebug`-API-Aufruf verstrichene Zeit in Millisekunden.
startSessionCount	Die Anzahl von `startSession`-API-Aufrufen seit der letzten Zurücksetzung oder dem Systemstart.
startSessionAverage	Die beim letzten `startSession`-API-Aufruf verstrichene Zeit in Millisekunden.
ActiveSessionCount	Die Anzahl Sitzungen, die zurzeit in der Interact-Laufzeitinstanz aktiv sind. **Anmerkung:** ActiveSessionCount in JMX MBean com.unicacorp.interact:type=api, group=Statistics berücksichtigt keine Ereignisse, deren zulässiges Zeitlimit überschritten wurde, und daher kann eine falsche aktive Anzahl angezeigt werden.

Attribute für Statistikdaten des Lernoptimierungsprogramms

Die Attribute für Statistikdaten des Lernoptimierungsprogramms sind Bestandteil der Laufzeitumgebung.

Tabelle 20. Statistikdaten des Lernoptimierungsprogramms

Attribut	Syntax
LearningOptimizerAcceptCalls	Die Anzahl von an das Lernmodul übergebenen Annahmeereignissen.
LearningOptimizer AcceptTrackingAverage	Die Gesamtzahl von Millisekunden, die zum Protokollieren der Annahmeereignisse in das Lernmodul benötigt wurde.
LearningOptimizerContactCalls	Die Anzahl von an das Lernmodul übergebenen Kontaktereignissen.
LearningOptimizer ContactTrackingAverage	Die Gesamtzahl von Millisekunden, die zum Protokollieren der Kontaktereignisse in das Lernmodul benötigt wurde.
LearningOptimizerLogOtherCalls	Die Anzahl von an das Lernmodul übergebenen Nicht-Kontakt- und Nicht-Annahmeereignissen.
LearningOptimizer LogOtherTrackingAverage	Die durchschnittliche Zeit in Millisekunden, die zum Protokollieren von anderen Ereignissen (Nicht-Kontakt und Nicht-Annahme) in das Lernmodul benötigt wurde.
LearningOptimizer NonRandomCalls	Die Häufigkeit, mit der die konfigurierte Learning-Implementierung angewandt wurde.
LearningOptimizer RandomCalls	Die Häufigkeit, mit der die konfigurierte Learning-Implementierung umgangen und eine Zufallsauswahl angewandt wurde.
LearningOptimizer RecommendCalls	Die Anzahl von an das Lernmodul übergebenen Empfehlungsanforderungen.

Tabelle 20. Statistikdaten des Lernoptimierungsprogramms (Fortsetzung)

Attribut	Syntax
LearningOptimizer RecommendAverage	Die Gesamtzahl von Millisekunden, die für die Learning-Empfehlungslogik benötigt wurde.

Attribute für Standardangebotsstatistikdaten

Die Attribute für Standardangebotsstatistikdaten sind Bestandteil der Laufzeitumgebung.

Tabelle 21. Standardangebotsstatistikdaten

Attribut	Syntax
LoadDefaultOffersAverage	Die beim Laden der Standardangebote verstrichene Zeit.
DefaultOffersCalls	Die Häufigkeit des Standardangebotsladens.

Attribute für Dispatcher für ausgelöste Nachrichten

Attribute für Dispatcher für ausgelöste Nachrichten sind Bestandteil der Laufzeitumgebung.

Tabelle 22. Dispatcher für ausgelöste Nachrichten

Attribut	Syntax
NumRequested	Die Gesamtzahl der Angebote, bei denen angefordert wurde, sie mit diesem Dispatcher zu senden.
NumDispatched	Die Gesamtzahl der Angebote, die von diesem Dispatcher erfolgreich gesendet wurden.
AvgExecutionTime	Die durchschnittliche Zeit in Millisekunden, die dieser Dispatcher für

Tabelle 22. Dispatcher für ausgelöste Nachrichten (Fortsetzung)

Attribut	Syntax
	das Senden eines Angebots benötigt. In der Berechnung werden nur die Angebote berücksichtigt, die erfolgreich an Gateways gesendet wurden.
CurrentQueueSize	Die Anzahl der Angebote, die derzeit darauf warten, gesendet zu werden.
GatewayInvocation	Die Anzahl der Angebote mit der durchschnittlichen Sendezeit in Millisekunden, die mit diesem Dispatcher an die einzelnen Gateways gesendet werden. Das Format seines Wertes ist `{gateway name=[number of offers, average dispatching time]}`.

Attribute für Gateways für ausgelöste Nachrichten

Attribute für Gateways für ausgelöste Nachrichten sind Bestandteil der Laufzeitumgebung.

Tabelle 23. Gateways für ausgelöste Nachrichten

Attribut	Syntax
NumValidationRequested	Die Gesamtzahl der Angebote, die von diesem Gateway zur Validierung angefordert wurden.
NumValidated	Die Gesamtzahl der Angebote, die von diesem Gateway erfolgreich validiert wurden.

Tabelle 23. Gateways für ausgelöste Nachrichten (Fortsetzung)

Attribut	Syntax
AvgValidationTime	Die durchschnittliche Zeit in Millisekunden, die dieses Gateway für die Validierung eines Angebots benötigt. In der Berechnung werden nur die Angebote berücksichtigt, die erfolgreich validiert wurden.
NumDeliveryRequested	Die Gesamtzahl der Angebote, die von diesem Gateway für die Bereitstellung angefordert wurden.
NumDelivered	Die Gesamtzahl der Angebote, die von diesem Gateway erfolgreich bereitgestellt wurden.
AvgDeliveryTime	Die durchschnittliche Zeit in Millisekunden, die dieses Gateway für die Bereitstellung eines Angebots benötigt. In der Berechnung werden nur die Angebote berücksichtigt, die erfolgreich bereitgestellt wurden.

Nachrichtenattribute für ausgelöste Nachrichten

Nachrichtenattribute für ausgelöste Nachrichten sind Bestandteil der Laufzeitumgebung.

Tabelle 24. Nachrichten für ausgelöste Nachrichten

Attribut	Syntax
ProcessSuccessCount	Die Gesamthäufigkeit, mit der diese ausgelöste Nachricht erfolgreich ausgeführt wurde.

Tabelle 24. Nachrichten für ausgelöste Nachrichten (Fortsetzung)

Attribut	Syntax
AvgSuccessProcessTime	Die durchschnittliche Zeit in Millisekunden, die diese ausgelöste Nachricht für jede erfolgreiche Ausführung benötigt.
ProcessErrorCount	Die Gesamthäufigkeit, mit der diese ausgelöste Nachricht nicht erfolgreich ausgeführt wurde.
AvgErrorProcessTime	Die durchschnittliche Zeit in Millisekunden, die diese ausgelöste Nachricht für jede nicht erfolgreiche Ausführung benötigt.
SelectBranchCount	Die Gesamthäufigkeit, mit der während der Verarbeitung ausgelöster Nachrichten Zweige ausgewählt wurden.
AvgSelectBranchTime	Die durchschnittliche Zeit in Millisekunden, die während der Verarbeitung ausgelöster Nachrichten für die Zweigauswahl benötigt wird.
SelectOfferCount	Die Gesamthäufigkeit, mit der während der Verarbeitung ausgelöster Nachrichten Angebote ausgewählt wurden.
AvgSelectOfferTime	Die durchschnittliche Zeit in Millisekunden, die während der Verarbeitung ausgelöster Nachrichten für die Angebotsauswahl benötigt wird.

Tabelle 24. Nachrichten für ausgelöste Nachrichten (Fortsetzung)

Attribut	Syntax
SelectChannelCount	Die Gesamthäufigkeit, mit der während der Verarbeitung ausgelöster Nachrichten Kanäle ausgewählt wurden.
AvgSelectChannelTime	Die durchschnittliche Zeit in Millisekunden, die während der Verarbeitung ausgelöster Nachrichten für die Kanalauswahl benötigt wird.
FlowchartWaitCount	Die Gesamthäufigkeit, mit der diese ausgelöste Nachricht darauf gewartet hat, dass die Segmentierung abgeschlossen wird.
AvgFlowchartWaitTime	Die durchschnittliche Zeit in Millisekunden, die diese Nachricht darauf gewartet hat, dass die Segmentierung abgeschlossen wird.
WaitFlowchartTimeoutCount	Die Gesamthäufigkeit, mit der das zulässige Zeitlimit dieser Nachricht überschritten wurde, während sie darauf gewartet hat, dass die Segmentierung abgeschlossen wird.

Tabelle 25. Activity-Orchestrator-Gateways Informationen

Attribut	Syntax
NumReceived	Die Anzahl der vom Activity-Orchestrator-Empfänger empfangenen Nachrichten.

Tabelle 25. Activity-Orchestrator-Gateways Informationen (Fortsetzung)

Attribut	Syntax
NumProcessed	Die Anzahl der verarbeiteten Nachrichten für den Activity-Orchestrator-Empfänger.
AvgProcessTime	Die durchschnittliche Verarbeitungszeit der Anzahl der verarbeiteten Nachrichten für den Activity-Orchestrator-Empfänger.

Tabelle 26. Kafka-Statistik Informationen

Attribute	Syntax
NumMessagesInTopic	Die Anzahl der im Thema aufgezeichneten Nachrichten zusammen mit dem Namen des Themas.
Aktiv	Der Betriebsstatus des Kafka-Servers.
TopicDetails	Die Details zu den Themen zusammen mit dem Leiter und den Anhängern.
ListOfTopics	Die Liste der Themen auf dem Kafka-Server, mit Ausnahme des Standardthemas.

Tabelle 27. Zookeeper-Statistik Informationen

Attribut	Syntax
Aktiv	Der Betriebsstatus des Zookeeper-Servers.

JMX-Operationen

Für die JMX-Überwachung sind mehrere Operationen verfügbar.

In der folgenden Tabelle werden die Operationen, die für die JMX-Überwachung verfügbar sind, beschrieben.

Gruppe	Attribut	Syntax
Protokollprozesskonfiguration	activateDebug	Setzt die Protokollebene für die in `Interact/conf/interact_log4j.properties` definierte Protokolldatei auf "debug".
Protokollprozesskonfiguration	activateError	Setzt die Protokollebene für die in `Interact/conf/interact_log4j.properties` definierte Protokolldatei auf "error".
Protokollprozesskonfiguration	activateFatal	Setzt die Protokollebene für die in `Interact/conf/interact_log4j.properties` definierte Protokolldatei auf "fatal".
Protokollprozesskonfiguration	activateInfo	Setzt die Protokollebene für die in `Interact/conf/interact_log4j.properties` definierte Protokolldatei auf "info".
Protokollprozesskonfiguration	activateTrace	Setzt die Protokollebene für die in `Interact/conf/interact_log4j.properties` definierte Protokolldatei auf "trace".
Protokollprozesskonfiguration	activateWarn	Setzt die Protokollebene für die in `Interact/conf/interact_log4j.properties` definierte Protokolldatei auf "warn".
Ländereinstellung	changeLocale	Ändert die Ländereinstellung des JMX-Clients. Unica Interact unterstützte Ländereinstellungen sind: `de`, `en`, `es`, und `fr`.
ContactResponseHistoryETLMonitor	Zurücksetzen	Setzt alle Zähler zurück.

Gruppe	Attribut	Syntax
Standardangebotsstatistikdaten	updatePollPeriod	Aktualisiert defaultOfferUpdatePollPeriod. Dieser Wert (in Sekunden) teilt dem System mit, wie lange es warten soll, bevor es Standardangebote im Cache aktualisiert. Bei der Einstellung `-1` liest das System die Anzahl der Standardangebote nur beim Starten.

Thread Überwachung

Zur Überwachung der Systemaktivitäten, wird der Link 'Thread Info' zu der Seite 'Admin' der Benutzeroberfläche von Interact Laufzeit hinzugefügt. Der Link Thread Info zeigt die Threads an, die derzeit auf dem Anwendungsserver, der diese Laufzeitinstanz mit den folgenden Informationen hostet, ausgeführt werden.

- ID: Thread ID.
- Threadname: Name dieses Threads.
- Aktiv: Gibt an, ob dieser Thread aktiv ist.
- Dämon: Gibt an, ob dieser Thread ein Dämon-Thread ist.
- CPU Zeit: Die gesamte von diesem Thread verbrauchte CPU Zeit in Millisekunden.
- Benutzerzeit: Die gesamte von diesem Thread verbrauchte Benutzer-CPU Zeit in Millisekunden.
- Wartezeit: Die Gesamtzeit in Millisekunden, die dieser Thread im Wartezustand verbracht hat.
- Wartezahl: Die Häufigkeit, mit der dieser Thread in den Wartezustand versetzt wird.
- Sperrzeit: Die Gesamtzeit in Millisekunden, die dieser Thread im gesperrten Zustand ist.
- Sperrzahl: Die Häufigkeit, mit der dieser Thread in den gesperrten Zustand versetzt wird.
- Status: Der aktuelle Status dieses Threads.

- Wartesperre: Die Sperre, auf die dieser Thread wartet. Es ist leer, wenn es nicht auf eine Sperre wartet.
- Gehaltene Monitor: Der Monitor (Sperre), den dieser Thread derzeit hält.
- Stack Trace: Der aktuelle Stack Trace dieses Threads. Standardmäßig wird nur der oberste Eintrag angezeigt. Aber wenn Sie darauf klicken, klappt es auf, um den gesamten Stack anzuzeigen.

Kapitel 8. Klassen und Methoden für die Java-, SOAP- und REST-API von Unica Interact

Die folgenden Abschnitte listen Anforderungen und andere Details auf, die Sie kennen sollten, bevor Sie mit der Unica Interact-API zu arbeiten beginnen.

Anmerkung: Dieser Abschnitt setzt voraus, dass Sie mit Ihrem Touchpoint, der Programmiersprache Java™ und der Arbeit mit einer Java-basierten API vertraut sind.

Die Unica Interact-API hat einen Java™-Clientadapter, der Java™-Serialisierung über HTTP verwendet. Zusätzlich stellt Unica Interact eine WSDL bereit, um SOAP-Clients zu unterstützen. Die WSDL stellt denselben Satz von Funktionen wie der Java™-Clientadapter bereit, daher sind die folgenden Abschnitte, abgesehen von den Beispielen, ebenfalls zutreffend.

Anmerkung: Die mehrmalige Verwendung eines Parameters in einem einzelnen API-Aufruf wird nicht unterstützt.

Unica Interact API-Klassen

Die Unica Interact-API basiert auf der Klasse `InteractAPI`.

Es gibt 6 unterstützende Benutzeroberflächen.

- AdvisoryMessage `(nützlicher Hinweis)`
- `BatchResponse`
- `NameValuePair`
- `Angebot`
- `OfferList`
- `Antwort`

Diese Benutzeroberflächen haben 3 unterstützende konkrete Klassen. Die folgenden zwei konkreten Klassen müssen instanziiert und als Argumente in die Unica Interact-API-Methoden übergeben werden.

- NameValuePairImpl
- CommandImpl

Eine dritte konkrete Klasse namens `AdvisoryMessageCode` ist verfügbar, um die Konstanten bereitzustellen, um ggf. die vom Server zurückgegebenen Nachrichtencodes zu unterscheiden.

Der Rest dieses Abschnitts beschreibt die Methoden, aus denen sich die Unica Interact-API zusammensetzt.

Methoden zur Übergabe der Authentifizierungsparameter, wenn die API-Authentifizierung vor API-Aufrufen aktiviert ist

Wenn die API-Authentifizierung aktiviert ist, können Sie die folgenden Methoden verwenden, um die Authentifizierungsparameter wie z. B. Anmeldeinformationen oder Token vor jedem API-Aufruf zu übergeben.

`setAuthenticationParameter`

Diese Methode setzt den Authentifizierungsparameter auf den angeforderten API-Aufruf.

`setAuthenticationParameter( Zeichenfolgebenutzername und Zeichenfolgekennwort)`

- Benutzername: Unica Platform Benutzername
- Kennwort

`setAuthenticationParameter(Zeichenfolgetoken)`

Token: Token vom Server bezogen

Java™ Anordnung über HTTP-Voraussetzungen

Die Java™-Clientadapter, der Java™-Serialisierung über HTTP verwendet. -

Die Voraussetzungen für die Verwendung des Java™-Clientadapters für die Java™-Serialisierung über HTTP sind:

1. Fügen Sie Ihrem `CLASSPATH` folgende Datei hinzu:

   `Unica Interact_Home/lib/interact_client.jar`
2. Alle Objekte, die zwischen dem Client und dem Server übergeben werden, befinden sich im Paket `com.unicacorp.interact.api`. Ausführliche Informationen finden Sie im Unica Interact-API-Javadoc, das auf dem Laufzeitserver unter `Unica Interact_Home/docs/apiJavaDoc` installiert ist. Sie können das Javadoc anzeigen, indem Sie die Datei `index.html` in dieser Position mit einem Web-Browser öffnen.
3. Um eine Instanz der `InteractAPI`-Klasse zu erhalten, rufen Sie die statische Methode `getInstance` mit der URL des Unica Interact-Laufzeitservers auf.

SOAP-Voraussetzungen

Bevor Sie auf den Laufzeitserver mit SOAP zugreifen, müssen Sie zur Konfiguration Ihrer Umgebung einige Aufgaben ausführen, damit die Voraussetzungen erfüllt sind.

Wichtig: Leistungstests zeigen, dass der Java™-Serialisierungsadapter mit viel größerer Geschwindigkeit als ein generierter SOAP-Client ausführt. Aus Leistungsgründen sollten Sie wann immer möglich den Java™-Serialisierungsadapter verwenden.

Um auf den Laufzeitserver mithilfe von SOAP zuzugreifen, müssen Sie wie folgt vorgehen:

1. Konvertieren Sie die Unica Interact-API-WSDL mit dem SOAP-Toolkit Ihrer Wahl.

   Die Unica Interact-API-WSDL ist mit Unica Interact im Verzeichnis `Interact/conf` installiert.

   Wenn Sie SOAP unter Verwendung der WSDL-XML-Dateien konfigurieren, müssen Sie Ihre URLs in den Hostnamen und Port des Laufzeitservers ändern.

   Sie finden den Text der WSDL am Ende des Unica Interact-Administrationshandbuchs.

2. Installieren und konfigurieren Sie den Laufzeitserver.

   Der Laufzeitserver muss aktiv sein, um Ihre Integration umfassend testen zu können.
3. Prüfen Sie, ob Sie die richtige SOAP-Version verwenden.

   Unica Interact verwendet axis2 1.3 als die SOAP-Infrastruktur auf den Unica Interact-Laufzeitservern. Informationen darüber, welche Versionen von SOAP axis2 1.3 unterstützt, finden Sie auf der folgenden Website:

   Apache Axis2

   Unica Interact wurde mit den axis2-, XFire-, JAX-WS-Ri-, DotNet-, SOAPUI- und IBM® RAD SOAP-Clients getestet.

Voraussetzungen für REST

Eine Methode des Aufrufs der Unica Interact-API besteht darin, Aufrufe im JSON-Format (JavaScript™ Object Notation) über HTTP zu verwenden. Dies wird im vorliegenden Dokument als REST-API bezeichnet. Die REST-API bietet gegenüber SOAP eine höhere Leistung, obwohl der Java™-Serialisierungsadapter immer noch die schnellste Methode für Unica Interact-API-Aufrufe ist.

Bevor Sie die REST-API verwenden, müssen Sie Folgendes bedenken:

- Die URL, die REST-Aufrufe an die Unica Interact-API unterstützt, lautet:

  `http://Unica Interact_Runtime_Server:PORT/interact/servlet/RestServlet`, der den tatsächlichen Hostnamen oder die IP-Adresse des Unica Interact-Laufzeitservers und den Port, auf dem Unica Interact bereitgestellt wird, ersetzt.
- Es gibt zwei Unica Interact-Klassen, die spezifisch für die REST-API sind: `RestClientConnector`, die als Helfer für die Verbindung zu einer Unica Interact-Laufzeitinstanz über REST mit dem Format von JSON dient, und `RestFieldConstants`, die das zugrunde liegende Format der JSON-Nachricht beschreibt, die für API-Anfragen und -Antworten verwendet wird.
- Ein REST-Beispielclient ist unter `Unica Interact _Home/samples/javaApi/InteractRestClient.java` verfügbar. Auch wenn es sich nur um einen einfachen

Beispielcode handelt, sollte er ein geeigneter Einstieg sein, um die Verwendung der REST-API zu demonstrieren.

- Eine vollständige Beschreibung der REST-API-Klassen zusammen mit allen anderen Informationen zur Unica Interact-API finden Sie im auf dem Laufzeitserver installierten JavaDoc unter `Unica Interact_Home/docs/apiJavaDoc`.
- Die REST-API gibt Sitzungs-IDs und Nachrichten im HTML-Format mit Escapezeichen und nicht im Unicode-Format zurück.
- Wenn die API-Authentifizierung aktiviert ist, müssen die Anmeldeinformationen oder das Token im Anforderungsheader übergeben werden.
    - Eingabe-Headerparameter
        - Anmeldedaten
            - m_user_name
            - Headerparameter - Platform-Benutzername
            - m_user_password
            - Headerparameter - Platform-Benutzerkennwort
        - Token
            - m_tokenId
            - Headerparameter - token
    - Ausgabe-Headerparameter
        - m_tokenId
        - Headerparameter - token

Zusätzlich zu den hier genannten Informationen unterstützt die REST-API alle Methoden, die von den anderen Protokollen zur Verwendung der Unica Interact-API unterstützt werden.

API JavaDoc

Zusätzlich zu dem Unica Interact-Administratorhandbuch ist das Javadoc für die Unica Interact-API mit dem Laufzeitserver installiert. Das Javadoc ist für Ihre Referenz im Verzeichnis `Unica Interact_Home/docs/apiJavaDoc` installiert.

API-Beispiele

Alle Beispiele in diesem Handbuch wurden mithilfe der Java™-Serialisierung über HTTP-Adapter erstellt. Die von der WSDL generierten Klassen können je nach SOAP-Toolkit und den von Ihnen ausgewählten Optionen variieren. Wenn Sie SOAP verwenden, funktionieren diese Beispiele in Ihrer Umgebung möglicherweise nicht auf genau dieselbe Weise.

Arbeiten mit Sitzungsdaten

Wenn Sie eine Sitzung mit der Methode `startSession` initialisieren, werden Sitzungsdaten in den Speicher geladen. Während der Sitzung können Sie die Sitzungsdaten (die eine Obermenge der statischen Profildaten sind) lesen und schreiben.

Die Sitzung enthält die folgenden Daten:

- Statische Profildaten
- Segmentzuordnungen
- Echtzeitdaten
- Angebotsempfehlungen

Alle Sitzungsdaten sind bis zum Aufruf der Methode `endSession` bzw. bis zum Ablauf der `sessionTimeout`-Zeit verfügbar. Mit dem Ende der Sitzung gehen alle Daten verloren, die nicht ausdrücklich in den Kontakt- oder Antwortverlauf oder eine andere Datenbanktabelle gespeichert werden.

Die Daten werden als ein Satz von Name/Wert-Paaren gespeichert. Wenn die Daten aus der Datenbanktabelle gelesen werden, ist der Name die Spalte der Tabelle.

Sie können diese Name/Wert-Paare während der Arbeit mit der Unica Interact-API erstellen. Sie müssen nicht alle Name/Wert-Paare in einem Globalbereich deklarieren. Wenn Sie neue Ereignisparameter als Name/Wert-Paare festlegen, fügt die Laufzeitumgebung die Name/Wert-Paare den Sitzungsdaten hinzu. Wenn Sie beispielsweise Ereignisparameter mit der Methode `postEvent` verwenden, fügt die Laufzeitumgebung die Ereignisparameter den Sitzungsdaten hinzu, selbst wenn die Ereignisparameter nicht in den Profildaten verfügbar waren. Diese Daten existieren nur in den Sitzungsdaten.

Sie können Sitzungsdaten jederzeit überschreiben. Beispiel: Wenn ein Abschnitt des Kundenprofils `creditScore` umfasst, können Sie einen Ereignisparameter mithilfe des benutzerdefinierten Typs `NameValuePair` übergeben. In der Klasse `NameValuePair` können Sie die Methoden `setName` und `setValueAsNumeric` verwenden, um den Wert zu ändern. Der Name muss übereinstimmen. Innerhalb der Sitzungsdaten muss beim Namen die Groß-/Kleinschreibung nicht berücksichtigt zu werden. Daher würden die Namen `creditscore` oder `CrEdItScOrE` jeweils `creditScore` überschreiben.

Nur die letzten in die Sitzungsdaten geschriebenen Daten werden aufbewahrt. Beispiel: `startSession` lädt die Profildaten für den Wert `lastOffer`. Eine Methode `postEvent` überschreibt `lastOffer`. Dann überschreibt eine zweite Methode `postEvent lastOffer`. Die Laufzeitumgebung bewahrt nur die Daten, die von der zweiten Methode `postEvent` geschrieben wurden, in den Sitzungsdaten.

Wenn die Sitzung endet, gehen die Daten verloren, außer Sie haben besondere Vorkehrungen getroffen, wie z. B. die Verwendung eines Prozesses "Momentaufnahme" in Ihrem interaktiven Ablaufdiagramm, um die Daten in eine Datenbanktabelle zu schreiben. Wenn Sie vorhaben, Prozesse "Momentaufnahme" zu verwenden, achten Sie darauf, dass die Namen den Einschränkungen Ihrer Datenbank entsprechen müssen. Wenn Sie beispielsweise nur 256 Zeichen für den Namen einer Spalte zulassen, sollte der Name für das Name-Wert-Paar 256 Zeichen nicht überschreiten.

Informationen zur Klasse InteractAPI

Die Klasse InteractAPI enthält die Methoden, die Sie verwenden, um Ihren Touchpoint in den Laufzeitserver zu integrieren. Alle anderen Klassen und Methoden in der Unica Interact-API unterstützen die Methoden in dieser Klasse.

Sie müssen Ihre Implementierung gegen `interact_client.jar` kompilieren, das sich im Verzeichnis `lib` Ihrer Unica Interact-Laufzeitumgebungsinstallation befindet. `interact_client.jar` hängt von `log4j-api`, `log4j-core`, `commons-lang`, `commons-lang3` und `commons-httpclient` ab. Diese Abhängigkeiten müssen manuell installiert werden, und ihre Speicherorte müssen manuell in den Klassenpfad der Client-Anwendung, die `interact_client.jar` verwendet, eingefügt werden.

endSession

Die `endSession`-Methode markiert das Ende der Laufzeitsitzung. Wenn der Laufzeitserver diese Methode empfängt, wird der Verlauf protokolliert und der Speicher gelöscht.

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

- **sessionID** - Eindeutige Zeichenfolge zur Identifizierung der Sitzung.
- **parameters** - NameValuePair-Objekte, die alle Parameter identifizieren, die mit der API-Anforderung übergeben werden müssen.#

Zeitlimitüberschreitung der Laufzeitsitzungen, wenn die endSession-Methode nicht aufgerufen wird. Das Zeitlimitintervall ist mit der `sessionTimeout`-Eigenschaft konfigurierbar.

Rückgabewert

Der Laufzeitserver beantwortet die `endSession`-Methode mit dem `Response`-Objekt, das die folgenden Attribute enthält:

- SessionID
- ApiVersion
- StatusCode
- AdvisoryMessages

Beispiel

Das folgende Beispiel zeigt die `endSession`-Methode und wie Sie die Antwort parsen können. `sessionId` ist die selbe Zeichenfolge zur Identifizierung der Sitzung, die vom `startSession`-Aufruf verwendet wird, der diese Sitzung gestartet hat.

```
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

Mit der `executeBatch`-Methode können Sie mehrere Methoden mit einer einzelnen Anfrage an den Laufzeitserver ausführen.

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

- **sessionID** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID. Diese Sitzungs-ID wird für alle Befehle verwendet, die dieser Methodenaufruf ausführt.
- **commandImpl[]** - Ein Array aus CommandImpl-Objekten, jeweils eines für jeden Befehl, der ausgeführt werden soll.

Durch den Aufruf dieser Methode wird das gleiche Ergebnis erzielt wie durch den expliziten Aufruf jeder einzelnen Methode im Befehl-Array. Diese Methode minimiert die Anzahl der tatsächlichen Anfragen an den Laufzeitserver. Der Laufzeitserver führt jede Methode seriell aus. Für jeden Aufruf werden alle Fehler oder Warnungen im entsprechenden

Response-Objekt für diesen Methodenaufruf aufgezeichnet. Wird ein Fehler gefunden, wird executeBatch mit den verbliebenen Aufrufen im Stapel fortgesetzt. Wenn der Aufruf einer beliebigen Methode in einem Fehler resultiert, wird dieser Fehler im Status auf der höchsten Ebene für das BatchResponse-Objekt angezeigt. Wenn keine Fehler aufgetreten sind, werden im Status auf der höchsten Ebene alle aufgetretenen Warnungen angezeigt. Wenn keine Warnungen aufgetreten sind, wird im Status auf der höchsten Ebene die erfolgreiche Ausführung des Stapels angezeigt.

Rückgabewert

Der Laufzeitserver beantwortet den `executeBatch` mit einem BatchResponse-Objekt.

Beispiel

Das folgende Beispiel zeigt, wie Sie mit einem einzigen `executeBatch`-Aufruf alle getOffer- und postEvent-Methoden aufrufen und danach die Antwort bearbeiten können.

```
/** 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());
  }
}
```

executeBatch()-XML-Anforderungen für Interact-SOAP-API schreiben

Anhand der folgenden Schritte können Sie XML-Anforderungen des Typs `executeBatch()` für die Unica Interact-SOAP-API schreiben.

Die Anforderungs-XML für SOAP-API-Aufrufe, die aus einer einzelnen Operation bestehen (`startSession, getOffers, setAudience, endSession` etc.), darf nicht direkt in einen `executeBatch()`-Aufruf kopiert oder eingefügt werden, der aus mehreren Operationen besteht. Die Unterbefehle in Aufrufen des Typs `executeBatch()` weisen leicht abweichende WSDL- und XML-Anforderungsstrukturen im Vergleich zu API-Aufrufen auf, die aus einer einzelnen Operation bestehen. Die strukturellen Unterschiede verursachen Fehlerantworten vom Server, falls die XML-Elemente aus API-Einzeloperationsanforderungen kopiert und in `executeBatch`-Mehrfachoperationsanforderungen eingefügt werden.

Beispielfehlerantworten:

```
** 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
```

Anhand der folgenden Schritte können Sie eine XML-Anforderung des Typs `executeBatch()` schreiben. Sie können während der Ausführung dieser Schritte auf Parameterwerte von API-Aufruffanforderungen aus einzelnen Operationen verweisen, dürfen aber keine XML-Elemente kopieren und einfügen.

1. Erstellen Sie mithilfe eines WSDL-Verarbeitungstools (zum Beispiel SoapUI) aus der Unica Interact-WSDL-Datei eine korrekt formatierte XML-Anforderung des Typs `executeBatch()`.
2. Fügen Sie nach der WSDL-Definition für untergeordnete `executeBatch()`-Elemente Unterbefehle zur Anforderung hinzu.
3. Vervollständigen Sie nach der WSDL-Definition für untergeordnete `executeBatch()`-Elemente die Argumente der Unterbefehle.

getInstance

Die `getInstance`-Methode erstellt eine Instanz des Unica Interact-APIs, das mit dem angegebenen Laufzeitserver kommuniziert.

```
getInstance(String URL)
```

Wichtig: Jede Anwendung, die Sie mit diesem `-API schreiben, muss` getInstanceUnica Interact aufrufen, um ein InteractAPI-Objekt zu instanziieren, das einem Laufzeitserver zugeordnet wird, der im URL-Parameter angegeben ist.

Wenn Sie eine Lastausgleichsfunktion für Servergruppen verwenden, können Sie den Hostnamen und den Port konfigurieren, indem Sie die Lastausgleichsfunktion verwenden. Wenn Sie keine Lastausgleichsfunktion verwenden, müssen Sie eine Logik einschließen, um turnusmäßig zwischen den verfügbaren Laufzeitservern zu wechseln.

Diese Methode eignet sich nur für die Java™-Serialisierung über HTTP-Adapter. In der WSDL (Web Services Description Language) für SOAP ist keine entsprechende Methode definiert. Jede SOAP-Clientimplementierung verfügt über eine eigene Methode zum Aufbau der Endpunkt-URL.

- **URL** - Eine Zeichenfolge, die die URL für die Laufzeitinstanz angibt. Beispiel: `http://localhost:7001/Interact/servlet/InteractJSService`.

Rückgabewert

Der Laufzeitserver gibt das InteractAPI zurück.

Beispiel

Das folgende Beispiel zeigt, wie Sie ein InteractAPI-Objekt instanziieren, das auf eine Laufzeitserverinstanz verweist, die auf der gleichen Maschine ausgeführt wird wie der Touchpoint.

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

getOffers

Mit der `getOffers`-Methode können Sie Angebote vom Laufzeitserver anfordern.

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

- **sessionID** - Eine Zeichenfolge zur Identifizierung der Sitzung.
- **interactionPoint** - eine Zeichenfolge, die den Namen des Interaktionspunkts angibt, auf den diese Methode verweist.

  Anmerkung: Dieser Name muss exakt mit dem Namen des im interaktiven Kanal definierten Interaktionspunkts übereinstimmen.

- **numberOfOffers** - eine Ganzzahl, die die Anzahl der angeforderten Angebote angibt.
- **parameters** - NameValuePair-Objekte, die alle Parameter identifizieren, die mit der API-Anforderung übergeben werden müssen.#

Bevor die `getOffers`-Methode ausgeführt wird, wartet sie so viele Millisekunden, wie in der `segmentationMaxWaitTimeInMS`-Eigenschaft angegeben sind, um die erneute Segmentierung abzuschließen. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `postEvent`-Methode aufrufen, die sofort eine erneute Segmentierung oder eine `setAudience`-Methode auslöst, bevor ein `getOffers`-Aufruf erfolgt.

Rückgabewert

Der Laufzeitserver beantwortet `getOffers` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode
- NameValuePair

Beispiel

Dieses Beispiel zeigt, wie Sie ein einzelnes Angebot für den Interaktionspunkt "Overview Page Banner 1" anfordern und danach die Antwort bearbeiten können.

`sessionId` ist die gleiche Zeichenfolge zur Identifizierung, die beim Start der Laufzeitsitzung mit dem `startSession`-Aufruf verwendet wurde.

```
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());
```

Dezimalstellen in Angebotsbewertungen werden in der getOffer-Antwort im NameValue-Paar zurückgegeben. Wenn Angebote an die anfordernden Eingangskanäle zurückgegeben

werden, verwenden die Kanäle die Bewertungen, um die Angebote zu priorisieren. Die Dezimalziffern werden nicht entfernt, sodass der Sender weiß, welches Angebot eine höhere Punktzahl hat, falls Dezimalzahlen zurückgegeben werden.

getOffersForMultipleInteractionPoints

Mit der `getOffersForMultipleInteractionPoints`-Methode können Sie Angebote vom Laufzeitserver für mehrere IPs mit Deduplizierung anfordern.

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

- **sessionID** - eine Zeichenfolge, die die aktuelle Sitzung angibt.
- **requestDetailsStr** - eine Zeichenfolge, die ein Array aus `GetOfferRequest`-Objekten angibt.
- **parameters** - NameValuePair-Objekte, die alle Parameter identifizieren, die mit der API-Anforderung übergeben werden müssen.#

  Jedes `GetOfferRequest`-Objekt legt fest:

  - **ipName** - Der Name des Interaktionspunkts (IP), für den das Objekt Angebote anfordert
  - **numberRequested** - Die Anzahl an eindeutigen Angeboten, die für den angegebenen IP erforderlich ist
  - **offerAttributes** - Anforderungen an die Attribute der gelieferten Angebote mit einer Instanz von `OfferAttributeRequirements`
  - **duplicationPolicy** - Duplizierungsrichtlinien-ID für die Angebote, die geliefert werden

    Duplizierungsrichtlinien bestimmen, ob doppelte Angebote an verschiedenen Interaktionspunkten in einem einzigen Methodenaufruf zurückgegeben werden. (Innerhalb eines einzelnen Interaktionspunktes werden doppelte Angebote niemals zurückgegeben). Derzeit werden zwei Duplizierungsrichtlinien unterstützt.

- NO_DUPLICATION (ID-Wert = 1). Keines der Angebote, die in den vorangegangenen `GetOfferRequest`-Instanzen enthalten waren, wird in diese `GetOfferRequest`-Instanz einbezogen (das heißt, Unica Interact wendet die Deduplizierung an).
- ALLOW_DUPLICATION (ID-Wert = 2). Alle Angebote, die die Voraussetzungen erfüllen, die in dieser `GetOfferRequest`-Instanz angegeben sind, werden einbezogen. Es findet kein Abgleich der Angebote statt, die in den vorangegangenen `GetOfferRequest`-Instanzen enthalten waren.

Die Reihenfolge der Anfragen im Array-Parameter ist auch die Reihenfolge der Priorität, in der die Angebote geliefert werden.

Beispiel: Angenommen, die IPs in der Anfrage heißen IP1 und IP2, duplizierte Angebote sind unzulässig (mit der Duplizierungsrichtlinien-ID = 1) und jeder IP fordert zwei Angebote an. Wenn Unica Interact die Angebote A, B und C für IP1 und die Angebote A und D für IP2 findet, enthält die Antwort die Angebote A und B für IP1 und nur das Angebot D für IP2.

Zusätzlicher Hinweis: Wenn die Duplizierungsrichtlinien-ID 1 lautet, werden Angebote, die über einen IP mit hoher Priorität geliefert wurden, nicht über diesen IP geliefert.

Bevor die `getOffersForMultipleInteractionPoints`-Methode ausgeführt wird, wartet sie so viele Millisekunden, wie in der `segmentationMaxWaitTimeInMS`-Eigenschaft angegeben sind, um die erneute Segmentierung abzuschließen. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `postEvent`-Methode aufrufen, die sofort eine erneute Segmentierung oder eine `setAudience`-Methode auslöst, bevor ein `getOffers`-Aufruf erfolgt.

Rückgabewert

Der Laufzeitserver beantwortet `getOffersForMultipleInteractionPoints` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion

- Array von OfferList
- SessionID
- StatusCode

Beispiel

```
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());
 }
```

Hinweis: Die Syntax von `requestStr` lautet folgendermaßen:

`requests_for_IP[<requests_for_IP]`

Wo

```
<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
```

Im Beispiel oben bedeutet `requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))})` für den Interaktionspunkt IP1 eine Lieferung von 5 möglichst verschiedenen Angeboten, die während dieses Methodenaufrufs nicht auch von einem anderen Interaktionspunkt zurückgegeben werden können. Alle 5 Angebote müssen über das numerische Attribut `attr1` mit dem Wert 1 und über das Zeichenfolgeattribut `attr2`mit dem Wert value2 verfügen. Von diesen 5 Angeboten dürfen maximal 3 über das Zeichenfolgeattribut `attr3` mit dem Wert value3 und maximal 3 über das numerische Attribut `attr4` mit dem Wert 4 verfügen.

Die zulässigen Attributtypen sind numerisch, Zeichenfolge und Datum/Uhrzeit und der Wert des Datum/Uhrzeit-Attributs muss dem Format `MM/dd/yyyy HH:mm:ss`entsprechen. Zum Abrufen der zurückgegebenen Angebote verwenden Sie die Methode `Response.getAllOfferLists()`. Zum besseren Verständnis der Syntax wird im Beispiel in `setGetOfferRequests` die gleiche Instanz von `GetOfferRequests` bevorzugt mit Java™-Objekten erstellt.

getProfile

Mit der `getProfile`-Methode können Sie Profildaten und temporäre Informationen über die Besucher des Touchpoints abrufen.

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

- **sessionID** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **parameters** - NameValuePair-Objekte, die alle Parameter identifizieren, die mit der API-Anforderung übergeben werden müssen.#

Rückgabewert

Der Laufzeitserver beantwortet `getProfile` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

Beispiel

Das folgende Beispiel zeigt, wie Sie `getProfile` verwenden und danach die Antwort bearbeiten können.

`sessionId` ist die gleiche Zeichenfolge zur Identifizierung, die beim Start der Sitzung mit dem `startSession`-Aufruf verwendet wurde.

```
response = api.getProfile(sessionId);
/**  Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
```

```
       System.out.println("getProfile call processed with no warnings or
errors");
       // Print the profile - it's just an array of NameValuePair objects
       for(NameValuePair nvp : response.getProfileRecord())
       {
           System.out.println("Name:"+nvp.getName());

if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
           {
               System.out.println("Value:"+nvp.getValueAsDate());
           }
           else
if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
           {
               System.out.println("Value:"+nvp.getValueAsNumeric());
           }
           else
           {
               System.out.println("Value:"+nvp.getValueAsString());
           }
       }
   }
   else if(response.getStatusCode() == Response.STATUS_WARNING)
   {
       System.out.println("getProfile call processed with a warning");
   }
   else
   {
       System.out.println("getProfile call processed with an error");
   }
   // For any non-successes, there should be advisory messages explaining
why
```

```
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getProfile",
   response.getAdvisoryMessages());
```

getVersion

Die `getVersion`-Methode gibt die Version der aktuellen Implementierung des Unica Interact Laufzeitservers zurück.

```
getVersion()
```

Es empfiehlt sich, diese Methode zu verwenden, wenn Sie den Touchpoint mit dem Unica Interact API initialisieren.

Rückgabewert

Der Laufzeitserver beantwortet `getVersion` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- StatusCode

Beispiel

Dieses Beispiel zeigt eine einfache Methode, wie Sie `getVersion` aufrufen und die Ergebnisse verarbeiten können.

```
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

Mit der `postEvent`-Methode können Sie jedes Ereignis ausführen, das im interaktiven Kanal definiert ist.

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

- **sessionID**: - eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **eventName**: eine Zeichenfolge zur Identifizierung des Ereignisnamens.

  Anmerkung: Der Name des Ereignisses muss mit dem im interaktiven Kanal definierten Ereignisnamen übereinstimmen. Bei diesem Namen braucht die Groß-/Kleinschreibung nicht berücksichtigt zu werden.

- **eventParameters**. `NameValuePairImpl`-Objekte, die alle Parameter identifizieren, die mit dem Ereignis übergeben werden müssen.

Wenn dieses Ereignis eine erneute Segmentierung auslöst, müssen Sie sicherstellen, dass alle vom interaktiven Ablaufdiagramm benötigten Daten in den Sitzungsdaten verfügbar sind. Wenn diese Werte noch nicht durch vorangegangene Aktionen ausgefüllt wurden (zum Beispiel durch `startSession`, durch `setAudience` oder beim Laden der Profiltabelle), müssen Sie für jeden fehlenden Wert einen eventParameter einschließen. Wenn Sie zum Beispiel alle Profiltabellen so konfiguriert haben, dass diese im Hauptspeicher geladen werden, müssen Sie für alle temporären Daten, die für interaktive Ablaufdiagramme erforderlich sind, jeweils ein NameValuePair einschließen.

Wenn Sie mehrere Zielgruppenebenen verwenden, haben Sie vermutlich verschiedene Sätze an eventParameters für jede Zielgruppenebene. Sie sollten daher eine entsprechende Logik einschließen, die gewährleistet, dass für jede Zielgruppenebene immer der richtige Parametersatz ausgewählt wird.

Wichtig: Wenn dieses Ereignis den Antwortverlauf protokolliert, müssen Sie den Verfahrenscode für das Angebot übergeben. Sie müssen den Namen für das NameValuePair als `UACIOfferTrackingCode` definieren.

Pro Ereignis können Sie immer nur einen Verfahrenscode übergeben. Wenn Sie den Verfahrenscode für einen Angebotskontakt nicht übergeben, protokolliert Unica Interact jeweils einen Angebotskontakt für jedes Angebot in der zuletzt empfohlenen Angebotsliste. Wenn Sie den Verfahrenscode für eine Antwort nicht übergeben, gibt Unica Interact einen Fehler zurück.

- Es gibt eine Reihe weiterer reservierter Parameter, die Sie mit `postEvent` und anderen Methoden verwenden können, die später in diesem Abschnitt erläutert werden.

Wenn Sie eine erneute Segmentierung anfordern oder in den Kontakt- oder Antwortverlauf schreiben, wird nicht auf eine Antwort gewartet.

Im Verlauf einer neuen Segmentierung werden nicht frühere Segmentierungsergebnisse für die aktuelle Zielgruppenebene bereinigt. Sie können den Parameter `UACIExecuteFlowchartByName` zum Definieren bestimmter Ablaufdiagramme verwenden, die ausgeführt werden sollen. Die `getOffers`-Methode wartet, bis die erneute

Segmentierung abgeschlossen ist, und wird erst danach ausgeführt. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `postEvent`-Methode aufrufen, die sofort eine erneute Segmentierung auslöst, bevor ein `getOffers`-Aufruf erfolgt.

Rückgabewert

Der Laufzeitserver beantwortet `postEvent` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Beispiel

Das folgende `postEvent`-Beispiel zeigt, wie Sie neue Parameter für ein Ereignis, das eine erneute Segmentierung auslöst, senden und danach die Antwort bearbeiten können.

`sessionId` ist die gleiche Zeichenfolge zur Identifizierung, die beim Start der Sitzung mit dem `startSession`-Aufruf verwendet wurde.

```
String eventName = "SearchExecution";

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

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

NameValuePair parmB3 = new NameValuePairImpl();
parmB3.setName("Browser");
```

```
parmB3.setValueAsString("IE6");
parmB3.setValueDataType(NameValuePair.DATA_TYPE_STRING);

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

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

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

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

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

/** Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
```

```
   {
       System.out.println("postEvent call processed with no warnings or
errors");
   }
   else if(response.getStatusCode() == Response.STATUS_WARNING)
   {
       System.out.println("postEvent call processed with a warning");
   }
   else
   {
       System.out.println("postEvent call processed with an error");
   }

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

setAudience

Mit der `setAudience`-Methode können Sie die Zielgruppen-ID und die Zielgruppenebene für Besucher festlegen.

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

- **sessionID** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **audienceID** - ein Array von `NameValuePairImpl`-Objekten zum Definieren der Zielgruppen-ID.
- **audienceLevel** - eine Zeichenfolge zum Definieren der Zielgruppenebene.

- **parameters** - `NameValuePairImpl`-Objekte zum Identifizieren aller Parameter, die mit `setAudience` übergeben werden müssen. Diese Werte werden in den Sitzungsdaten gespeichert und können zur Segmentierung verwendet werden.

  Sie benötigen für jede Spalte in Ihrem Profil einen Wert. Dies ist eine Obermenge aus allen Spalten in den Echtzeitdaten und in allen Tabellen, die für den interaktiven Kanal definiert sind. Wenn Sie bereits alle Sitzungsdaten mit `startSession` oder `postEvent` ausgefüllt haben, ist es nicht erforderlich, neue Parameter zu senden.

Die `setAudience`-Methode löst eine erneute Segmentierung aus. Die `getOffers`-Methode wartet, bis die erneute Segmentierung abgeschlossen ist, und wird erst danach ausgeführt. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `setAudience`-Methode aufrufen, bevor ein `getOffers`-Aufruf erfolgt.

Die `setAudience`-Methode lädt auch die Profildaten für die Zielgruppen-ID. Mit der `setAudience`-Methode können Sie erzwingen, dass erneut die gleichen Profildaten geladen werden wie mit der `startSession`-Methode.

Rückgabewert

Der Laufzeitserver beantwortet `setAudience` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Beispiel

In diesem Beispiel bleibt die Zielgruppenebene gleich, aber die ID ändert sich, wie wenn sich ein anonymer Benutzer anmeldet und dann bekannt wird.

`sessionId` und `audienceLevel` sind die gleichen Zeichenfolgen, die beim Start der Sitzung mit dem `startSession`-Aufruf verwendet wurden, um die Sitzung und die Zielgruppenebene zu identifizieren.

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

NameValuePair[] newAudienceId = { custId2 };

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

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

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

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

setDebug

Mit der `setDebug`-Methode können Sie den Detaillierungsgrad der Protokollierung für alle Codepfade für die Sitzung festlegen.

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

- **sessionID** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **debug** - ein boolescher Ausdruck zum Aktivieren oder Inaktivieren von Daten zur Fehlerbehebung. Gültige Werte sind `true` oder `false`. Wenn der Wert wahr ist, protokolliert Unica Interact die Daten zur Fehlerbehebung im Protokoll des Laufzeitservers.

Rückgabewert

Der Laufzeitserver beantwortet `setDebug` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Beispiel

Im folgenden Beispiel wird die Fehlerbehebungsstufe der Sitzung geändert.

`sessionId` ist die gleiche Zeichenfolge zur Identifizierung, die beim Start der Sitzung mit dem `startSession`-Aufruf verwendet wurde.

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

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

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

startSession

Die `startSession`-Methode erstellt und definiert eine Laufzeitsitzung.

```
startSession(String sessionID,
   boolean relyOnExistingSession,
   boolean debug,
```

```
    String interactiveChannel,
    NameValuePairImpl[] audienceID,
    String audienceLevel,
    NameValuePairImpl[] parameters)
```

`startSession` kann bis zu fünf Aktionen auslösen:

- Erstellen der Laufzeitsitzung.
- Laden der Besucherprofildaten für die aktuelle Zielgruppenebene in der Laufzeitsitzung inklusive aller Dimensionstabellen, die in der für den interaktiven Kanal definierten Tabellenzuordnung zum Laden markiert sind.
- Auslösen der Segmentierung, indem alle interaktiven Ablaufdiagramme für die aktuelle Zielgruppenebene ausgeführt werden.
- Laden von Angebotsunterdrückungsdaten in der Sitzung, wenn die Eigenschaft `enableOfferSuppressionLookup` auf "true" gesetzt ist.
- Laden von Bewertungsüberschreibungsdaten in der Sitzung, wenn die Eigenschaft `enableScoreOverrideLookup` auf "true" gesetzt ist.

Die `startSession`-Methode benötigt die folgenden Parameter:

- **sessionID** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID. Sie müssen die Sitzungs-ID definieren. Sie können zum Beispiel eine Kombination aus Kunden-ID und Zeitmarke verwenden.

  Sie müssen eine Sitzungs-ID angeben, um zu definieren, was eine Laufzeitsitzung auszeichnet. Dieser Wert wird vom Client verwaltet. Alle Methodenaufrufe für die gleiche Sitzungs-ID müssen vom Client synchronisiert werden. Das Verhalten für gleichzeitige API-Aufrufe mit der gleichen Sitzungs-ID ist nicht definiert.
- **relyOnExistingSession** - ein boolescher Ausdruck, der definiert, ob diese Sitzung eine neue oder eine vorhandene Sitzung verwendet. Gültige Werte sind `true` oder `false`. Wenn der Wert `true` ist, müssen Sie eine vorhandene Sitzungs-ID mit der `startSession`-Methode angeben. Wenn der Wert `false` ist, müssen Sie eine neue Sitzungs-ID angeben.

Wenn Sie `relyOnExistingSession` auf `true` setzen und eine Sitzung vorhanden ist, verwendet die Laufzeitumgebung die vorhandenen Sitzungsdaten. Es werden keine Daten erneut geladen und es wird keine Segmentierung ausgelöst. Wenn die Sitzung nicht vorhanden ist, erstellt die Laufzeitumgebung eine neue Sitzung inklusive Laden der Daten und Auslösen der Segmentierung. Wenn die Sitzungsdauer des Touchpoints länger als die Laufzeitsitzung ist, kann es sinnvoll sein, `relyOnExistingSession` auf "true" zu setzen und mit allen `startSession`-Aufrufen zu verwenden. Beispiel: Die Sitzung einer Website ist 2 Stunden lang aktiv, während die Laufzeitsitzung nur 20 Minuten lang aktiv ist.

Wenn Sie `startSession` zweimal mit der gleichen Sitzungs-ID aufrufen, gehen alle Sitzungsdaten des ersten `startSession`-Aufrufs verloren, wenn `relyOnExistingSession` auf false gesetzt ist.

- **debug** - ein boolescher Ausdruck zum Aktivieren oder Inaktivieren von Daten zur Fehlerbehebung. Gültige Werte sind `true` oder `false`. Wenn der Wert true ist, protokolliert Unica Interact die Daten zur Fehlerbehebung in den Protokollen des Laufzeitservers. Das Debug-Flag wird für jede Sitzung individuell gesetzt. Somit können Sie die Daten zur Fehlerbehebung für einzelne Sitzungen verfolgen.
- **interactiveChannel** - eine Zeichenfolge, die den Namen des interaktiven Kanals definiert, auf den diese Sitzung verweist. Dieser Name muss exakt mit dem in Unica Campaign definierten Namen des interaktiven Kanals übereinstimmen.
- **audienceID** - ein Array aus NameValuePairImpl-Objekten, wobei die Namen mit den physischen Spaltennamen aller Tabellen übereinstimmen müssen, in denen die Zielgruppen-ID enthalten ist.
- **audienceLevel** - eine Zeichenfolge zum Definieren der Zielgruppenebene.
- **parameters** - NameValuePairImpl-Objekte zum Identifizieren aller Parameter, die mit `startSession` übergeben werden müssen. Diese Werte können zur Segmentierung verwendet werden.

  Wenn Sie mehrere interaktive Ablaufdiagramme für die gleiche Zielgruppenebene haben, müssen Sie eine Obermenge mit allen Spalten in allen Tabellen einschließen. Wenn Sie die Laufzeit so konfigurieren, dass die Profiltabelle geladen wird, und die Profiltabelle alle benötigten Spalten enthält, ist es nicht erforderlich, Parameter zu

übergeben, es sei denn, Sie möchten die Daten in der Profiltabelle überschreiben. Wenn die Profiltabelle eine Untermenge der benötigten Spalten enthält, müssen Sie die fehlenden Spalten als Parameter einschließen.

Wenn `audienceID` oder `audienceLevel` ungültig und `relyOnExistingSession` false ist, schlägt der `startSession`-Aufruf fehl. Wenn `interactiveChannel` ungültig ist, schlägt `startSession` fehl, unabhängig davon, ob `relyOnExistingSession` true oder false ist.

Wenn `relyOnExistingSession` true ist und Sie einen zweiten `startSession`-Aufruf mit der gleichen `sessionID` durchführen, nachdem die erste Sitzung bereits abgelaufen ist, erstellt Unica Interact eine neue Sitzung.

Wenn `relyOnExistingSession` true ist und Sie einen zweiten `startSession`-Aufruf mit der gleichen `sessionID`, aber mit einer anderen `audienceID` oder `audienceLevel` durchführen, ändert der Laufzeitserver die Zielgruppe für die vorhandene Sitzung.

Wenn `relyOnExistingSession` true ist und Sie einen zweiten `startSession`-Aufruf mit der gleichen `sessionID`, aber mit einem anderen `interactiveChannel` durchführen, erstellt der Laufzeitserver eine neue Sitzung.

Rückgabewert

Der Laufzeitserver beantwortet `startSession` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages (wenn StatusCode nicht 0 ist)
- ApiVersion
- SessionID
- StatusCode

Anmerkung: Aufgrund der Einschränkungen von IEEE 754-Gleitkommazahlen können nicht alle numerischen Werte, einschließlich AudienceIDs, SessionIDs und Sitzungsparameter, in Interact exakt dargestellt werden, auch wenn sie in der Profiltabelle exakt dargestellt werden können. Darüber hinaus dürfen ganzzahlige

Werte größer als 2^53 (9007199254740992), nicht als Zielgruppen-ID-Werte verwendet werden.

Beispiel

Das folgende Beispiel zeigt eine Möglichkeit zum Aufrufen von `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` ist eine Methode, um das von `startSession` zurückgegebene Antwortobjekt zu bearbeiten.

```
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());
}
```

Angebotsdeduplizierung über Angebotsattribute

Unter Verwendung der Anwendungsprogrammierschnittstelle (Application Programming Interface, API) von Unica Interact werden mit zwei API-Aufrufen Angebote bereitgestellt: `getOffers` und `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` kann die Rückgabe duplizierter Angebote auf der Ebene OfferID verhindern, kann jedoch keine Angebote über eine Angebotskategorie deduplizieren. Daher war z. B. vorher bei Unica Interact für eine Rückgabe von nur einem Angebot aus den einzelnen Angebotskategorien eine Problemumgehung

erforderlich. Durch die Einführung von zwei Parametern im API-Aufruf `startSession` sind Angebotsdeduplizierungen über Angebotsattribute, wie z. B. die Kategorie, jetzt möglich.

In dieser Liste finden Sie eine Übersicht über die Parameter, die dem API-Aufruf `startSession` hinzugefügt wurden. Sie finden weitere Informationen zu diesen Parametern oder zu sonstigen Aspekten der Unica Interact-API im Unica Interact-Administratorhandbuch sowie in den Javadoc-Dateien, die in Ihrer Unica Interact-Installation enthalten sind. Sie finden diese Dateien im Pfad `<Unica Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Wenn Sie einen `startSession`-API-Aufruf mit Angebotsdeduplizierung erstellen möchten, damit nachfolgende `getOffer`-Aufrufe immer nur jeweils ein Angebot aus jeder Kategorie zurückgeben, müssen Sie den Parameter `UACIOfferDedupeAttribute` als Bestandteil des API-Aufrufs einschließen. Sie können einen Parameter wie folgt im Format `name,value,type` angeben:

  `UACIOfferDedupeAttribute,<attributeName>,string`

  In diesem Beispiel würden Sie <attributeName> durch den Namen des Angebotsattributs ersetzen, das Sie als Kriterium für die Deduplizierung verwenden möchten (zum Beispiel "Category" (Kategorie)).

  Anmerkung: Unica Interact überprüft die Angebote, die den von Ihnen angegebenen Attributwert (zum Beispiel "Category") aufweisen. Anschließend wird eine Deduplizierung durchgeführt, damit mit Ausnahme des Angebots, das die höchste Bewertung aufweist, alle Angebote entfernt werden. Wenn die Angebote mit einem Attributduplikat auch identische Bewertungen aufweisen, trifft Unica Interact eine Zufallsauswahl unter den übereinstimmenden Angeboten und gibt diese zurück.

- `UACINoAttributeDedupeIfFewerOf`. Wenn Sie den Parameter `UACIOfferDedupeAttribute` im `startSession`-Aufruf einschließen, können Sie auch den Parameter `UACINoAttributeDedupeIfFewerOf` festlegen. Damit geben Sie an, wie sich das Programm verhalten soll, wenn die Angebotsliste nach der Deduplizierung

nicht genügend Angebote enthält, damit die ursprüngliche Anforderung erfüllt werden kann.

Wenn Sie beispielsweise für `UACIOfferDedupeAttribute` die Verwendung der Angebotskategorie zur Deduplizierung von Angeboten festlegen und Ihr nachfolgender getOffers-Aufruf die Rückgabe von acht Angeboten anfordert, kann es sein, dass aufgrund der Deduplizierung weniger als acht auswählbare Angebote verfügbar sind. Wenn Sie den Parameter `UACINoAttributeDedupeIfFewerOf` auf "true" setzen, werden in diesem Fall einige der deduplizierten Angebote zur Kandidatenliste hinzugefügt, damit die angeforderte Anzahl der Angebote erfüllt wird. Wenn Sie in diesem Beispiel den Parameter auf "false" setzen, wird die angeforderte Anzahl nicht durch die zurückgegebene Anzahl an Angeboten erreicht.

`UACINoAttributeDedupeIfFewerOf` Die Standardeinstellung ist "true".

Angenommen, Sie haben wie folgt als Parameter für `startSession` die Angebotskategorie (Category) als Deduplizierungskriterium angegeben:

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

Diese Parameterkombination bewirkt, dass Unica Interact Angebote auf Basis des Angebotsattributs "Category" dedupliziert und selbst dann nur die deduplizierten Angebote zurückgibt, wenn die resultierende Anzahl der Angebote die angeforderte Anzahl unterschreitet (da der Parameter `UACINoAttributeDedupeIfFewerOffer` auf "false" gesetzt ist).

Wenn Sie einen `getOffers`-API-Aufruf ausgeben, könnte die ursprüngliche Gruppe der auswählbaren Angebote die folgenden Angebote enthalten:

- Category=Electronics: Angebot A1 mit einer Bewertung von 100 und Angebot A2 mit einer Bewertung von 50.
- Category=Smartphones: Angebot B1 mit einer Bewertung von 100, Angebot B2 mit einer Bewertung von 80 und Angebot B3 mit einer Bewertung von 50.
- Category=MP3Players: Angebot C1 mit einer Bewertung von 100, Angebot C2 mit einer Bewertung von 50.

In diesem Fall gab es zwei doppelte Angebote, die mit der ersten Kategorie übereinstimmen, drei doppelte Angebote, die mit der zweiten Kategorie übereinstimmen, und zwei doppelte Angebote, die mit der dritten Kategorie übereinstimmen. Als Angebote werden die Angebote mit der höchsten Bewertung aus jeder Kategorie zurückgegeben, also Angebot A1, Angebot B1 und Angebot C1.

Obwohl der `getOffers`-API-Aufruf sechs Angebote angefordert hat, werden nur drei Angebote zurückgegeben, da der Parameter `UACINoAttributeDedupeIfFewerOffer` in diesem Beispiel auf "false" gesetzt ist.

Wenn der `getOffers`-API-Aufruf sechs Angebote angefordert hat und in diesem Beispiel keine Angabe für den Parameter `UACINoAttributeDedupeIfFewerOffer` gemacht oder der Parameter explizit auf "true" gesetzt wurde, werden einige der doppelten Angebote in das Ergebnis aufgenommen, damit die angeforderte Anzahl erfüllt wird.

Reservierte Parameter

Mit dem Unica Interact API werden mehrere reservierte Parameter verwendet. Einige werden für den Laufzeitserver benötigt und andere können für zusätzliche Funktionen verwendet werden.

API-Parameter

Feature	Parameter	Beschreibung
In benutzerdefinierter Tabelle protokollieren (Scope-Sitzung)	`UACICustomLoggerTableName`	Der Name einer Tabelle in der Datenquelle der Laufzeittabellen. Wenn Sie diesen Parameter mit einem gültigen Tabellennamen angeben, schreibt die Laufzeitumgebung alle Sitzungsdaten in die ausgewählte Tabelle. In der Tabelle werden alle Spaltennamen ausgefüllt, die mit den NameValuePair-Sitzungsdaten übereinstimmen. Die Laufzeitumgebung schreibt eine Null in jede Spalte, die nicht mit einem Name/Wert-Paar der Sitzung übereinstimmt. Mit den Konfigurationseigen-

Feature	Parameter	Beschreibung
		schaften `customLogger` können Sie den Prozess verwalten, der in die Datenbank schreibt.
In einer bestimmten Datei protokollieren (Scope-Sitzung)	`UACILogSeparationFileName`	Benutzerdefinierter Name der Protokolldatei: Mit diesem Parameter kann die Laufzeitumgebung alle zu dieser Sitzung gehörenden Protokolle in die angegebene Protokolldatei schreiben. Die Datei mit dem angegebenen Namen wird am Standardprotokollort erstellt, falls sie nicht verfügbar ist. Für den Simulator wird der Parameter automatisch hinzugefügt. Das Namensformat der Protokolldatei ist <`Simulator-Basic-{scenarioName}.log`>, <`Simulator-Advanced-{scenarioName}.log`>, und <`Simulator-Coverage-{scenarioName}.log`> bzw. für grundsätzliche, erweiterte und Abdeckungsszenarien.
Mehrere Antworttypen (Scope-Aufruf)	`UACILogToLearning`	Eine Ganzzahl mit dem Wert 1 oder 0. 1 gibt an, dass die Laufzeitumgebung das Ereignis als Annahme an das Lernsystem protokollieren oder die Angebotsunterdrückung innerhalb einer Sitzung aktivieren soll. 0 gibt an, dass die Laufzeitumgebung das Ereignis nicht im Lernsystem protokollieren oder die Angebotsunterdrückung innerhalb einer Sitzung aktivieren soll. Mit diesem Parameter können Sie mehrere `postEvent`Methoden erstellen, die verschiedene Antworttypen protokollieren, ohne

Feature	Parameter	Beschreibung
		das Lernen zu beeinflussen. Es ist nicht erforderlich, diesen Parameter für Ereignisse zu definieren, die einen Kontakt, eine Annahme oder eine Ablehnung protokollieren. Sie müssen diesen Parameter in Verbindung mit `UACIResponseTypeCode` verwenden. Falls Sie `UACILOGTOLEARNING` nicht definieren, verwendet die Laufzeitumgebung den Standardwert 0 (es sei denn, das Ereignis löst einen Protokollkontakt, eine Annahme oder eine Ablehnung aus).
	`UACIResponseTypeCode`	Ein Wert, der einen Antworttypcode darstellt. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` sein.
Antwortverfolgung (Scope-Aufruf)	`UACIOfferTrackingCode`	Der Verfahrenscode für das Angebot. Sie müssen diesen Parameter definieren, wenn das Ereignis in den Kontakt- oder Antwortverlauf protokolliert. Pro Ereignis können Sie immer nur einen Verfahrenscode übergeben. Wenn Sie den Verfahrenscode für einen Angebotskontakt nicht übergeben, protokolliert die Laufzeitumgebung jeweils einen Angebotskontakt für jedes Angebot in der zuletzt empfohlenen Angebotsliste. Wenn Sie den Verfahrenscode für eine Antwort nicht übergeben, gibt die Laufzeitumgebung einen Fehler zurück. Wenn Sie die sitzungsübergreifende Antwortverfolgung konfigurieren, können Sie mit dem Parameter `UACIOfferTrackingcodeType`

Feature	Parameter	Beschreibung
		den Verfolgungscodetyp definieren, welcher anstelle des Verfahrenscodes verwendet werden soll.
Sitzungsübergreifendes Antwortverfolgung (Scope-Aufruf)	`UACIOfferTrackingCodeType`	Eine Zahl, die den Typ des Verfolgungscodes definiert. 1 ist der Standardverfahrenscode und 2 ist der Angebotscode. Alle Codes müssen gültige Einträge in der Tabelle `UACI_TrackingType` sein. Sie können dieser Tabelle weitere und angepasste Codes hinzufügen.
Spezifische Ausführung des Ablaufdiagramm (Umfangsaufruf)	`UACIExecuteFlowchartByName`	Wenn Sie diesen Parameter für eine Methode definieren, die eine Segmentierung auslöst (`startSession`, `setAudience` oder `postEvent`, das eine erneute Segmentierung auslöst), wird nur die angegebenen Ablaufdiagramme von Unica Interact ausgeführt anstatt der Ausführung von allen Ablaufdiagrammen für die aktuelle Zielgruppenebene. Um eine Liste mit Ablaufdiagrammen anzugeben, trennen Sie diese durch eine vertikale Linie (|).
Begrenzung des Angebotsfeldes in der getOffers API bei Interact RunTime (Scope-Sitzung)	`UACIOfferFields` und `UACIExcludeOfferFields`	Sie können die Angebotsattribute `UACIOfferFields` und `UACIExcludeOfferFields` ein- bzw. ausschließen, indem Sie diese Attribute im Parameter von `startSession`, `setAudience` oder einem `postEvent` übergeben.

Feature	Parameter	Beschreibung
Ereignismuster zurücksetzen (Scope-Aufruf)	• `ResetEventPattern`. Dieser Parameter setzt die Zustände der Ereignismuster für alle Muster für die Zielgruppen-ID zurück. • `PatternToReset`. Dieser Parameter setzt die Zustände der Ereignismuster für bestimmte Ereignismuster zurück. Entweder die Muster-ID oder der Musternamen können zurückgesetzt werden. Dieser Parameter kann nur mit dem Parameter ResetEventPattern übergeben werden.	Sie können die Zustände eines bestimmten Musters oder aller Muster für eine bestimmte Zielgruppen-ID, die der Sitzung zugeordnet ist, zurücksetzen. Sie müssen das Ereignis mit einem speziellen Parameter posten. Mit dem Ereignis dürfen keine expliziten Aktionen verbunden sein.
Angebotsunterdrückung zurücksetzen (Scope-Aufruf)	• `ResetOfferSuppression`. Dieser Parameter setzt die Unterdrückungsregeln für alle Angebote für die	Sie können die Zustände einer bestimmten Angebotsunterdrückungsregel oder aller Angebotsunterdrückungsregeln für eine bestimmte Zielgruppen-ID, die der Sitzung zugeordnet ist, zurücksetzen. Sie müssen das Ereignis mit einem speziellen Parameter

Feature	Parameter	Beschreibung
	Zielgruppen-ID zurück. • `OfferToResetSuppression`. Dieser Parameter setzt die Angebotsunterdrückungsregeln für bestimmte Angebote zurück. Entweder Angebots-ID oder Angebotscode können zurückgesetzt werden. Um die Angebotsunterdrückung für Angebote mit mehrteiligem Angebotscode zurückzusetzen, muss der Angebotscode durch Kommas getrennt werden. Zum Beispiel: Angebotscode1, Angebotscode2, Angebotscode3 und so weiter. Dieser Parameter kann nur mit dem Parameter Re-	posten. Mit dem Ereignis dürfen keine expliziten Aktionen verbunden sein.

Feature	Parameter	Beschreibung
	setOfferSuppression übergeben werden.	
Sitzungsparameter vor API-Aufruf entfernen (Scope-Sitzung)	`UACIPreRemoveParameter`. Komma getrennte Namen der Parameter, die Sie vor dem API-Aufruf aus der Sitzung entfernen möchten.	Dieser Parameter ist hilfreich, wenn Sie einige Parameter entfernen möchten, die bereits durch frühere API-Aufrufe in der Sitzung festgelegt wurden. Es können mehrere Parameternamen übergeben und durch Kommas getrennt werden.
Sitzungsparameter nach API-Aufruf entfernen (Scope-Sitzung)	`UACIPostRemoveParameter`. Komma getrennte Namen der Parameter, die Sie nach Ausführung des API-Aufrufs aus der Sitzung entfernen möchten.	Dieser Parameter ist hilfreich, wenn Sie einige Parameter entfernen möchten, die für die weitere Verarbeitung nicht erforderlich sind. Es können mehrere Parameternamen übergeben und durch Kommas getrennt werden.
Synchrone Ausführung des Ablaufdiagramme (Scope-Sitzung)	`UACIWaitForSegmentation`	Wenn Sie diesen Parameter für eine Methode definieren, die die Segmentierung auslöst (startSession, setAudience oder ein postEvent, das die Neusegmentierung auslöst), werden Flussdiagramme für die aktuelle Zielgruppenebene synchron ausgeführt. Das Timeout für diese synchrone Ausführung wird durch die Platform-Konfiguration unter dem Pfad `Affinium|interact|offerserving|segmentationMaxWaitTimeInMS` definiert Beispiel: `UACIWaitForSegmentation|true|string`

Feature	Parameter	Beschreibung
Holen Sie sich die zwischengespeicherten Angebote (Scope-Aufruf)	UACICachedOffers	Wenn Sie diesen Parameter für den API-Aufruf von getOffers definieren, werden die Angebote des vorherigen Aufrufs von getOffers zurückgegeben, ohne die Arbitrierungslogik auszuführen. Wenn die vorherige Liste der Angebote leer ist, findet eine Angebotsschlichtung statt und neue Angebote werden zurückgegeben. Die Funktion zur Zwischenspeicherung von Angeboten ist aktiviert, wenn die Platform-Konfiguration unter dem Pfad `Affinium|interact|offerserving|enableOfferCaching` auf True gesetzt wird. Beispiel: `UACICachedOffers|true|string`
Angebote von zugehörigen Zielgruppen-IDs einschließen (Scope-Sitzung)	`UACISupplementAudience`	Die zusätzlichen Zielgruppen-IDs, deren qualifizierte Angebote eingeschlossen werden müssen, während Angebote für die Zielgruppen-ID der Sitzung abgerufen werden. Das Format lautet `audienceLevel1,audienceId1Field1=value1[,audienceId1FieldN=valueN]`. Dieser Parameter kann mehrfach verwendet werden, wenn mehrere Zielgruppen-IDs zur Angebotseinholung erforderlich sind.
Zugehörige Sitzungen (Scope-Aufruf)	`UACIEmbeddedSession`	Gibt an, ob dieses Batch in einer separaten Sitzung ausgeführt werden soll. 1 - Ja, 0 - Nein. Das ist nur in startSession und setAudience wirksam, wenn diese API Anfrage der erste Befehl in einem executeBatch Aufruf ist.

Feature	Parameter	Beschreibung
Zugehörige Sitzungen (Scope-Aufruf)	`UACIIncludeArbitration`	Gibt an, ob die Zusammenfassung von Angebotsentscheidung in die Antwort auf die Anfragen getOffers und `getOfersForMultipleInteractionPoints` eingeschlossen werden soll. Wenn die API Historisierung aktiviert ist, werden dieselben Informationen auch zusammen mit dieser gespeichert. 1 - Ja, 0 - Nein.
Kontaktverfolgung (Scope-Aufruf)	`UACIContactStatusCode`	Der Parameter ist eine Zeichenfolge, die einen Kontakttypcode darstellt. Der Parameterwert muss ein gültiger Eintrag in der Tabelle UA_ContactStatus sein.
Antwortverfolgung (Scope-Aufruf)	`UACIResponseTypeCode`	Ein Wert, der einen Antworttypcode darstellt. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` sein.
(Scope-Sitzung)	`UACIPurgePriorWhiteListOnLoad`	Bei dem Aufruf der Methode `setAudience` lädt dieser Parameter die Whitelist-Tabelle in einer bestehenden Sitzung erneut. Wird `UACIPurgePriorWhiteListOnLoad` auf 1 gesetzt, werden die zuvor geladenen Inhalte der Whitelist aus dieser Sitzung entfernt.
(Scope-Sitzung)	`UACIPurgePriorBlackListOnLoad`	Bei dem Aufruf der Methode `setAudience` lädt dieser Parameter die Blacklist-Tabelle in einer bestehenden Sitzung erneut. Wird `UACIPurgePriorWhiteListOnLoad` auf 1 gesetzt, werden die zuvor geladenen Inhalte der Whitelist aus dieser Sitzung entfernt.

Feature	Parameter	Beschreibung
Nicht-Duplizierung des Angebots (Scope-Sitzung)	`UACINoAttributeDedupeIfFewerOffer`	Wenn Sie `UACIOfferDedupeAttribute` in den Aufruf startSession einschließen, können Sie auch den Parameter `UACIOAttributeDedupeIfFewerOffer` festlegen. Damit geben Sie an, wie sich das Programm verhalten soll, wenn die Angebotsliste nach der Nicht-Duplizierung nicht genügend Angebote enthält, damit die ursprüngliche Anfrage erfüllt werden kann. Z.B. Wenn Sie `UACIOfferDedupeAttribute` so festlegen, dass die Angebotskategorie zur Nicht-Duplizierung von Angeboten verwendet wird und laut dem getOffers-Aufruf acht Angebote zurückgegeben werden sollen, führt die Nicht-Duplizierung möglicherweise zu weniger als acht qualifizierten Angeboten. In diesem Fall würde die Festlegung des Parameters `UACINoAttributeDedupeIfFewerOffer` auf `true` dazu führen, dass einige der duplizierten Angebote der Auswahlliste hinzugefügt werden, um die angeforderte Anzahl von Angeboten zu erfüllen. Standardmäßig wird `UACINoAttributeDedupeIfFewerOffer` auf 'true' gesetzt.
Angebotsunterdrückung (Scope-Sitzung)	`UACIgnoreBlackList`	TRUE – Wenn wir diesen Parameter als true übergeben, wird der Inhalt der Black List-Tabelle nicht verwendet, um die ansonsten qualifizierten Angebote zu unterdrücken.

Feature	Parameter	Beschreibung
		FALSE – Wenn dies als false übergeben wird, werden alle in der Black List verfügbaren Angebote nicht angezeigt/an den Benutzer zurückgegeben. This is the default behavior.
Angebotsunterdrückung (Scope-Sitzung)	`UACIIgnoreSuppressionRules`	TRUE – Wenn wir diesen Parameter als true übergeben, werden alle Regeln zur Angebotsunterdrückung in Echtzeit während der Angebotsentscheidung übersprungen. FALSE - Wenn wir dies als "false" übergeben, werden alle in Echtzeit unterdrückten Angebote nicht angezeigt/zurückgegeben, wie es die Regel vorsieht. This is the default behavior.
Angebotsfilterung (Scope-Aufruf)	`UACIEnableOfferMappingFilter`	Dieser Parameter wird zusammen mit dem Filternamen zur Laufzeit bei getOffers definiert. Der jeweilige Filter wird angewendet, um Angebote aus der Zuordnungstabelle von FlexOffers zu erhalten.
Angebotsfilterung (Scope-Aufruf)	`UACIDisableOfferMappingFilter`	Dieser Parameter wird zusammen mit dem Filternamen zur Laufzeit bei getOffers definiert. Der jeweilige Filter wird nicht angewendet, um Angebote aus der Tabelle zu erhalten
Ereignismusterzustände (Scope-Sitzung)	`UACIMergePatternStates`	Wird dieser Parameter auf 1 gesetzt und sich die Zielgruppen-ID einer Sitzung von anonym (nicht im Profil gefunden) zu bekannt (im Profil vorhanden) ändert, werden die in der aktuellen Sitzung ausgeführten Ereignisaktivitäten beibehalten und mit den

Feature	Parameter	Beschreibung
		bestehenden Aktivitäten der bekannten Zielgruppen-ID verbunden. Falls der Wert auf 0 gesetzt ist (Standardeinstellung), werden die Ereignisaktivitäten der anonymen Zielgruppen-ID verworfen. Bei der Angabe dieses Parameters, wird die Konfigurationseinstellung `Affinium|interact|services|eventPattern:mergeUnknownUserInSessionStates` überschrieben.
Ereignismusterzustände (Scope-Sitzung)	`UACISavePatternStates`	Wird dieser Parameter auf 1 gesetzt und die Zielgruppen-ID einer Sitzung anonym ist (nicht im Profil gefunden), werden nach Beendigung dieser Sitzung, die Zustände der in der aktuellen Sitzung aktualisierten Ereignismuster gespeichert. Wird dieser Parameter auf 0 (Standardwert) gesetzt und die Zielgruppen-ID einer Sitzung anonym ist (nicht im Profil gefunden), werden nach Beendigung dieser Sitzung, die Zustände der in der aktuellen Sitzung aktualisierten Ereignismuster verworfen. Bei der Angabe dieses Parameters, wird die folgende Konfigurationseinstellung `Affinium|interact|services|eventPattern:mergeUnknownUserInSessionStates` überschrieben.
Ereignismusterzustände	`UACIShowPatternStates`	Die Zustände von Ereignismustern, die zur aktuellen Zielgruppen-ID gehören, in

Feature	Parameter	Beschreibung
(Scope-Sitzung)		die Antwort des postEvent API Aufrufs einzuschließen.
Profilfilterung (Scope-Sitzung)	`UACIProfileFields`	Eine durch Pipe (|) getrennte Zeichenfolge, wobei jede Komponente ein Profilfeldname ist. Wenn dieser Parameter übergeben wird und nicht leer ist, werden beim Aufruf der Methode getProfile nur die angegebenen Feldwerte zurückgegeben.
Profilfilterung (Scope-Sitzung)	`UACIGetProfileShowDimFields`	Wenn dieser Parameter mit dem Wert 1 übergeben wird, werden die Dimensionsfelder (normalerweise die Felder in Dimensionsprofiltabelle(n)) beim Aufruf getProfile zurückgegeben. Andernfalls werden nur die Hauptfelder zurückgegeben. Der Standardwert ist '0'.
Angebotsfilterung (Scope-Sitzung)	`UACIQueryOffersBySQL`	Erkennung, ob die Engine die OffersBySQL Logik beim Aufruf von GetOffers ausführen soll oder nicht. Zur Aktivierung, setzen Sie `Affinium|interact|profile|Audience Levels|Customer|Offers By Raw SQL|enableOffersByRawSQL` auf true.
Angebotsfilterung (Scope-Sitzung)	`UACIOffersBySQLTemplate`	Abruf vom Angebots-SQL aus der Konfiguration (`Affinium|interact|profile|Audience Levels|Customer|Offers By Raw SQL|<SQL Template name>`). Das von diesem Parameter referenzierte SQL überschreibt das standardmäßige SQL (`Affinium|interact|profile|Audience Levels|Customer|Offers By Raw SQL|de-`

Feature	Parameter	Beschreibung
		`faultSQLTemplate`), das in der Konfiguration für die Funktion 'Angebote nach Raw SQL' angegeben wird.
Angebotsattribute (Ausgabe)	`UACICellCode`	Wenn ein Angebotsattribut ein Attribut enthält, das mit "`UACICellCode`" übereinstimmt, wird der Zellencode für das Verfahren als Wert für dieses Angebotsattribut bei einem getOffers-Aufruf zurückgegeben.
Angebotsattribute (Ausgabe)	`UACICellName`	Wenn ein Angebotsattribut ein Attribut enthält, das mit "`UACICellName`" übereinstimmt, wird der Zellenname für das Verfahren als Wert für dieses Angebotsattribut bei einem getOffers-Aufruf zurückgegeben.
Angebotsattribute (Ausgabe)	`UACIZoneID`	Wenn ein Angebotsattribut ein Attribut enthält, das mit "`UACIZoneID`" übereinstimmt, wird die Zone-ID für das Verfahren als Wert für dieses Angebotsattribut bei einem getOffers-Aufruf zurückgegeben.
Angebotsattribute (Ausgabe)	`UACISegmentID`	Wenn ein Angebotsattribut ein Attribut enthält, das mit "`UACISegmentID`" übereinstimmt, wird die Segment-ID für das Verfahren als Wert für dieses Angebotsattribut bei einem getOffers-Aufruf zurückgegeben.
Angebotsattribute (Ausgabe)	`UACIOfferID`	Wenn ein Angebotsattribut ein Attribut enthält, das mit "UACIOfferID" übereinstimmt, wird die Angebots-ID für das Verfahren als Wert für dieses Angebotsattribut bei einem getOffers-Aufruf zurückgegeben.

Feature	Parameter	Beschreibung
Angebotsattribute (Ausgabe)	`UACIOfferListID`	Wenn das Rückangebot aus einer in einer Regel konfigurierten Angebotsliste ausgewählt wird, wird dieses Attribut in die Antwort aufgenommen, um die ID der Angebotsliste für das Vefahren anzugeben.
Angebotsattribute (Ausgabe)	`UACIABTestBranchName`	Wenn das Rückangebot das Ergebnis eines A/B-Tests ist, wird dieses Attribut in die Antwort aufgenommen, um den Namen des ausgewählten A/B-Testing-Bereich anzugeben.

Reservierte Parameter für die Laufzeitumgebung

Die folgenden reservierten Parameter werden von der Laufzeitumgebung verwendet. Verwenden Sie diese Namen nicht für Ihre Ereignisparameter.

- `UACIResponseTypeCode`
- `UACIContactStatusCode`
- `UACILogToLearning`
- `UACIOfferTrackingCode`
- `UACIOfferTrackingCodeType`
- `ResetEventPattern`
- `PatternToReset`
- `ResetOfferSuppression`
- `OfferToResetSuppression`
- `UACICustomLoggerTableName`
- `UACIExecuteFlowchartByName`
- `UACIOffersBySQLTemplate`
- `UACIQueryOffersBySQL`
- `UACIGetProfileShowDimFields`
- `UACIProfileFields`
- `UACIOfferFields`

- UACIExcludeOfferFields
- UACIPurgePriorBlackListOnLoad
- UACIPurgePriorWhiteListOnLoad
- UACIMergePatternStates
- UACIResetMergedPatternStates
- UACISavePatternStates
- UACIShowPatternStates
- UACIOfferDedupeAttribute
- UACINoAttributeDedupeIfFewerOffer
- UACIIgnoreBlackList
- UACIIgnoreSuppressionRules
- UACILogSeparationFileName
- UACICachedOffers
- UACIWaitForSegmentation
- UACIPreRemoveParameter
- UACIPostRemoveParameter
- UACIInteractiveChannelName
- UACIInteractiveChannelID
- UACIInteractionPointName
- UACIInteractionPointID
- UACIEventName
- UACIEventID
- UACISessionID
- UACIEnableOfferMappingFilter
- UACIDisableOfferMappingFilter
- UACICellCode
- UACICellName
- UACIOfferID
- UACISegmentID
- UACIZoneID
- UACIABTestBranchName
- UACIEmbeddedSession

- `UACIEventTime`
- `UACIIncludeArbitration`
- `UACIIncludeAttributeMetadata`

Informationen zur Klasse AdvisoryMessage

Die Klasse `advisoryMessage` enthält Methoden, die das Empfehlungsnachrichtenobjekt definieren. Das Empfehlungsnachrichtenobjekt ist im Antwortobjekt enthalten. Jede Methode in der InteractAPI gibt ein Antwortobjekt zurück. (Ausgenommen die Methode `executeBatch`, die ein batchResponse-Objekt zurückgibt.)

Wenn es einen Fehler oder eine Warnung gibt, füllt der Unica Interact-Server das Empfehlungsnachrichtenobjekt auf. Das Empfehlungsnachrichtenobjekt enthält die folgenden Attribute:

- **DetailMessage** - eine ausführliche Beschreibung der Empfehlungsnachricht. Dieses Attribut ist möglicherweise nicht für alle Empfehlungsnachrichten verfügbar. Wenn es verfügbar ist, ist die DetailMessage möglicherweise nicht lokalisiert.
- **Message** - eine Kurzbeschreibung der Empfehlungsnachricht.
- **MessageCode** - eine Codenummer für die Empfehlungsnachricht.
- **StatusLevel** - eine Codenummer für die Dringlichkeit der Empfehlungsnachricht.

Sie rufen die advisoryMessage-Objekte mithilfe der Methode `getAdvisoryMessages` ab.

getDetailMessage

Die `getDetailMessage`-Methode gibt die ausführliche und detaillierte Beschreibung eines Advisory Message-Objekts zurück. Nicht alle Nachrichten haben eine detaillierte Nachricht.

```
getDetailMessage()
```

Rückgabewert

Das Advisory Message-Objekt gibt eine Zeichenfolge zurück.

Beispiel

```
// 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

Die `getMessage`-Methode gibt die Kurzbeschreibung eines Advisory Message-Objekts zurück.

```
getMessage()
```

Rückgabewert

Das Advisory Message-Objekt gibt eine Zeichenfolge zurück.

Beispiel

Die folgende Methode druckt die Nachricht und die detaillierte Nachricht eines AdvisoryMessage-Objekts aus.

```
// 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

Die `getMessageCode`-Methode gibt den internen Fehlercode zurück, der einem Advisory Message-Objekt zugeordnet ist, wenn die Statusebene 2 ist (STATUS_LEVEL_ERROR).

```
getMessageCode()
```

Rückgabewert

Das AdvisoryMessage-Objekt gibt eine Ganzzahl zurück.

Beispiel

Die folgende Methode druckt den Nachrichtencode eines AdvisoryMessage-Objekts aus.

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

getStatusLevel

Die `getStatusLevel`-Methode gibt die Statusebene eines Advisory Message-Objekts zurück.

```
getStatusLevel()
```

Rückgabewert

Das Advisory Message-Objekt gibt eine Ganzzahl zurück.

- 0 - STATUS_LEVEL_SUCCESS - Die aufgerufene Methode wurde ohne Fehler ausgeführt.
- 1 - STATUS_LEVEL_WARNING - Die aufgerufene Methode wurde mit mindestens einer Warnung (aber ohne Fehler) ausgeführt.
- 2 - STATUS_LEVEL_ERROR - Die aufgerufene Methode wurde nicht erfolgreich ausgeführt und weist mindestens einen Fehler auf.

Beispiel

Die folgende Methode druckt die Statusebene eines AdvisoryMessage-Objekts aus.

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

Informationen zur Klasse AdvisoryMessageCode

Die Klasse `advisoryMessageCode` enthält Methoden, die die Empfehlungsnachrichtencodes definieren. Sie rufen die Empfehlungsnachrichtencodes mit der Methode `getMessageCode` ab.

Codes von Empfehlungsnachrichten

Sie rufen die Empfehlungsnachrichtencodes mit der Methode `getMessageCode` ab.

In dieser Tabelle werden die Codes von Empfehlungsnachrichten aufgelistet und beschrieben:

Code	Nachrichtentext	Syntax
1	INVALID_SESSION_ID	Die Sitzungs-ID verweist nicht auf eine gültige Sitzung.
2	ERROR_TRYING_TO_ABORT_ SEGMENTATION	Beim Versuch, die Segmentierung während eines `endSession`-Aufrufs abzubrechen, ist ein Fehler aufgetreten.
3	INVALID_INTERACTIVE_CHANNEL	Das für den interaktiven Kanal übergebene Argument verweist nicht auf einen gültigen interaktiven Kanal.
4	INVALID_EVENT_NAME	Das für das Ereignis übergebene Argument verweist nicht auf ein gültiges Ereignis für den aktuellen interaktiven Kanal.
5	INVALID_INTERACTION_POINT	Das für den Interaktionspunkt übergebene Argument verweist nicht auf einen gültigen Interaktionspunkt für den aktuellen interaktiven Kanal.
6	ERROR_WHILE_MAKING_INITIAL_ SEGMENTATION_REQUEST	Bei der Übergabe einer Segmentierungsanfrage ist ein Fehler aufgetreten.
7	SEGMENTATION_RUN_FAILED	Die Segmentierung wurde nur teilweise ausgeführt und ist dann fehlgeschlagen.
8	PROFILE_LOAD_FAILED	Das Laden der Profil- oder Dimensionstabellen ist fehlgeschlagen.
9	OFFER_SUPPRESSION_LOAD_FAILED	Das Laden der Angebotsunterdrückungstabelle ist fehlgeschlagen.

Code	Nachrichtentext	Syntax
10	COMMAND_METHOD_ UNRECOGNIZED	Für einen Befehl innerhalb eines `executeBatch`-Aufrufs wurde eine ungültige Befehlsmethode angegeben.
11	ERROR_TRYING_TO_POST_EVENT _-PARAMETERS	Bei der Eingabe der Ereignisparameter ist ein Fehler aufgetreten.
12	LOG_SYSTEM_EVENT_EXCEPTION	Ausnahmebedingung bei der Übergabe des Systemereignisses (Sitzung beenden, Angebot erhalten, Profil erhalten, Zielgruppe einstellen, Debuggen einstellen oder Sitzung starten) zur Protokollierung.
13	LOG_USER_EVENT_EXCEPTION	Ausnahmebedingung bei der Übergabe des Benutzerereignisses zur Protokollierung.
13	ERROR_TRYING_TO_LOOK_UP_EVENT	Bei der Suche nach dem Ereignisnamen ist ein Fehler aufgetreten.
15	ERROR_TRYING_TO_LOOK_UP _INTERACTIVE_CHANNEL	Bei der Suche nach dem Namen des interaktiven Kanals ist ein Fehler aufgetreten.
16	ERROR_TRYING_TO_LOOK_UP _INTERACTION_POINT	Bei der Suche nach dem Namen des Interaktionspunkts ist ein Fehler aufgetreten.
17	RUNTIME_EXCEPTION_ ENCOUNTERED	Es ist eine nicht erwartete Laufzeitausnahmebedingung aufgetreten.
18	ERROR_TRYING_TO_EXECUTE_ ASSOCIATED_ACTION	Fehler beim Ausführen der zugehörigen Aktion (erneute Segmentierung auslösen, Angebotskontakt protokollieren,

Code	Nachrichtentext	Syntax
		Angebotsannahme protokollieren oder Angebotsablehnung protokollieren).
19	ERROR_TRYING_RUN_FLOWCHART	Bei der Ausführung des Ablaufdiagramms ist ein Fehler aufgetreten.
20	FLOWCHART_FAILED	Eine Ablaufdiagrammausführung ist fehlgeschlagen.
21	FLOWCHART_ABORTED	Eine Ablaufdiagrammausführung wurde abgebrochen.
22	FLOWCHART_NEVER_RUN	Ein angegebenes Ablaufdiagramm wurde nie ausgeführt.
23	FLOWCHART_STILL_RUNNING	Ein Ablaufdiagramm wird immer noch ausgeführt.
24	ERROR_WHILE_READING_ PARAMETERS	Beim Lesen von Parametern ist ein Fehler aufgetreten.
25	ERROR_WHILE_LOADING_ RECOMMENDED_OFFERS	Fehler beim Laden der empfohlenen Angebote
26	ERROR_WHILE_LOGGING_DEFAULT _-TEXT_STATISTICS	Fehler beim Protokollieren von Standardtextstatistiken (Anzahl, wie oft die Standardzeichenfolge für den Interaktionspunkt angezeigt wurde).
27	SCORE_OVERRIDE_LOAD_FAILED	Die Bewertungsüberschreibungstabelle konnte nicht geladen werden.
28	NULL_AUDIENCE_ID	Die Zielgruppen-ID ist leer.
29	UNRECOGNIZED_AUDIENCE_LEVEL	Eine nicht erkannte Zielgruppenebene wurde angegeben.
30	MISSING_AUDIENCE_FIELD	Ein Zielgruppenfeld fehlt.

Code	Nachrichtentext	Syntax
31	INVALID_AUDIENCE_FIELD_TYPE	Ein ungültiger Zielgruppenfeldtyp wurde angegeben.
32	UNSUPPORTED_AUDIENCE_FIELD _TYPE	Nicht unterstützter Zielgruppenfeldtyp
33	TIMEOUT_REACHED_ON_GET_ OFFERS_CALL	Das Zeitlimit für den Aufruf `getOffers` wurde ohne Rückgabe von Angeboten erreicht.
34	INTERACT_INITIALIZATION_NOT_ COMPLETED_SUCCESSFULLY	Die Initialisierung des Laufzeitservers wurde nicht erfolgreich abgeschlossen.
35	SESSION_ID_UNDEFINED	Die Sitzungs-ID ist nicht definiert.
36	INVALID_NUMBER_OF_OFFERS_ REQUESTED	Eine ungültige Anzahl an Angeboten wurde angefordert.
37	NO_SESSION_EXIST_BUT_WILL_ CREATE_NEW_ONE	Da keine Sitzung vorhanden war, wurde eine Sitzung erstellt.
38	AUDIENCE_ID_NOT_FOUND_IN_ PROFILE_TABLE	Die angegebene Zielgruppen-ID ist nicht in der Profiltabelle enthalten.
39	LOG_CUSTOM_LOGGER_EVENT_ EXCEPTION	Bei der Übergabe eines Datenereignisses zur benutzerdefinierten Protokollierung ist eine Ausnahmebedingung aufgetreten.
40	SPECIFIED_FLOWCHART_FOR_ EXECUTION_DOES_NOT_EXIST	Das angegebene Ablaufdiagramm kann nicht ausgeführt werden, da es nicht vorhanden ist.
41	AUDIENCE_NOT_DEFINED_IN_ CONFIGURATION	Die angegebene Zielgruppe ist nicht in der Konfiguration definiert.

Code	Nachrichtentext	Syntax
42	Angebote können nicht aus SQL geladen werden	Das rohe SQL kann keine Angebote laden.
43	Ungültige Duplizierungsrichtlinie	Die Duplizierungsrichtlinie für Sitzungs-ID ist nicht gültig.
44	Fehler beim Speichern der Ereignismusterzustände für Zielgruppen-ID	Beim Speichern der Ereignismusterzustände für Zielgruppen-ID ist ein Fehler aufgetreten.
45	Ungültiger Ausdruck	Der Ausdruck ist ungültig
46	Leerer Ausdruck	Der Ausdruck ist leer.
47	Fehler beim Versuch, ein Ereignismuster auszuwerten	Beim Versuch, ein Ereignismuster auszuwerten, ist ein Fehler aufgetreten.
48	Fehler beim Laden des Ereignismusterzustands für Zielgruppen-ID	Beim Laden des Ereignismusterzustands für die Zielgruppen-ID ist ein Fehler aufgetreten.
49	Fehler beim Laden/Aktualisieren der Sitzung	Beim Laden oder Aktualisieren der Sitzung ist ein Fehler aufgetreten.
50	Fehler beim Laden/Aktualisieren des Ereignismusterzustands für Zielgruppen-ID	Beim Laden oder Aktualisieren des Ereignismusterzustands für die Zielgruppen-ID ist ein Fehler aufgetreten
51	Fehler beim Erstellen des Laufzeitdienstes OpDetection	Beim Erstellen des Laufzeitdienstes OpDetection ist ein Fehler aufgetreten.
52	Fehler beim Abrufen der letzten Ereignismusterzustände von OpDetection für sessionId	Beim Abrufen der letzten Ereignismusterzustände von OpDetection für sessionId ist ein Fehler aufgetreten.

Code	Nachrichtentext	Syntax
53	Fehler beim Posten eines Ereignisses in OpDetection für sessionId	Ein Fehler ist beim Posten des Ereignisses in OpDetection für sessionId aufgetreten.
54	Fehler beim Kontaktieren des OpDetection-Dienstes für Server	Bei der Kontaktaufnahme mit dem OpDetection-Dienst für den Server ist ein Fehler aufgetreten.
55	OpDetection-Dienst wird in den nächsten {0} Minuten nicht mehr kontaktiert	OpDetection-Dienst kann während der nächsten {0} Minuten nicht kontaktiert werden.
56	Fehlende(r) erforderliche(r) Parameter {0} für Ereignis {1}.	Erforderliche(r) Parameter {0} für {1}Ereignis nicht vorhanden.
57	Das Ereignis ist für die Protokollierung als Kontakt konfiguriert, es wurden jedoch keine Angebote empfohlen	Das Ereignis ist für die Protokollierung als Kontakt konfiguriert, es wurden jedoch keine Angebote empfohlen
58	Protokoll zur Kontaktaufnahme angefordert, aber treatmentCode nicht gültig	Das Protokoll zur Kontaktaufnahme wurde angefordert, aber der treatmentCode ist für sessionCode nicht gültig.
59	Bei der Ausführung des erweiterten Szenariosimulators konnte keine Stapelantwort abgerufen werden. Weitere Einzelheiten hierzu finden Sie in der Protokolldatei auf dem Server.	Es konnte keine Stapelantwort für die Ausführung des erweiterten Szenariosimulators abgerufen werden. Weitere Einzelheiten hierzu finden Sie in der Protokolldatei auf dem Server.
60	Protokoll für Antwort angefordert, aber Aufzeichnungscode nicht gültig für sessionCode	Protokoll für Antwort angefordert, aber Aufzeichnungscode nicht gültig für sessionCode.
61	Protokoll zur Kontaktaufnahme angefordert, aber Kontaktstatuscode nicht	Protokoll zur Kontaktaufnahme angefordert, aber Kontaktstatuscode nicht gültig für sessionCode.

Code	Nachrichtentext	Syntax
	gültig für sessionCode: {0} Statuscode des Kontakts: {1}	
62	Protokoll für Antwort angefordert, aber Antwortencode nicht gültig für session-Code {0} Code des Antworttyps: {1}	Protokoll für Antwort angefordert, aber Antwortencode nicht gültig für session-Code.
63	Die Authentifizierung ist fehlgeschla-gen. Token ist nicht verfügbar	Die Authentifizierung ist fehlgeschla-gen, da das Token nicht verfügbar ist.
64	Die Authentifizierung ist fehlgeschla-gen. Sitzungs-ID ist nicht verfügbar	Die Authentifizierung ist fehlgeschla-gen, da die Sitzungs-ID nicht verfügbar ist.
65	Die Authentifizierung ist fehlgeschla-gen. Das Token ist der angeforderten Sitzungs-ID nicht zugeordnet.	Die Authentifizierung ist fehlgeschla-gen, da das Token nicht der angefor-derten Sitzungs-ID zugeordnet ist.
66	Unberechtigte Anforderung. Ungültiges Token oder ungültige Berechtigungs-nachweise.	Die Anforderung ist nicht autorisiert, da Token oder Berechtigungsnachweise ungültig sind.
67	Fehler beim Erstellen des Tokens	Beim Erstellen eines Tokens ist ein Fehler aufgetreten.
68	Fehler: Bei der Ausführung des einfa-chen Szenariosimulators konnte keine Stapelantwort abgerufen werden. Wei-tere Einzelheiten hierzu finden Sie in der Protokolldatei auf dem Server.	Bei der Ausführung des einfachen Sze-nariosimulators konnte keine Stapel-antwort abgerufen werden.
69	Keine gültige Lizenz	Die Lizenz ist ungültig.
70	Sitzung kann nicht erstellt oder gefun-den werden	Sitzung kann nicht erstellt oder gefun-den werden

Code	Nachrichtentext	Syntax
71	Flussdiagramm für Sitzungs-ID fehlgeschlagen: {0}	Das Flussdiagramm hat die Sitzung nicht bestanden.
72	Zeitüberschreitung bei der Ausführung des Flussdiagramms für die Sitzungs-ID: {0}	Zeitüberschreitung bei der Ausführung des Flussdiagramms für die Sitzung.
73	Die Kampagnen-API zum Abrufen statischer Segmente ist für die Sitzungs-ID fehlgeschlagen: {0}	Die Kampagnen-API zeigt an, dass statische Segmente für die Sitzung fehlgeschlagen sind.
74	Kampagnen-API zum Abrufen einer Zeitüberschreitung für statische Segmente für die Sitzungs-ID: {0}	Die Kampagnen-API erhält eine Zeitüberschreitung für statische Segmente für die Sitzung.
101	Antwort für Befehl	Antwort für Befehl
102	Statuscode	Statuscode
103	API-Version	API-Version
104	Sitzungs-ID	Sitzungs-ID
105	Empfehlungsnachricht	Empfehlungsnachricht
106	Nachrichtencode	Nachrichtencode
107	Nachricht	Nachricht
108	Detaillierte Nachricht	Detaillierte Nachricht
109	Statusebene	Statusebene
110	Keine Empfehlungsnachricht	Keine Empfehlungsnachricht
111	Kein Profildatensatz	Kein Profildatensatz
112	Angebote für Interaktionspunkt	Angebote für Interaktionspunkt
113	Standardtext	Standardtext

Code	Nachrichtentext	Syntax
114	Empfohlenen Angebote	Empfohlenen Angebote
115	Angebotsbewertung	Angebotsbewertung
116	Angebotsname	Angebotsname
117	Angebotsbeschreibung	Angebotsbeschreibung
118	Angebotscode	Angebotscode
119	Angebotsparameter	Angebotsparameter120
120	Verfahrenscode	Verfahrenscode
121	Keine empfohlenen Angebote	Keine empfohlenen Angebote
122	Keine Angebotsliste	Keine Angebotsliste
123	Name	Name
124	Typ	Typ
125	Datumswert	Datumswert
126	Zeichenfolgewert	Zeichenfolgewert
127	Numerischer Wert	Numerischer Wert
128	Gültigen Empfänger und/oder gültiges Gateway nicht gefunden	Gültigen Empfänger und/oder gültiges Gateway nicht gefunden
129	Ungültiger Empfänger: {0}	Der Empfänger ist ungültig.
130	Ungültiges Gateway: {0}	Das Gateway ist ungültig.
131	Nachricht an Empfänger {0} für Gateway {1} erfolgreich gesendet	Nachricht an Empfänger für Gateway erfolgreich gesendet
132	Kein gültiger Empfänger verfügbar	Kein gültiger Empfänger verfügbar
133	Ungültiges Format des Parameters - {0}	Das Format des Parameters ist ungültig.

Code	Nachrichtentext	Syntax
134	Ungültiges Format von AudienceID	Das Format von AudienceID ist ungültig.
135	Ungültiger Parameter {0}, ungültiger Wert {1}.	Der Parameter ist mit einem ungültigen Wert ungültig.
136	Ungültige AudienceID {0}, ungültiger Wert {1}	Die AudienceID ist mit einem ungültigen Wert ungültig.
137	Ungültiger Parameter {0}, ungültiger Datentyp {1}.	Der Parameter ist mit einem ungültigen Datentyp ungültig.
138	Ungültige AudienceID {0}, ungültiger Datentyp {1}	Die AudienceID ist mit einem ungültigen Datentyp ungültig.
139	Ungültiger Parameter {0}, erwarteter Datentyp {1}, aber empfangenes {2}	Ungültiger Parameter {0}, erwarteter Datentyp {1}, aber empfangenes {2}
160	Das Angebot zum Zurücksetzen der Unterdrückungsregeln wurde nicht gefunden: {0}.	Das Angebot zum Zurücksetzen der Unterdrückungsregeln wurde nicht gefunden.
141	Das zurückzusetzende Ereignismuster kann nicht gefunden werden: {0}	Das zurückzusetzende Ereignismuster kann nicht gefunden werden.
142	Ungültiges Anforderungsformat, fehlender Parameter: {0}	Das Anforderungsformat ist mit einem fehlenden Parameter ungültig.
201	Die für die API bereitgestellten Eingaben sind ungültig	Die für die API bereitgestellten Eingaben sind ungültig.
202	Interner Fehler aufgetreten. Weitere Informationen finden Sie in den Protokollen.	Ein interner Fehler ist aufgetreten. Weitere Informationen finden Sie in den Protokollen.
203	Ungültige IC-ID oder Name: {0}	Die IC-ID oder der IC-Name ist ungültig.

Code	Nachrichtentext	Syntax
204	Fehler beim Suchen des IC-Namens: {0} im Bereitstellungscache.	Fehler beim Suchen des IC-Namens: {0} im Bereitstellungscache.
205	Der Eingabeparameter {0} fehlt.	Der Eingabeparameter fehlt.
206	Authentifizierung fehlgeschlagen	Authentifizierung fehlgeschlagen
207	Benutzername ist erforderlich	Der Benutzername ist erforderlich.

Informationen zur Klasse BatchResponse

Die Klasse `BatchResponse` enthält Methoden, die die Ergebnisse der Methode `executeBatch` definieren.

Das Batch-Antwortobjekt enthält die folgenden Attribute:

- **BatchStatusCode** - Der höchste Statuscodewert für alle Antworten, die von der Methode `executeBatch` angefordert werden.
- **Responses** - Ein Array der Antwortobjekte, die von der Methode `executeBatch` angefordert werden.

getBatchStatusCode

Die `getBatchStatusCode`-Methode gibt den höchsten Statuscode aus dem Array von Befehlen zurück, die die `executeBatch`-Methode ausgeführt hat.

```
getBatchStatusCode()
```

Rückgabewert

Die `getBatchStatusCode`-Methode gibt eine Ganzzahl zurück.

- 0 - STATUS_SUCCESS - Die aufgerufene Methode wurde ohne Fehler ausgeführt.
- 1 - STATUS_WARNING - Die aufgerufene Methode wurde mit mindestens einer Warnung (aber ohne Fehler) ausgeführt.
- 2 - STATUS_ERROR - Die aufgerufene Methode wurde nicht erfolgreich ausgeführt und weist mindestens einen Fehler auf.

Beispiel

Das folgende Codebeispiel zeigt ein Beispiel zum Abrufen von `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

Die `getResponses`-Methode gibt das Array von Antwortobjekten zurück, die dem Array von Befehlen entsprechen, die die `executeBatch`-Methode ausgeführt hat.

```
getResponses()
```

Rückgabewert

Die `getResponses`-Methode gibt ein Array von `Response`-Objekten zurück.

Beispiel

Das folgende Beispiel wählt alle Antworten aus und druckt alle Advisory Messages, wenn der Befehl nicht erfolgreich war.

```
for(Response response : batchResponse.getResponses())
{
    if(response.getStatusCode()!=Response.STATUS_SUCCESS)
    {
        printDetailMessageOfWarningOrError("executeBatchCommand",
   response.getAdvisoryMessages());
    }
}
```

Informationen zur Command-Benutzeroberfläche

Die Methode `executeBatch` erfordert, dass Sie ein Array von Objekten übergeben, das die Command-Benutzeroberfläche implementiert. Sie sollten die Standardimplementierung `CommandImpl` verwenden, um die Befehlsobjekte zu übergeben.

Die folgende Tabelle listet den Befehl, die Methode der Interact-API-Klasse, die der Befehl darstellt, und die Command-Benutzeroberflächenmethoden auf, die Sie für jeden Befehl verwenden müssen. Sie müssen keine Sitzungs-ID einbeziehen, weil die Methode `executeBatch` bereits die Sitzungs-ID enthält.

Befehl	Interact-API-Methode	Command-Benutzer-oberflächenmethode
COMMAND_ENDSESSION	endSession	Keine.
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	Keine.
COMMAND_GETVERSION	getVersion	Keine.
COMMAND_POSTEVENT	postEvent	• setEvent • setEventParameters
COMMAND_SETAUDIENCE	setAudience	• setAudienceID • setAudienceLevel • setEventParameters
COMMAND_SETDEBUG	setDebug	setDebug
COMMAND_STARTSESSION	startSession	• setAudienceID • setAudienceLevel • setDebug • setEventParameters • setInteractiveChannel • setRelyOnExistingSession

setAudienceID

Die `setAudienceID`-Methode definiert die AudienceID für die Befehle `setAudience` und `startSession`.

```
setAudienceID(audienceID)
```

- **audienceID** - ein Array von NameValuePair-Objekten, die die AudienceID definieren.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `startSession` und `setAudience` aufruft.

```
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

Die `setAudienceLevel`-Methode definiert die Zielgruppenebene für die Befehle `setAudience`und `startSession`.

```
setAudienceLevel(audienceLevel)
```

- audienceLevel- eine Zeichenfolge, die die Zielgruppenebene enthält.

Wichtig: Der *audienceLevel*-Name muss exakt mit dem in Unica Campaign definierten Namen der Zielgruppenebene übereinstimmen. Hierbei muss auf die Groß-/Kleinschreibung geachtet werden.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `startSession` und `setAudience` aufruft.

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

```
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Make the call */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

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

setDebug

Die `setDebug`-Methode definiert die Fehlerbehebungsstufe für den `startSession`-Befehl.

```
setDebug(debug)
```

Wenn der Wert `true` ist, protokolliert der Laufzeitserver die Daten zur Fehlerbehebung im Protokoll des Laufzeitservers. Wenn der Wert `false` ist, protokolliert der Laufzeitserver keine Daten zur Fehlerbehebung. Das Debug-Flag wird für jede Sitzung individuell gesetzt. Somit können Sie die Daten zur Fehlerbehebung für einzelne Laufzeitsitzungen verfolgen.

- **debug** - ein boolescher Wert (`true` oder `false`).

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `startSession` und `setDebug` aufruft.

```
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

Die `setEvent`-Methode definiert den Namen des Ereignisses, das der `postEvent`-Befehl verwendet.

```
setEvent(event)
```

- **event** - Eine Zeichenfolge, die den Ereignisnamen enthält.

Wichtig: Der *event*-Name muss exakt mit dem im interaktiven Kanal definierten Namen übereinstimmen. Hierbei muss auf die Groß-/Kleinschreibung geachtet werden.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `postEvent` aufruft.

```
String eventName = "SearchExecution";

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

setEventParameters

Die `setEventParameters`-Methode definiert die Ereignisparameter, das der `postEvent`-Befehl verwendet. Diese Werte werden in den Sitzungsdaten gespeichert.

```
setEventParameters(eventParameters)
```

- **eventParameters** - ein Array von NameValuePair-Objekten, die die Ereignisparameter definieren.

Wenn das Ereignis zum Beispiel ein Angebot im Kontaktprotokoll protokolliert, müssen Sie den Verfahrenscode des Angebots einschließen.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `postEvent` aufruft.

```
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

Die setGetOfferRequests-Methode legt den Parameter zum Abrufen der Angebote fest, die der `getOffersForMultipleInteractionPoints`-Befehl verwendet.

```
setGetOfferRequests(numberRequested)
```

- **numberRequested** - ein Array von `GetOfferRequest`-Objekten, die den Parameter zum Abrufen von Angeboten definieren.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `GetOfferRequest`-Methode, die `setGetOfferRequests` aufruft.

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

```
    GetOfferRequest request2 = new GetOfferRequest(3,
GetOfferRequest.ALLOW_DUPLICATION);
    request2.setIpName("IP2");
    OfferAttributeRequirements offerAttributes2 = new
OfferAttributeRequirements();
    offerAttributes2.setNumberRequested(3);
    offerAttributes2.setAttributes(new NameValuePairImpl[] {new
NameValuePairImpl("attr5",
         NameValuePair.DATA_TYPE_STRING, "value5")});
    request2.setOfferAttributes(offerAttributes2);

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

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

setInteractiveChannel

Die `setInteractiveChannel`-Methode definiert den Namen des interaktiven Kanals, den der `startSession`-Befehl verwendet.

```
setInteractiveChannel(interactiveChannel)
```

- **interactiveChannel**- eine Zeichenfolge, die den Namen des interaktiven Kanals enthält.

Wichtig: Der *interactiveChannel* muss genau mit dem in Unica Campaign definierten Namen des interaktiven Kanals übereinstimmen. Hierbei muss auf die Groß-/Kleinschreibung geachtet werden.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `startSession` aufruft.

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

setInteractionPoint

Die `setInteractionPoint`-Methode definiert den Namen des Interaktionspunkts, den die Befehle `getOffers` und `postEvent` verwenden.

```
setInteractionPoint(interactionPoint)
```

- **interactionPoint** - eine Zeichenfolge, die den Namen des Interaktionspunkts enthält.

Wichtig: Der *interactionPoint* muss genau mit dem im interaktiven Kanal definierten Namen des Interaktionspunktes übereinstimmen. Hierbei muss auf die Groß-/Kleinschreibung geachtet werden.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `getOffers` aufruft.

```
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

Die `setMethodIdentifier`-Methode definiert den Typ des Befehls, der im Befehlsobjekt enthalten ist.

```
setMethodIdentifier(methodIdentifier)
```

- **methodIdentifier** - eine Zeichenfolge, die den Typ des Befehls enthält.

  Gültige Werte sind:

  - **COMMAND_ENDSESSION** - stellt die `endSession`-Methode dar.
  - **COMMAND_GETOFFERS** - stellt die `getOffers`-Methode dar.
  - **COMMAND_GETPROFILE** - stellt die `getProfile`-Methode dar.
  - **COMMAND_GETVERSION** - stellt die `getVersion-Methode` dar.
  - **COMMAND_POSTEVENT** - stellt die `postEvent`-Methode dar.
  - **COMMAND_SETAUDIENCE** - stellt die `setAudience-Methode` dar.
  - **COMMAND_SETDEBUG** - stellt die `setDebug`-Methode dar.
  - **COMMAND_STARTSESSION** - stellt die `startSession`-Methode dar.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `getVersion` und `endSession` aufruft.

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

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

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

setNumberRequested

Die `setNumberRequested`-Methode definiert die Anzahl der Angebote, die der `getOffers`-Befehl anfordert.

```
setNumberRequested(numberRequested)
```

- **numberRequested** - eine Ganzzahl, die die Anzahl an Angeboten definiert, die der `getOffers`-Befehl anfordert.

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `getOffers` aufruft.

```
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

Die `setRelyOnExistingSession`-Methode definiert eine boolesche Variable, die definiert, ob der `startSession`-Befehl eine vorhandene Sitzung verwendet oder nicht.

```
setRelyOnExistingSession(relyOnExistingSession)
```

Wenn der Wert `true` ist, muss die Sitzungs-ID für `executeBatch` mit einer vorhandenen Sitzungs-ID übereinstimmen. Wenn der Wert `false` ist, müssen Sie eine neue Sitzungs-ID mit der `executeBatch`-Methode angeben.

- **relyOnExistingSession** - ein boolescher Wert (`true` oder `false`).

Rückgabewert

Keine.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer `executeBatch`-Methode, die `startSession` aufruft.

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

Informationen zur NameValuePair-Benutzeroberfläche

Viele Methoden in der Unica Interact-API geben entweder NameValuePair-Objekte zurück oder erfordern, dass Sie NameValuePair-Objekte als Argumente übergeben. Bei der Übergabe als Argumente in eine Methode sollten Sie die Standardimplementierung `NameValuePairImpl` verwenden.

getName

Die `getName`-Methode gibt die Namenskomponente eines NameValuePair-Objekts zurück.

```
getName()
```

Rückgabewert

Die `getName`-Methode gibt eine Zeichenfolge zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die das Antwortobjekt für `getProfile` verarbeitet.

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

getValueAsDate

Die `getValueAsDate`-Methode gibt den Wert eines NameValuePair-Objekts zurück.

```
getValueAsDate()
```

Verwenden Sie `getValueDataType`, bevor Sie `getValueAsDate` verwenden, um sicherzustellen, dass auf den richtigen Datentyp verwiesen wird.

Rückgabewert

Die `getValueAsDate`-Methode gibt ein Datum zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die ein NameValuePair verarbeitet und den Wert ausdruckt, wenn es sich um ein Datum handelt.

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

getValueAsNumeric

Die `getValueAsNumeric`-Methode gibt den Wert eines NameValuePair-Objekts zurück.

```
getValueAsNumeric()
```

Verwenden Sie `getValueDataType`, bevor Sie `getValueAsNumeric` verwenden, um sicherzustellen, dass auf den richtigen Datentyp verwiesen wird.

Rückgabewert

Die `getValueAsNumeric`-Methode gibt ein Doppelzeichen zurück. Beispiel: Wenn Sie einen ursprünglich als Ganzzahl in der Profiltabelle gespeicherten Wert abrufen, `gibt getValueAsNumeric` ein Doppelzeichen zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die ein NameValuePair verarbeitet und den Wert ausdruckt, wenn dieser numerisch ist.

```
if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
{
```

```
    System.out.println("Value:"+nvp.getValueAsNumeric());
}
```

getValueAsString

Die `getValueAsString`-Methode gibt den Wert eines NameValuePair-Objekts zurück.

```
getValueAsString()
```

Verwenden Sie `getValueDataType`, bevor Sie `getValueAsString` verwenden, um sicherzustellen, dass auf den richtigen Datentyp verwiesen wird.

Rückgabewert

Die `getValueAsString`-Methode gibt eine Zeichenfolge zurück. Beispiel: Wenn Sie einen ursprünglich als char, varchar oder char[10] in der Profiltabelle gespeicherten Wert abrufen, gibt `getValueAsString` eine Zeichenfolge zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die ein NameValuePair verarbeitet und den Wert ausdruckt, wenn es sich um eine Zeichenfolge handelt.

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

getValueDataType

Die `getValueDataType`-Methode gibt den Datentyp eines NameValuePair-Objekts zurück.

```
getValueDataType()
```

Verwenden Sie `getValueDataType`, bevor Sie `getValueAsDate`, `getValueAsNumeric` oder `getValueAsString` verwenden, um sicherzustellen, dass auf den richtigen Datentyp verwiesen wird.

Rückgabewert

Die `getValueDataType`-Methode gibt eine Zeichenfolge zurück, die angibt, ob das NameValuePair ein Datum, eine Zahl oder eine Zeichenfolge enthält.

Gültige Werte sind:

- **DATA_TYPE_DATETIME** - ein Datum, das einen Wert mit Datum und Uhrzeit enthält.
- **DATA_TYPE_NUMERIC** - ein Doppelzeichen, das einen Zahlenwert enthält.
- **DATA_TYPE_STRING** - eine Zeichenfolge, die einen Textwert enthält.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die das Antwortobjekt aus einer `getProfile`-Methode verarbeitet.

```
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

Die `setName`-Methode definiert die Namenskomponente eines NameValuePair-Objekts.

```
setName(name)
```

- **name** - eine Zeichenfolge, die die Namenskomponente eines NameValuePair-Objekts enthält.

Rückgabewert

Keine.

Beispiel

Im folgenden Beispiel wird die Namenskomponente in einem NameValuePair definiert.

```
NameValuePair custId = new NameValuePairImpl();
custId.setName("CustomerId");
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
```

setValueAsDate

Die `setValueAsDate`-Methode definiert den Wert eines NameValuePair-Objekts.

```
setValueAsDate(valueAsDate)
```

- **valueAsDate** - ein Datum, das den Wert mit Datum und Uhrzeit für das NameValuePair-Objekt enthält.

Rückgabewert

Keine.

Beispiel

Im folgenden Beispiel wird die Wertkomponente in einem NameValuePair definiert, wenn der Wert ein Datum ist.

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

setValueAsNumeric

Die `setValueAsNumeric`-Methode definiert den Wert eines NameValuePair-Objekts.

```
setValueAsNumeric(valueAsNumeric)
```

- **valueAsNumeric** - ein Doppelzeichen, das den numerischen Wert eines NameValuePair-Objekts enthält.

Rückgabewert

Keine.

Beispiel

Im folgenden Beispiel wird die Wertkomponente in einem NameValuePair definiert, wenn es sich um einen numerischen Wert handelt.

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

setValueAsString

Die `setValueAsString`-Methode definiert den Wert eines NameValuePair-Objekts.

```
setValueAsString(valueAsString)
```

- **valueAsString** - eine Zeichenfolge, die den Wert eines NameValuePair-Objekts enthält

Rückgabewert

Keine.

Beispiel

Im folgenden Beispiel wird die Wertkomponente in einem NameValuePair definiert, wenn es sich um einen numerischen Wert handelt.

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

setValueDataType

Die `setValueDataType`-Methode definiert den Datentyp eines NameValuePair-Objekts.

```
getValueDataType(valueDataType)
```

Gültige Werte sind:

- **DATA_TYPE_DATETIME** - ein Datum, das einen Wert mit Datum und Uhrzeit enthält.
- **DATA_TYPE_NUMERIC** - ein Doppelzeichen, das einen Zahlenwert enthält.
- **DATA_TYPE_STRING** - eine Zeichenfolge, die einen Textwert enthält.

Rückgabewert

Keine.

Beispiel

Im folgenden Beispiel wird der Datentyp des Werts in einem NameValuePair festgelegt.

```
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(Umfang)

Die setScope-Methode definiert die Lebensdauer dieses NameValuePair-Objekts.

Die folgenden Werte sind zulässig.

- `Scope.INVOCATION`: Das NameValuePair-Objekt ist während des Prozesses dieses API-Aufrufs wirksam. Sein Wert wird nicht in der Sitzung gespeichert.
- `Scope.SESSION`: Das NameValuePair-Objekt ist vom Beginn dieses API-Aufrufs an wirksam, bis es entfernt wird. Sein Wert wird in der Sitzung gespeichert. Dies ist die Standardeinstellung.

Rückgabewert

Keine

getScope()

Die getScope-Methode erhält den Umfang der NamevaluePair-Methode.

Die folgenden Werte sind zulässig.

- `Scope.INVOCATION`: Das NameValuePair-Objekt ist während des Prozesses dieses API-Aufrufs wirksam. Sein Wert wird nicht in der Sitzung gespeichert.
- `Scope.SESSION`: Das NameValuePair-Objekt ist vom Beginn dieses API-Aufrufs an wirksam, bis es entfernt wird. Sein Wert wird in der Sitzung gespeichert.

Rückgabewert

Die getScope-Methode gibt einen Wert von Scope enum zurück.

Informationen zur Klasse Offer

Die Klasse `Offer` enthält Methoden, die ein Offer-Objekt definieren. Dieses Angebotsobjekt enthält viele der Eigenschaften eines Angebots in Unica Campaign.

Das Angebotsobjekt enthält die folgenden Attribute:

- **AdditionalAttributes** - Name/Wert-Paare, die benutzerdefinierte Angebotsattribute enthalten, die Sie in Unica Campaign definiert haben.
- **Description** - Die Beschreibung des Angebots.
- **EffectiveDate** - Das Wirksamkeitsdatum des Angebots.
- **ExpirationDate** - Das Ablaufdatum des Angebots.
- **OfferCode** - Der Angebotscode des Angebots.
- **OfferName** - Der Name des Angebots.
- **TreatmentCode** - Der Verfahrenscode des Angebots.
- **Score** - Der Marketing-Score des Angebots oder die durch `ScoreOverrideTable` definierte Bewertung, wenn die Eigenschaft `enableScoreOverrideLookup` true ist.

getAdditionalAttributes

Die `getAdditionalAttributes`-Methode gibt die benutzerdefinierten Angebotsattribute zurück, die in Unica Campaign definiert sind.

```
getAdditionalAttributes()
```

Rückgabewert

Die `getAdditionalAttributes`-Methode gibt ein Array aus NameValuePair-Objekten zurück.

Beispiel

Das folgende Beispiel zeigt, wie Sie alle zusätzlichen Attribute sortieren, das gültige Datum und das Ablaufdatum überprüfen und die weiteren Attribute ausdrucken können.

```
   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

Die `getDescription`-Methode gibt die Beschreibung des in Unica Campaign definierten Angebots zurück.

```
getDescription()
```

Rückgabewert

Die `getDescription`-Methode gibt eine Zeichenfolge zurück.

Beispiel

Im folgenden Beispiel wird die Beschreibung eines Angebots ausgedruckt.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
     System.out.println("Offer Description:"+offer.getDescription());
}
```

getOfferCode

Die `getTreatmentCode`-Methode gibt den Verfahrenscode des Angebots zurück, wie in Unica Campaign definiert.

```
getOfferCode()
```

Rückgabewert

Die `getOfferCode`-Methode gibt ein Array aus Zeichenfolgen zurück, die den Angebotscode des Angebots enthalten.

Beispiel

Im folgenden Beispiel wird der Angebotscode eines Angebots ausgedruckt.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
   System.out.println("Offer Code:"+offer.getOfferCode());
}
```

getOfferName

Die `getOfferName`-Methode gibt den in Unica Campaign definierten Namen des Angebots zurück.

```
getOfferName()
```

Rückgabewert

Die `getOfferName`-Methode gibt eine Zeichenfolge zurück.

Beispiel

Im folgenden Beispiel wird der Name eines Angebots gedruckt.

```
for(Offer offer : offerList.getRecommendedOffers())
{
// print offer
System.out.println("Offer Name:"+offer.getOfferName());
}
```

getScore

Die Methode `getScore` gibt eine Bewertung zurück, die auf den von Ihnen konfigurierten Angeboten basiert.

```
getScore()
```

Die `getScore`-Methode gibt eine der folgenden Optionen zurück:

- Wenn die Standardangebotstabelle, die Tabelle für die Bewertungsüberschreibung oder das integrierte Lernmodul nicht aktiviert ist, gibt diese Methode den auf der Registerkarte "Interaktionsstrategie" definierten Marketing-Score des Angebots zurück.
- Wenn Sie die Tabelle mit den Standardangeboten oder die Tabelle für die Bewertungsüberschreibung, nicht aber das integrierte Lernmodul aktiviert haben, gibt diese Methode die Bewertung des Angebots zurück, wie durch den Vorrang von Bedingungen zwischen der Tabelle mit den Standardangeboten, der Bewertung des Marketiers und der Tabelle für die Bewertungsüberschreibung definiert.
- Wenn Sie das integrierte Lernmodul aktiviert haben, gibt diese Methode die zum Sortieren der Angebote verwendete endgültige Bewertung zurück.

Rückgabewert

Die Methode `getScore` gibt eine Ganzzahl zurück, die die Bewertung des Angebots darstellt.

Beispiel

Im folgenden Beispiel wird die Punktzahl eines Angebots ausgedruckt.

```
for(Offer offer : offerList.getRecommendedOffers())
{
// print offer
System.out.println("Offer Score:"+offer.getOfferScore());
}
```

getTreatmentCode

Die `getTreatmentCode`-Methode gibt den Verfahrenscode des Angebots zurück, wie in Unica Campaign definiert.

```
getTreatmentCode()
```

Da Unica Campaign diesen Verfahrenscode zum Identifizieren der Instanz des erstellten Angebots verwendet, muss dieser Code als ein Ereignisparameter zurückgegeben werden, wenn die `postEvent`-Methode verwendet wird, um ein Kontakt-, Annahme- oder Ablehnungsereignis des Angebots zu protokollieren. Wenn Sie die Annahme oder Ablehnung eines Angebots protokollieren, müssen Sie den Namenswert im NameValuePair, das den Verfahrenscode darstellt, auf `UACIOfferTrackingCode`setzen.

Rückgabewert

Die `getTreatmentCode`-Methode gibt eine Zeichenfolge zurück.

Beispiel

Im folgenden Beispiel wird der Verfahrenscode eines Angebots ausgedruckt.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
    System.out.println("Offer Treatment Code:"+offer.getTreatmentCode());
}
```

Informationen zur Klasse OfferList

Die Klasse `OfferList` enthält Methoden, die die Ergebnisse der Methode `getOffers` definieren.

Das OfferList-Objekt enthält die folgenden Attribute:

- **DefaultString** - Die Standardzeichenfolge, die für den Interaktionspunkt im interaktiven Kanal definiert ist.
- **RecommendedOffers** - Ein Array der Angebotsobjekte, die von der Methode `getOffers` angefordert werden.

Die Klasse `OfferList` arbeitet mit Listen von Angeboten. Diese Klasse steht nicht mit Unica Campaign-Angebotslisten in Beziehung.

getDefaultString

Die `getDefaultString`-Methode gibt die Standardzeichenfolge für den Interaktionspunkt zurück, wie in Unica Campaign definiert.

```
getDefaultString()
```

Wenn das RecommendedOffers-Objekt leer ist, sollten Sie Ihren Touchpoint so konfigurieren, dass er diese Zeichenfolge darstellt, um sicherzustellen, dass einige Inhalte dargestellt werden.Unica Interact füllt das DefaultString-Objekt nur dann, wenn das RecommendedOffers-Objekt leer ist.

Rückgabewert

Die `getDefaultString`-Methode gibt eine Zeichenfolge zurück.

Beispiel

Das folgende Beispiel ruft die Standardzeichenfolge ab, wenn das offerList-Objekt keine Angebote enthält.

```
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

Die `getRecommendedOffers`-Methode gibt ein Array aus Offer-Objekten zurück, die von der `getOffers`-Methode angefordert wurden.

```
getRecommendedOffers()
```

Wenn die Antwort auf `getRecommendedOffer` leer ist, sollte der Touchpoint das `getDefaultString`-Ergebnis darstellen.

Rückgabewert

Die `getRecommendedOffers`-Methode gibt ein Offer-Objekt zurück.

Beispiel

Das folgende Beispiel verarbeitet das OfferList-Objekt und druckt die Namen aller empfohlenen Angebote aus.

```
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());
```

Informationen zur Klasse Response

Die Klasse `Response` enthält Methoden, die die Ergebnisse einer der InteractAPI-Klassenmethoden definieren.

Das Antwortobjekt enthält die folgenden Attribute:

- **AdvisoryMessages** - ein Array von Empfehlungsnachrichten. Dieses Attribut wird nur aufgefüllt, wenn es während der Ausführung der Methode Warnungen oder Fehler gab.
- **ApiVersion** - eine Zeichenfolge mit der API-Version. Dieses Attribut wird durch die Methode `getVersion` aufgefüllt.
- **OfferList** - das OfferList-Objekt, das die von der Methode `getOffers` angeforderten Angebote enthält.
- **ProfileRecord** - ein Array von Name/Wert-Paaren, das Profildaten enthält. Dieses Attribut wird durch die Methode `getProfile` aufgefüllt.
- **SessionID** - eine Zeichenfolge, die die Sitzungs-ID definiert. Dies wird von allen `InteractAPI`-Klassenmethoden zurückgegeben.
- **StatusCode** - eine Zahl, die angibt, ob die Methode ohne Fehler, mit einer Warnung oder mit Fehlern ausgeführt wurde. Dies wird von allen `InteractAPI`-Klassenmethoden zurückgegeben.

getAdvisoryMessages

Die `getAdvisoryMessages`-Methode gibt ein Array aus Advisory Messages aus dem Response-Objekt zurück.

```
getAdvisoryMessages()
```

Rückgabewert

Die `getAdvisoryMessages`-Methode gibt ein Array aus Advisory Message-Objekten zurück.

Beispiel

Das folgende Beispiel ruft die AdvisoryMessage-Objekte aus einem Response-Objekt ab und durchläuft diese, um die Nachrichten auszudrucken.

```
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

Die `getApiVersion` Methode gibt die API-Version eines Response-Objekts zurück.

```
getApiVersion()
```

Die `getVersion`-Methode füllt das ApiVersion-Attribut eines Response-Objekts aus.

Rückgabewert

Das Response-Objekt gibt eine Zeichenfolge zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die das Antwortobjekt für `getVersion` verarbeitet.

```
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

Die `getOfferList`-Methode gibt das OfferList-Objekt eines Response-Objekts zurück.

```
getOfferList()
```

Die `getOffers`-Methode füllt das OfferList-Objekt eines Response-Objekts aus.

Rückgabewert

Das Response-Objekt gibt ein OfferList-Objekt zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die das Antwortobjekt für `getOffers` verarbeitet.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
    // print offer
    System.out.println("Offer Name:"+offer.getOfferName());
    }
}
```

getAllOfferLists

Die `getAllOfferLists`-Methode gibt ein Array aus allen OfferLists eines Response-Objekts zurück.

```
getAllOfferLists()
```

Dies wird von der `getOffersForMultipleInteractionPoints`-Methode verwendet, die das OfferList-Arrayobjekt eines Response-Objekts ausfüllt.

Rückgabewert

Das Response-Objekt gibt ein OfferList-Array zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die das Antwortobjekt für `getOffers` verarbeitet.

```
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

Die `getProfileRecord`-Methode gibt die Profileinträge für die aktuelle Sitzung als ein Array aus NameValuePair-Objekten zurück. Diese Profileinträge beinhalten auch alle `eventParameters`, die zuvor in der Laufzeitsitzung hinzugefügt wurden.

```
getProfileRecord()
```

Die `getProfile`-Methode füllt die NameValuePair-Objekte für einen Profildatensatz eines Response-Objekts aus.

Rückgabewert

Das Response-Objekt gibt ein Array aus NameValuePair-Objekten zurück.

Beispiel

Das folgende Beispiel ist ein Auszug aus einer Methode, die das Antwortobjekt für `getOffers` verarbeitet.

```
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

Die `getSessionID`-Methode gibt eine Sitzungs-ID zurück.

```
getSessionID()
```

Rückgabewert

Die `getSessionID`-Methode gibt eine Zeichenfolge zurück.

Beispiel

Das folgende Beispiel zeigt eine Nachricht, die Sie am Anfang oder am Ende der Fehlerbehandlung anzeigen können, um anzugeben, auf welche Sitzung sich die Fehler beziehen.

```
System.out.println("This response pertains to
 sessionId:"+response.getSessionID());
```

getStatusCode

Die `getStatusCode`-Methode gibt den Statuscode eines Response-Objekts zurück.

```
getStatusCode()
```

Rückgabewert

Das Response-Objekt gibt eine Ganzzahl zurück.

- 0 - STATUS_SUCCESS - Die aufgerufene Methode wurde ohne Fehler ausgeführt. Möglicherweise sind Advisory Messages vorhanden.
- 1 - STATUS_WARNING - Die aufgerufene Methode wurde mit mindestens einer Warnung (aber ohne Fehler) ausgeführt. Weitere Informationen finden Sie in den Advisory Messages.
- 2 - STATUS_ERROR - Die aufgerufene Methode wurde nicht erfolgreich ausgeführt und hat mindestens eine Fehlernachricht. Weitere Informationen finden Sie in den Advisory Messages.

Beispiel

Das folgende Beispiel zeigt, wie Sie `getStatusCode` zur Fehlerbehandlung verwenden können.

```
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());
}
```

Kapitel 9. Klassen und Methoden für die Unica Interact-JavaScript-API

In den folgenden Abschnitten werden Anforderungen und andere Details aufgelistet, die Sie kennen sollten, bevor Sie mit der Unica Interact-JavaScript-API zu arbeiten beginnen.

Die Unica Interact-API unterstützt eine JavaScript-Version für die Kommunikation zwischen dem Endbenutzerclient (Browser) und dem Server.

Anmerkung: In diesem Abschnitt wird vorausgesetzt, dass Sie mit APIs vertraut sind, die auf JavaScript basieren.

Anmerkung: Die mehrmalige Verwendung eines Parameters in einem einzelnen API-Aufruf wird nicht unterstützt.

JavaScript-Voraussetzungen

Bevor Sie die Unica Interact-JavaScript-API auf einer Website verwenden, müssen Sie die Datei `interactapi.js` auf den Webseiten einschließen.

Arbeiten mit Sitzungsdaten

Wenn Sie eine Sitzung mit der Methode `startSession` initialisieren, werden Sitzungsdaten in den Speicher geladen. Während der Sitzung können Sie die Sitzungsdaten (die eine Obermenge der statischen Profildaten sind) lesen und schreiben.

Die Sitzung enthält die folgenden Daten:

- Statische Profildaten
- Segmentzuordnungen
- Echtzeitdaten
- Angebotsempfehlungen

Alle Sitzungsdaten sind bis zum Aufruf der Methode `endSession` bzw. bis zum Ablauf der `sessionTimeout`-Zeit verfügbar. Mit dem Ende der Sitzung gehen alle Daten verloren, die nicht ausdrücklich in den Kontakt- oder Antwortverlauf oder eine andere Datenbanktabelle gespeichert werden.

Die Daten werden als ein Satz von Name/Wert-Paaren gespeichert. Wenn die Daten aus der Datenbanktabelle gelesen werden, ist der Name die Spalte der Tabelle.

Sie können diese Name/Wert-Paare während der Arbeit mit der Unica Interact-API erstellen. Sie müssen nicht alle Name/Wert-Paare in einem Globalbereich deklarieren. Wenn Sie neue Ereignisparameter als Name/Wert-Paare festlegen, fügt die Laufzeitumgebung die Name/Wert-Paare den Sitzungsdaten hinzu. Wenn Sie beispielsweise Ereignisparameter mit der Methode `postEvent` verwenden, fügt die Laufzeitumgebung die Ereignisparameter den Sitzungsdaten hinzu, selbst wenn die Ereignisparameter nicht in den Profildaten verfügbar waren. Diese Daten existieren nur in den Sitzungsdaten.

Sie können Sitzungsdaten jederzeit überschreiben. Beispiel: Wenn ein Abschnitt des Kundenprofils `creditScore` umfasst, können Sie einen Ereignisparameter mithilfe des benutzerdefinierten Typs `NameValuePair` übergeben. In der Klasse `NameValuePair` können Sie die Methoden `setName` und `setValueAsNumeric` verwenden, um den Wert zu ändern. Der Name muss übereinstimmen. Innerhalb der Sitzungsdaten muss beim Namen die Groß-/Kleinschreibung nicht berücksichtigt zu werden. Daher würden die Namen `creditscore` oder `CrEdItScOrE` jeweils `creditScore` überschreiben.

Nur die letzten in die Sitzungsdaten geschriebenen Daten werden aufbewahrt. Beispiel: `startSession` lädt die Profildaten für den Wert `lastOffer`. Eine Methode `postEvent` überschreibt `lastOffer`. Dann überschreibt eine zweite Methode `postEvent` `lastOffer`. Die Laufzeitumgebung bewahrt nur die Daten, die von der zweiten Methode `postEvent` geschrieben wurden, in den Sitzungsdaten.

Wenn die Sitzung endet, gehen die Daten verloren, außer Sie haben besondere Vorkehrungen getroffen, wie z. B. die Verwendung eines Prozesses "Momentaufnahme" in Ihrem interaktiven Ablaufdiagramm, um die Daten in eine Datenbanktabelle zu schreiben. Wenn Sie vorhaben, Prozesse "Momentaufnahme" zu verwenden, achten Sie darauf, dass die Namen den Einschränkungen Ihrer Datenbank entsprechen müssen. Wenn Sie

beispielsweise nur 256 Zeichen für den Namen einer Spalte zulassen, sollte der Name für das Name-Wert-Paar 256 Zeichen nicht überschreiten.

Mit Callback-Parameter arbeiten

Die Callback-Funktion ist ein zusätzlicher Parameter jeder Methode der Unica Interact-JavaScript-API.

Der Hauptbrowserprozess ist eine aus einem Einzelthread bestehende Ereignisschleife. Wenn eine Operation mit langer Laufzeit in einer Ereignisschleife aus einem Einzelthread ausgeführt wird, wird der Prozess blockiert. So wird die Verarbeitung weiterer Ereignisse durch den Prozess blockiert, während er auf den Abschluss der Operation wartet.
Zur Vermeidung von Blockierungen bei Operationen mit langer Laufzeit wird von XMLHttpRequest eine asynchrone Schnittstelle bereitgestellt. Dabei wird ein Callback zur Ausführung nach Abschluss der Operation übergeben; während der Verarbeitung wird die Steuerung an die Hauptereignisschleife zurückgegeben und somit nicht der Prozess blockiert.

Wenn die Methode erfolgreich war, wird von der Callback-Funktion `onSuccess` aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Wenn Sie zum Beispiel Angebote auf Ihrer Webseite anzeigen möchten, würden Sie die Methode `getOffers` und den Callback zum Anzeigen auf der Seite verwenden. Die Webseite verhält sich normal und wartet nicht darauf, dass die Angebote von Unica Interact zurückgegeben werden. Wenn die Angebote stattdessen von Unica Interact zurückgegeben werden, wird die Antwort im Callback-Parameter zurückgesendet. Sie können die Callback-Daten auswerten und Angebote auf der Seite anzeigen.

Sie können einen generischen Callback für alle Funktionen oder auch spezifische Callbacks für bestimmte Funktionen verwenden.

Sie können `var callback = InteractAPI.Callback.create(onSuccess, onError);` zum Erstellen einer generischen Callback-Funktion verwenden.

Sie können die folgende Funktion zum Erstellen einer spezifischen Callback-Funktion für die Methode "getOffers" verwenden.

```
var callbackforGetOffer = InteractAPI.Callback.create(onSuccessofGetOffer,
onErrorofGetOffer);
```

Informationen zur Klasse InteractAPI

Die Klasse InteractAPI enthält die Methoden, die Sie verwenden, um Ihren Touchpoint in den Laufzeitserver zu integrieren. Alle anderen Klassen und Methoden in der Unica Interact-API unterstützen die Methoden in dieser Klasse.

Sie müssen Ihre Implementierung anhand von `interact_client.jar` im `lib`-Verzeichnis Ihrer Unica Interact-Laufzeitumgebungsinstallation kompilieren.

startSession

Die `startSession`-Methode erstellt und definiert eine Laufzeitsitzung.

```
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` kann bis zu fünf Aktionen auslösen:

- Erstellen der Laufzeitsitzung.
- Laden der Besucherprofildaten für die aktuelle Zielgruppenebene in der Laufzeitsitzung inklusive aller Dimensionstabellen, die in der für den interaktiven Kanal definierten Tabellenzuordnung zum Laden markiert sind.
- Auslösen der Segmentierung, indem alle interaktiven Ablaufdiagramme für die aktuelle Zielgruppenebene ausgeführt werden.
- Laden von Angebotsunterdrückungsdaten in der Sitzung, wenn die Eigenschaft `enableOfferSuppressionLookup` auf "true" gesetzt ist.
- Laden von Bewertungsüberschreibungsdaten in der Sitzung, wenn die Eigenschaft `enableScoreOverrideLookup` auf "true" gesetzt ist.

Die `startSession`-Methode benötigt die folgenden Parameter:

- **sessionID** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID. Sie müssen die Sitzungs-ID definieren. Sie können zum Beispiel eine Kombination aus Kunden-ID und Zeitmarke verwenden.

  Sie müssen eine Sitzungs-ID angeben, um zu definieren, was eine Laufzeitsitzung auszeichnet. Dieser Wert wird vom Client verwaltet. Alle Methodenaufrufe für die gleiche Sitzungs-ID müssen vom Client synchronisiert werden. Das Verhalten für gleichzeitige API-Aufrufe mit der gleichen Sitzungs-ID ist nicht definiert.
- **relyOnExistingSession** - ein boolescher Ausdruck, der definiert, ob diese Sitzung eine neue oder eine vorhandene Sitzung verwendet. Gültige Werte sind `true` oder `false`. Wenn der Wert `true` ist, müssen Sie eine vorhandene Sitzungs-ID mit der `startSession`-Methode angeben. Wenn der Wert `false` ist, müssen Sie eine neue Sitzungs-ID angeben.

  Wenn Sie `relyOnExistingSession` auf `true` setzen und eine Sitzung vorhanden ist, verwendet die Laufzeitumgebung die vorhandenen Sitzungsdaten. Es werden keine Daten erneut geladen und es wird keine Segmentierung ausgelöst. Wenn die Sitzung nicht vorhanden ist, erstellt die Laufzeitumgebung eine neue Sitzung inklusive Laden der Daten und Auslösen der Segmentierung. Wenn die Sitzungsdauer des Touchpoints

länger als die Laufzeitsitzung ist, kann es sinnvoll sein, `relyOnExistingSession` auf "true" zu setzen und mit allen `startSession`-Aufrufen zu verwenden. Beispiel: Die Sitzung einer Website ist 2 Stunden lang aktiv, während die Laufzeitsitzung nur 20 Minuten lang aktiv ist.

Wenn Sie `startSession` zweimal mit der gleichen Sitzungs-ID aufrufen, gehen alle Sitzungsdaten des ersten `startSession`-Aufrufs verloren, wenn `relyOnExistingSession` auf false gesetzt ist.

- **debug** - ein boolescher Ausdruck zum Aktivieren oder Inaktivieren von Daten zur Fehlerbehebung. Gültige Werte sind `true` oder `false`. Wenn der Wert true ist, protokolliert Unica Interact die Daten zur Fehlerbehebung in den Protokollen des Laufzeitservers. Das Debug-Flag wird für jede Sitzung individuell gesetzt. Somit können Sie die Daten zur Fehlerbehebung für einzelne Sitzungen verfolgen.
- **interactiveChannel** - eine Zeichenfolge, die den Namen des interaktiven Kanals definiert, auf den diese Sitzung verweist. Dieser Name muss exakt mit dem in Unica Campaign definierten Namen des interaktiven Kanals übereinstimmen.
- **audienceID** - ein Array aus NameValuePairImpl-Objekten, wobei die Namen mit den physischen Spaltennamen aller Tabellen übereinstimmen müssen, in denen die Zielgruppen-ID enthalten ist.
- **audienceLevel** - eine Zeichenfolge zum Definieren der Zielgruppenebene.
- **parameters** - NameValuePairImpl-Objekte zum Identifizieren aller Parameter, die mit `startSession` übergeben werden müssen. Diese Werte werden in den Sitzungsdaten gespeichert und können zur Segmentierung verwendet werden.

  Wenn Sie mehrere interaktive Ablaufdiagramme für die gleiche Zielgruppenebene haben, müssen Sie eine Obermenge mit allen Spalten in allen Tabellen einschließen. Wenn Sie die Laufzeit so konfigurieren, dass die Profiltabelle geladen wird, und die Profiltabelle alle benötigten Spalten enthält, ist es nicht erforderlich, Parameter zu übergeben, es sei denn, Sie möchten die Daten in der Profiltabelle überschreiben. Wenn die Profiltabelle eine Untermenge der benötigten Spalten enthält, müssen Sie die fehlenden Spalten als Parameter einschließen.
- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Wenn `audienceID` oder `audienceLevel` ungültig und `relyOnExistingSession` false ist, schlägt der `startSession`-Aufruf fehl. Wenn `interactiveChannel` ungültig ist, schlägt `startSession` fehl, unabhängig davon, ob `relyOnExistingSession` true oder false ist.

Wenn `relyOnExistingSession` true ist und Sie einen zweiten `startSession`-Aufruf mit der gleichen `sessionID` durchführen, nachdem die erste Sitzung bereits abgelaufen ist, erstellt Unica Interact eine neue Sitzung.

Wenn `relyOnExistingSession` true ist und Sie einen zweiten `startSession`-Aufruf mit der gleichen `sessionID`, aber mit einer anderen `audienceID` oder `audienceLevel` durchführen, ändert der Laufzeitserver die Zielgruppe für die vorhandene Sitzung.

Wenn `relyOnExistingSession` true ist und Sie einen zweiten `startSession`-Aufruf mit der gleichen `sessionID`, aber mit einem anderen `interactiveChannel` durchführen, erstellt der Laufzeitserver eine neue Sitzung.

Rückgabewert

Der Laufzeitserver beantwortet `startSession` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages (wenn StatusCode nicht 0 ist)
- ApiVersion
- SessionID
- StatusCode

Angebotsdeduplizierung über Angebotsattribute

Unter Verwendung der Anwendungsprogrammierschnittstelle (Application Programming Interface, API) von Unica Interact werden mit zwei API-Aufrufen Angebote bereitgestellt: `getOffers` und `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` kann die Rückgabe duplizierter Angebote auf der Ebene OfferID verhindern, kann jedoch keine Angebote über eine Angebotskategorie deduplizieren. Daher war z. B. vorher bei Unica Interact für eine Rückgabe von nur einem Angebot aus den einzelnen Angebotskategorien eine Problemumgehung erforderlich. Durch die Einführung von zwei Parametern im API-Aufruf `startSession` sind Angebotsdeduplizierungen über Angebotsattribute, wie z. B. die Kategorie, jetzt möglich.

In dieser Liste finden Sie eine Übersicht über die Parameter, die dem API-Aufruf `startSession` hinzugefügt wurden. Sie finden weitere Informationen zu diesen Parametern oder zu sonstigen Aspekten der Unica Interact-API im Unica Interact-Administratorhandbuch sowie in den Javadoc-Dateien, die in Ihrer Unica Interact-Installation enthalten sind. Sie finden diese Dateien im Pfad `<Unica Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Wenn Sie einen `startSession`-API-Aufruf mit Angebotsdeduplizierung erstellen möchten, damit nachfolgende `getOffer`-Aufrufe immer nur jeweils ein Angebot aus jeder Kategorie zurückgeben, müssen Sie den Parameter `UACIOfferDedupeAttribute` als Bestandteil des API-Aufrufs einschließen. Sie können einen Parameter wie folgt im Format `name,value,type` angeben:

  `UACIOfferDedupeAttribute,<attributeName>,string`

  In diesem Beispiel würden Sie <attributeName> durch den Namen des Angebotsattributs ersetzen, das Sie als Kriterium für die Deduplizierung verwenden möchten (zum Beispiel "Category" (Kategorie)).

  Anmerkung: Unica Interact überprüft die Angebote, die den von Ihnen angegebenen Attributwert (zum Beispiel "Category") aufweisen. Anschließend wird eine Deduplizierung durchgeführt, damit mit Ausnahme des Angebots, das die höchste Bewertung aufweist, alle Angebote entfernt werden. Wenn die Angebote mit einem Attributduplikat auch identische Bewertungen aufweisen, trifft Unica Interact eine Zufallsauswahl unter den übereinstimmenden Angeboten und gibt diese zurück.

- `UACINoAttributeDedupeIfFewerOffer`. Wenn Sie den Parameter `UACIOfferDedupeAttribute` im `startSession`-Aufruf einschließen, können Sie auch den Parameter `UACINoAttributeDedupeIfFewerOffer` festlegen. Damit geben Sie an, wie sich das Programm verhalten soll, wenn die Angebotsliste nach der Deduplizierung nicht genügend Angebote enthält, damit die ursprüngliche Anforderung erfüllt werden kann.

Wenn Sie beispielsweise für `UACIOfferDedupeAttribute` die Verwendung der Angebotskategorie zur Deduplizierung von Angeboten festlegen und Ihr nachfolgender getOffers-Aufruf die Rückgabe von acht Angeboten anfordert, kann es sein, dass aufgrund der Deduplizierung weniger als acht auswählbare Angebote verfügbar sind. Wenn Sie den Parameter `UACINoAttributeDedupeIfFewerOffer` auf "true" setzen, werden in diesem Fall einige der deduplizierten Angebote zur Kandidatenliste hinzugefügt, damit die angeforderte Anzahl der Angebote erfüllt wird. Wenn Sie in diesem Beispiel den Parameter auf "false" setzen, wird die angeforderte Anzahl nicht durch die zurückgegebene Anzahl an Angeboten erreicht.

`UACINoAttributeDedupeIfFewerOffer` Die Standardeinstellung ist "true".

Angenommen, Sie haben wie folgt als Parameter für `startSession` die Angebotskategorie (Category) als Deduplizierungskriterium angegeben:

```
UACIOfferDedupeAttribute,Category,string;
```

```
UACINoAttributeDedupeIfFewerOffer,1,string
```

Standardmäßig dedupliziert `UACIOfferDedupeAttribute` keine Angebote, wenn weniger als die angeforderte Anzahl Angebote zurückgegeben werden. Der Parameter `UACINoAttributeDedupeIfFewerOffer` muss jedoch angegeben und auf 1 gesetzt werden, um sicherzustellen, dass die Deduplizierung stattfindet, wenn weniger als die angeforderten Angebote zurückgegeben werden.

Diese Parameterkombination bewirkt, dass Unica Interact Angebote auf Basis des Angebotsattributs "Category" dedupliziert und selbst dann nur die deduplizierten Angebote zurückgibt, wenn die resultierende Anzahl der Angebote die angeforderte Anzahl unterschreitet (da der Parameter `UACINoAttributeDedupeIfFewerOffer` auf "false" gesetzt ist).

Wenn Sie einen `getOffers`-API-Aufruf ausgeben, könnte die ursprüngliche Gruppe der auswählbaren Angebote die folgenden Angebote enthalten:

- Category=Electronics: Angebot A1 mit einer Bewertung von 100 und Angebot A2 mit einer Bewertung von 50.
- Category=Smartphones: Angebot B1 mit einer Bewertung von 100, Angebot B2 mit einer Bewertung von 80 und Angebot B3 mit einer Bewertung von 50.
- Category=MP3Players: Angebot C1 mit einer Bewertung von 100, Angebot C2 mit einer Bewertung von 50.

In diesem Fall gab es zwei doppelte Angebote, die mit der ersten Kategorie übereinstimmen, drei doppelte Angebote, die mit der zweiten Kategorie übereinstimmen, und zwei doppelte Angebote, die mit der dritten Kategorie übereinstimmen. Als Angebote werden die Angebote mit der höchsten Bewertung aus jeder Kategorie zurückgegeben, also Angebot A1, Angebot B1 und Angebot C1.

Obwohl der `getOffers`-API-Aufruf sechs Angebote angefordert hat, werden nur drei Angebote zurückgegeben, da der Parameter `UACINoAttributeDedupeIfFewerOffer` in diesem Beispiel auf "false" gesetzt ist.

Wenn der `getOffers`-API-Aufruf sechs Angebote angefordert hat und in diesem Beispiel keine Angabe für den Parameter `UACINoAttributeDedupeIfFewerOffer` gemacht oder der Parameter explizit auf "true" gesetzt wurde, werden einige der doppelten Angebote in das Ergebnis aufgenommen, damit die angeforderte Anzahl erfüllt wird.

postEvent

Mit der `postEvent`-Methode können Sie jedes Ereignis ausführen, das im interaktiven Kanal definiert ist.

```
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**: - eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **eventName**: eine Zeichenfolge zur Identifizierung des Ereignisnamens.

  Anmerkung: Der Name des Ereignisses muss mit dem im interaktiven Kanal definierten Ereignisnamen übereinstimmen. Bei diesem Namen braucht die Groß-/Kleinschreibung nicht berücksichtigt zu werden.

- **eventParameters**. `NameValuePairImpl`-Objekte, die alle Parameter identifizieren, die mit dem Ereignis übergeben werden müssen. Diese Werte werden in den Sitzungsdaten gespeichert.

  Wenn dieses Ereignis eine erneute Segmentierung auslöst, müssen Sie sicherstellen, dass alle vom interaktiven Ablaufdiagramm benötigten Daten in den Sitzungsdaten verfügbar sind. Wenn diese Werte noch nicht durch vorangegangene Aktionen ausgefüllt wurden (zum Beispiel durch `startSession`, durch `setAudience` oder beim Laden der Profiltabelle), müssen Sie für jeden fehlenden Wert einen eventParameter einschließen. Wenn Sie zum Beispiel alle Profiltabellen so konfiguriert haben, dass diese im Hauptspeicher geladen werden, müssen Sie für alle temporären Daten, die für interaktive Ablaufdiagramme erforderlich sind, jeweils ein NameValuePair einschließen.

  Wenn Sie mehrere Zielgruppenebenen verwenden, haben Sie vermutlich verschiedene Sätze an eventParameters für jede Zielgruppenebene. Sie sollten daher eine entsprechende Logik einschließen, die gewährleistet, dass für jede Zielgruppenebene immer der richtige Parametersatz ausgewählt wird.

  Wichtig: Wenn dieses Ereignis den Antwortverlauf protokolliert, müssen Sie den Verfahrenscode für das Angebot übergeben. Sie müssen den Namen für das NameValuePair als `UACIOfferTrackingCode` definieren.

Pro Ereignis können Sie immer nur einen Verfahrenscode übergeben. Wenn Sie den Verfahrenscode für einen Angebotskontakt nicht übergeben, protokolliert Unica Interact jeweils einen Angebotskontakt für jedes Angebot in der zuletzt empfohlenen Angebotsliste. Wenn Sie den Verfahrenscode für eine Antwort nicht übergeben, gibt Unica Interact einen Fehler zurück.

- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.
- Es gibt eine Reihe weiterer reservierter Parameter, die Sie mit `postEvent` und anderen Methoden verwenden können, die später in diesem Abschnitt erläutert werden.

Wenn Sie eine erneute Segmentierung anfordern oder in den Kontakt- oder Antwortverlauf schreiben, wird nicht auf eine Antwort gewartet.

Im Verlauf einer neuen Segmentierung werden nicht frühere Segmentierungsergebnisse für die aktuelle Zielgruppenebene bereinigt. Sie können den Parameter `UACIExecuteFlowchartByName` zum Definieren bestimmter Ablaufdiagramme verwenden, die ausgeführt werden sollen. Die `getOffers`-Methode wartet, bis die erneute Segmentierung abgeschlossen ist, und wird erst danach ausgeführt. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `postEvent`-Methode aufrufen, die sofort eine erneute Segmentierung auslöst, bevor ein `getOffers`-Aufruf erfolgt.

Rückgabewert

Der Laufzeitserver beantwortet `postEvent` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profil
- SessionID
- StatusCode

getOffers

Mit der `getOffers`-Methode können Sie Angebote vom Laufzeitserver anfordern.

```
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);

}
```

- **sessionID** - Eindeutige Zeichenfolge zur Identifizierung der Sitzung.
- **interactionPoint** - eine Zeichenfolge, die den Namen des Interaktionspunkts angibt, auf den diese Methode verweist.

  Anmerkung: Dieser Name muss exakt mit dem Namen des im interaktiven Kanal definierten Interaktionspunkts übereinstimmen.

- **nofRequested** - eine Ganzzahl, die die Anzahl der angeforderten Angebote angibt.
- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Bevor die `getOffers`-Methode ausgeführt wird, wartet sie so viele Millisekunden, wie in der `segmentationMaxWaitTimeInMS`-Eigenschaft angegeben sind, um die erneute Segmentierung abzuschließen. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `postEvent`-Methode aufrufen, die sofort eine erneute Segmentierung oder eine `setAudience`-Methode auslöst, bevor ein `getOffers`-Aufruf erfolgt.

Rückgabewert

Der Laufzeitserver beantwortet `getOffers` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profil
- SessionID
- StatusCode
- NameValuePair

Dezimalstellen in Angebotsbewertungen werden in der getOffer-Antwort im NameValue-Paar zurückgegeben. Wenn Angebote an die anfordernden Eingangskanäle zurückgegeben werden, verwenden die Kanäle die Bewertungen, um die Angebote zu priorisieren. Die Dezimalziffern werden nicht entfernt, sodass der Sender weiß, welches Angebot eine höhere Punktzahl hat, falls Dezimalzahlen zurückgegeben werden.

getOffersForMultipleInteractionPoints

Mit der `getOffersForMultipleInteractionPoints`-Methode können Sie Angebote vom Laufzeitserver für mehrere IPs mit Deduplizierung anfordern.

```
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);

}
```

- **sessionID** - Eine Zeichenfolge zur Identifizierung der Sitzung.
- **requestDetailsStr** - eine Zeichenfolge, die ein Array aus `GetOfferRequest`-Objekten angibt.

  Jedes `GetOfferRequest`-Objekt legt fest:

  - **ipName** - Der Name des Interaktionspunkts (IP), für den das Objekt Angebote anfordert
  - **numberRequested** - Die Anzahl an eindeutigen Angeboten, die für den angegebenen IP erforderlich ist
  - **offerAttributes** - Anforderungen an die Attribute der gelieferten Angebote mit einer Instanz von `OfferAttributeRequirements`
  - **duplicationPolicy** - Duplizierungsrichtlinien-ID für die Angebote, die geliefert werden

Duplizierungsrichtlinien bestimmen, ob doppelte Angebote an verschiedenen Interaktionspunkten in einem einzigen Methodenaufruf zurückgegeben werden. (Innerhalb eines einzelnen Interaktionspunktes werden doppelte Angebote niemals zurückgegeben). Derzeit werden zwei Duplizierungsrichtlinien unterstützt.

- NO_DUPLICATION (ID-Wert = 1). Keines der Angebote, die in den vorangegangenen `GetOfferRequest`-Instanzen enthalten waren, wird in diese `GetOfferRequest`-Instanz einbezogen (das heißt, Unica Interact wendet die Deduplizierung an).
- ALLOW_DUPLICATION (ID-Wert = 2). Alle Angebote, die die Voraussetzungen erfüllen, die in dieser `GetOfferRequest`-Instanz angegeben sind, werden einbezogen. Es findet kein Abgleich der Angebote statt, die in den vorangegangenen `GetOfferRequest`-Instanzen enthalten waren.

◦ **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Die Reihenfolge der Anfragen im Array-Parameter ist auch die Reihenfolge der Priorität, in der die Angebote geliefert werden.

Beispiel: Angenommen, die IPs in der Anfrage heißen IP1 und IP2, duplizierte Angebote sind unzulässig (mit der Duplizierungsrichtlinien-ID = 1) und jeder IP fordert zwei Angebote an. Wenn Unica Interact die Angebote A, B und C für IP1 und die Angebote A und D für IP2 findet, enthält die Antwort die Angebote A und B für IP1 und nur das Angebot D für IP2.

Zusätzlicher Hinweis: Wenn die Duplizierungsrichtlinien-ID 1 lautet, werden Angebote, die über einen IP mit hoher Priorität geliefert wurden, nicht über diesen IP geliefert.

Bevor die `getOffersForMultipleInteractionPoints`-Methode ausgeführt wird, wartet sie so viele Millisekunden, wie in der `segmentationMaxWaitTimeInMS`-Eigenschaft angegeben sind, um die erneute Segmentierung abzuschließen. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `postEvent`-Methode aufrufen, die sofort eine erneute

Segmentierung oder eine `setAudience`-Methode auslöst, bevor ein `getOffers`-Aufruf erfolgt.

Rückgabewert

Der Laufzeitserver beantwortet `getOffersForMultipleInteractionPoints` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- Array von OfferList
- Profil
- SessionID
- StatusCode

setAudience

Mit der `setAudience`-Methode können Sie die Zielgruppen-ID und die Zielgruppenebene für Besucher festlegen.

```
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** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **audienceID** - ein Array von `NameValuePairImpl`-Objekten zum Definieren der Zielgruppen-ID.

- **audienceLevel** - eine Zeichenfolge zum Definieren der Zielgruppenebene.
- **parameters** - `NameValuePairImpl`-Objekte zum Identifizieren aller Parameter, die mit `setAudience` übergeben werden müssen. Diese Werte werden in den Sitzungsdaten gespeichert und können zur Segmentierung verwendet werden.

  Sie benötigen für jede Spalte in Ihrem Profil einen Wert. Dies ist eine Obermenge aus allen Spalten in den Echtzeitdaten und in allen Tabellen, die für den interaktiven Kanal definiert sind. Wenn Sie bereits alle Sitzungsdaten mit `startSession` oder `postEvent` ausgefüllt haben, ist es nicht erforderlich, neue Parameter zu senden.
- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Die `setAudience`-Methode löst eine erneute Segmentierung aus. Die `getOffers`-Methode wartet, bis die erneute Segmentierung abgeschlossen ist, und wird erst danach ausgeführt. Daher kann es zu einer Verzögerung kommen, wenn Sie eine `setAudience`-Methode aufrufen, bevor ein `getOffers`-Aufruf erfolgt.

Die `setAudience`-Methode lädt auch die Profildaten für die Zielgruppen-ID. Mit der `setAudience`-Methode können Sie erzwingen, dass erneut die gleichen Profildaten geladen werden wie mit der `startSession`-Methode.

Die `setAudience`-Methode lädt die Whitelist- und Blacklist-Tabelle in einer vorhandenen Sitzung erneut. Sie können die `setAudience`-Methode mit den Parametern `UACIPurgePriorWhiteListOnLoad` und `UACIPurgePriorBlackListOnLoad` verwenden, um die Whitelist- und Blacklist-Tabelle in einer vorhandenen Sitzung erneut zu laden.

Standardmäßig wird der gesamte Inhalt der Blacklist entfernt, wenn die `setAudience`-Methode aufgerufen wird. Sie können die Parameter `UACIPurgePriorWhiteListOnLoad` und `UACIPurgePriorBlackListOnLoad` im `setAudience`-Aufruf wie folgt einstellen:

- Wenn Sie `UACIPurgePriorBlackListOnLoad= 0` festlegen, wird der gesamte Inhalt der Whitelist-Tabelle beibehalten.
- Wenn Sie `UACIPurgePriorWhiteListOnLoad= 1` festlegen, wird der Inhalt der Tabelle entfernt und der Inhalt der Whitelist oder Blacklist für die Zielgruppen-ID wird aus der Datenbank geladen. Nach dem Abschluss wird die erneute Segmentierung gestartet.

Rückgabewert

Der Laufzeitserver beantwortet `setAudience` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profil
- SessionID
- StatusCode

getProfile

Mit der `getProfile`-Methode können Sie Profildaten und temporäre Informationen über die Besucher des Touchpoints abrufen.

```
function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        InteractAPI.getProfile(ssId, callback);

}
```

- **sessionID** - Eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Rückgabewert

Der Laufzeitserver beantwortet `getProfile` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- OfferList
- ProfileRecord
- SessionID
- StatusCode

endSession

Die `endSession`-Methode markiert das Ende der Laufzeitsitzung. Wenn der Laufzeitserver diese Methode empfängt, wird der Verlauf protokolliert und der Speicher gelöscht.

```
function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        InteractAPI.endSession(ssId, callback);

}
```

- **sessionID** - Eindeutige Zeichenfolge zur Identifizierung der Sitzung.
- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Zeitlimitüberschreitung der Laufzeitsitzungen, wenn die endSession-Methode nicht aufgerufen wird. Das Zeitlimitintervall ist mit der `sessionTimeout`-Eigenschaft konfigurierbar.

Rückgabewert

Der Laufzeitserver beantwortet die `endSession`-Methode mit dem `Response`-Objekt, das die folgenden Attribute enthält:

- SessionID
- ApiVersion
- OfferList
- Profil
- StatusCode
- AdvisoryMessages

setDebug

Mit der `setDebug`-Methode können Sie den Detaillierungsgrad der Protokollierung für alle Codepfade für die Sitzung festlegen.

```
function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        InteractAPI.setDebug(ssId, isDebug, callback);

}
```

- **sessionID** - eine Zeichenfolge zur Identifizierung der Sitzungs-ID.
- **debug** - ein boolescher Ausdruck zum Aktivieren oder Inaktivieren von Daten zur Fehlerbehebung. Gültige Werte sind `true` oder `false`. Wenn der Wert wahr ist, protokolliert Unica Interact die Daten zur Fehlerbehebung im Protokoll des Laufzeitservers.
- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Rückgabewert

Der Laufzeitserver beantwortet `setDebug` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profil
- SessionID
- StatusCode

getVersion

Die `getVersion`-Methode gibt die Version der aktuellen Implementierung des Unica Interact Laufzeitservers zurück.

```
function callGetVersion(commandsToExecute, callback) {

        InteractAPI.getVersion(callback);



}
```

Es empfiehlt sich, diese Methode zu verwenden, wenn Sie den Touchpoint mit dem Unica Interact API initialisieren.

- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Rückgabewert

Der Laufzeitserver beantwortet `getVersion` mit einem Response-Objekt, das die folgenden Attribute enthält:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profil
- SessionID
- StatusCode

executeBatch

Mit der `executeBatch`-Methode können Sie mehrere Methoden mit einer einzelnen Anfrage an den Laufzeitserver ausführen.

```
function callExecuteBatch(commandsToExecute, callback) {

        if (!commandsToExecute)
            return ;

        InteractAPI.executeBatch(commandsToExecute.ssid,
        commandsToExecute.commands, callback);
}
```

- **sessionID** - Eindeutige Zeichenfolge zur Identifizierung der Sitzung. Diese Sitzungs-ID wird für alle Befehle verwendet, die dieser Methodenaufruf ausführt.
- **commands** - Ein Array aus Befehlsobjekten, jeweils eines für jeden Befehl, der ausgeführt werden soll.
- **callback** - Wenn die Methode erfolgreich war, wird von der `Callback`-Funktion onSuccess aufgerufen. Wenn die Methode nicht erfolgreich war, wird von der Callback-Funktion `onError` aufgerufen.

Durch den Aufruf dieser Methode wird das gleiche Ergebnis erzielt wie durch den expliziten Aufruf jeder einzelnen Methode im Befehl-Array. Diese Methode minimiert die Anzahl der tatsächlichen Anfragen an den Laufzeitserver. Der Laufzeitserver führt jede Methode seriell aus. Für jeden Aufruf werden alle Fehler oder Warnungen im entsprechenden

Response-Objekt für diesen Methodenaufruf aufgezeichnet. Wird ein Fehler gefunden, wird executeBatch mit den verbliebenen Aufrufen im Stapel fortgesetzt. Wenn der Aufruf einer beliebigen Methode in einem Fehler resultiert, wird dieser Fehler im Status auf der höchsten Ebene für das BatchResponse-Objekt angezeigt. Wenn keine Fehler aufgetreten sind, werden im Status auf der höchsten Ebene alle aufgetretenen Warnungen angezeigt. Wenn keine Warnungen aufgetreten sind, wird im Status auf der höchsten Ebene die erfolgreiche Ausführung des Stapels angezeigt.

Rückgabewert

Der Laufzeitserver beantwortet den `executeBatch` mit einem BatchResponse-Objekt.

Beispiel für JavaScript-API

```
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, getNameValuePairs(params)));
        }
        else {
            InteractAPI.postEvent(ssId, ev, getNameValuePairs(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) {

    }
```

Beispiel für JavaScript-Antwortobjekt "onSuccesss"

In diesem Beispiel werden drei Variablen für das JavaScript-Antwortobjekt erläutert: "offerLists", "messages" und "profile".

`offerList` gibt eine Liste ohne null zurück, wenn Sie `getOffer` oder `getOffersForMultipleInteractionPoints` als API oder als Bestandteil von Batchbefehlen

aufrufen. Sie müssen für dieses Objekt immer null überprüfen, bevor Sie eine Operation für diese Variable ausführen.

Sie sollten immer den Status der JavaScript-Antwort `messages` überprüfen.

`Profile` wird nicht null zurückgegeben, wenn Sie `getProfile` als API oder als Bestandteil von Batchbefehlen verwenden. Wenn Sie `getProfile` nicht verwenden, können Sie diese Variable ignorieren. Sie müssen für dieses Objekt immer null überprüfen, bevor Sie eine Operation für diese Variable ausführen.

```
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) {
            profile = [];
            for (var nvpCt=0; nvpCt<response.profile.length;nvpCt++) {
                var nvpObj =
this._buildNameValuePair(response.profile[nvpCt]);
                if (nvpObj) profile.push(nvpObj);
            }
        }

        return InteractAPI.Response.create(response.sessionId,
                                           response.statusCode, offerList,
                                           profile, response.version,
                                           messages) ;
    };
```

Kapitel 10. Informationen zur ExternalCallout-API

Unica Interact stellt ein erweiterbares Makro, EXTERNALCALLOUT, für die Verwendung mit Ihren interaktiven Ablaufdiagrammen bereit. Dieses Makro ermöglicht es Ihnen, angepasste Logik auszuführen, um mit externen Systemen während Ablaufdiagrammausführungen zu kommunizieren. Beispiel: Wenn Sie die Kreditbewertung eines Kunden während einer Ablaufdiagrammausführung berechnen wollen, können Sie eine Java™-Klasse (ein Callout) dafür erstellen und dann das Makro EXTERNALCALLOUT in einem Select-Prozess in Ihrem interaktiven Ablaufdiagramm verwenden, um die Kreditbewertung von Ihrem Callout abzurufen.

Das Konfigurieren von EXTERNALCALLOUT besteht hauptsächlich aus zwei Schritten. Erstens müssen Sie eine Java™-Klasse erstellen, die die ExternalCallout-API implementiert. Zweitens müssen Sie die notwendigen Unica Platform-Konfigurationseigenschaften auf dem Laufzeitserver in der Kategorie `Interact | Ablaufdiagramm | ExternalCallouts` konfigurieren.

Zusätzlich zu den Informationen in diesem Abschnitt steht die JavaDoc für die ExternalCallout-API auf jedem Unica Interact-Laufzeitserver im Verzeichnis `Interact/docs/externalCalloutJavaDoc` zur Verfügung.

IAffiniumExternalCallout-Benutzeroberfläche

Die ExternalCallout-API ist in der Benutzeroberfläche `IAffiniumExternalCallout` enthalten. Sie müssen die Benutzeroberfläche `IAffiniumExternalCallout` implementieren, um das Makro EXTERNALCALLOUT zu verwenden.

Die Klasse, die `IAffiniumExternalCallout` implementiert, sollte einen Konstruktor haben, mit dem sie durch den Laufzeitserver initialisiert werden kann.

- Wenn es keine Konstruktoren in der Klasse gibt, erstellt der Java™-Compiler einen Standardkonstruktor, was ausreichend ist.
- Wenn es Konstruktoren mit Argumenten gibt, sollte ein öffentlicher Konstruktor ohne Argument bereitgestellt werden, der vom Laufzeitserver verwendet wird.

Bei der Entwicklung Ihres externen Callouts beachten Sie Folgendes:

- Jede Ausdrucksauswertung mit einem externen Callout erstellt eine neue Instanz der Klasse. Sie müssen Probleme mit der Threadsicherheit für statische Mitglieder in der Klasse steuern.
- Wenn Ihr externes Callout Systemressourcen wie Dateien oder eine Datenbankverbindung verwendet, müssen Sie diese Verbindungen verwalten. Der Laufzeitserver hat keine Funktion, um Verbindungen automatisch zu bereinigen.

Sie müssen Ihre Implementierung anhand von `interact_externalcallout.jar` im `lib`-Verzeichnis Ihrer Unica Interact-Laufzeitumgebungsinstallation kompilieren.

`IAffiniumExternalCallout` ermöglicht es dem Laufzeitserver, Daten aus Ihrer Java™ Klasse anzufordern. Die Benutzeroberfläche besteht aus vier Methoden:

- `getNumberOfArguments`
- `getValue`
- `initialize`
- `shutdown`

Hinzufügen eines Web-Service zur Verwendung mit dem Makro EXTERNALCALLOUT

Mit diesem Verfahren fügen Sie einen Web-Service zur Verwendung mit dem Makro `EXTERNALCALLOUT` hinzu. Das Makro `EXTERNALCALLOUT` erkennt Callouts nur, wenn Sie die entsprechenden Konfigurationseigenschaften definiert haben.

Fügen Sie in Unica Platform für die Laufzeitumgebung folgende Konfigurationseigenschaften in der Kategorie `Interact > Ablaufdiagramm > externalCallouts` hinzu oder definieren Sie sie.

Konfigurationseigenschaft	Einstellung
Kategorie `externalCall-outs`	Erstellen Sie eine Kategorie für Ihr externes Callout

Konfigurationseigenschaft	Einstellung
`Klasse`	Die Klassennamen für Ihr externes Callout
`classpath`	Der Klassenpfad zu den Klassendateien Ihres externen Callouts
Kategorie `Parameter Data`	Wenn Ihr externes Callout Parameter erfordert, erstellen Sie neue Parameterkonfigurationseigenschaften für sie und weisen Sie jeder einen `Wert` zu

getNumberOfArguments

Interact ermöglicht die Übergabe einer variablen Anzahl von Argumenten an Ihr externes Callout. Die Methode getNumberOfArguments muss -1 zurückgeben, um eine variable Anzahl von Argumenten zuzulassen. Die Methode `getNumberOfArguments` gibt die Anzahl an Argumenten zurück, die von der zur Integration verwendeten Java™ Klasse erwartet wird.

```
getNumberOfArguments()
```

Rückgabewert

Die getNumberOfArguments-Methode gibt eine Ganzzahl zurück.

Beispiel

Das folgende Beispiel zeigt das Drucken der Anzahl der Argumente.

```
public int getNumberOfArguments() { return 0; }
```

getValue

Die `getValue`-Methode führt die zentralen Funktionen des Aufrufs durch und gibt die Ergebnisse zurück.

```
getValue(audienceID, configData, arguments)
```

Die `getValue`-Methode benötigt die folgenden Parameter:

- **audienceID** - ein Wert, der die Zielgruppen-ID angibt.
- **configData** - eine Zuordnung mit Schlüssel/Wert-Paaren von Konfigurationsdaten, die der Aufruf benötigt.
- **arguments** - die Argumente, die der Aufruf benötigt. Jedes Argument kann eine Zeichenfolge, ein Doppelzeichen, ein Datum oder eine Liste daraus sein. Ein Listenargument kann Nullwerte enthalten, aber eine Liste kann zum Beispiel nicht eine Zeichenfolge und ein Doppelzeichen enthalten.

  Sie sollten den Argumenttyp innerhalb der Implementierung überprüfen.

Wenn die `getValue`-Methode aus einem beliebigen Grund fehlschlägt, wird `CalloutException` zurückgegeben.

Rückgabewert

Die getValue-Methode gibt eine Liste mit Zeichenfolgen zurück.

Beispiel

```
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;
}
```

Initialisieren

Die Methode `initialize` wird beim Start des Laufzeitservers einmal aufgerufen. Alle eventuell vorhandenen Operationen, die die Leistung während der Laufzeit beeinträchtigen

können, sollten durch diese Methode durchgeführt werden, zum Beispiel das Laden einer Datenbanktabelle.

```
initialize(configData)
```

Für die Methode `initialize` ist der folgende Parameter erforderlich:

- **configData** - eine Zuordnung mit Schlüssel/Wert-Paaren von Konfigurationsdaten, die der Aufruf benötigt.

  Unica Interact liest diese Werte aus den Parametern, die in der Kategorie `Interact > Flowchart > External Callouts > [External Callout] > Parameter Data` für das externe Callout definiert sind.

Wenn die Methode `initialize` aus irgendeinem Grund fehlschlägt, wird eine `CalloutException` ausgelöst.

Rückgabewert

Keine.

Beispiel

```
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

Die `shutdown`-Methode wird beim Beenden des Laufzeitservers einmal aufgerufen. Alle eventuell erforderlichen Bereinigungsaufgaben sollten zu diesem Zeitpunkt ausgeführt werden.

```
shutdown(configData)
```

Die Methode `shutdown` erfordert die folgenden Parameter.

- **configData** - eine Zuordnung mit Schlüssel/Wert-Paaren von Konfigurationsdaten, die der Aufruf benötigt.

Wenn die Methode `shutdown` aus irgendeinem Grund fehlschlägt, wird eine `CalloutException` ausgelöst.

Rückgabewert

Keine.

Beispiel

```
public void shutdown(Map<String, String> configurationData) throws
 CalloutException
{
    // shutdown scoreQueryUtility here
}
```

Beispiel für die ExternalCallout-API

In diesem Beispiel wird ein externes Callout erstellt, das eine Kreditbewertung abruft.

Erstellen Sie ein externes Callout, das eine Kreditbewertung abruft:

1. Erstellen Sie eine Datei namens `GetCreditScore.java` mit den folgenden Inhalten. Diese Datei setzt voraus, dass es eine Klasse namens `ScoreQueryUtility` gibt, die eine Bewertung aus einer Modellanwendung abruft.

```
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.CalloutExceptio
n;
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. Kompilieren Sie `GetCreditScore.java` zu `GetCreditScore.class`.
3. Erstellen Sie eine JAR-Datei namens `creditscore.jar`, die `GetCreditScore.class` und die anderen verwendeten Klassendateien enthält.
4. Kopieren Sie die JAR-Datei an eine Position auf dem Laufzeitserver, z. B. `/data/interact/creditscore.jar`.
5. Erstellen Sie ein externes Callout mit dem Namen `GetCreditScore` und dem Klassenpfad `/data/interact/creditscore.jar` in der Kategorie `externalCallouts` auf der Seite **Konfigurationen verwalten**.
6. In einem interaktiven Ablaufdiagramm kann das Callout als `EXTERNALCALLOUT('GetCreditScore')` verwendet werden.

Benutzeroberfläche IInteractProfileDataService

Die Profildatenservices-API ist in der Benutzeroberfläche `iInteractProfileDataService` enthalten. Diese Benutzeroberfläche ermöglicht Ihnen, über eine oder mehrere externe Datenquellen (z. B. eine Flachdatei, einen Web-Service usw.) hierarchische Daten in eine Unica Interact-Sitzung zu importieren, wenn die Unica Interact-Sitzung startet oder wenn sich die Zielgruppen-ID einer Unica Interact-Sitzung ändert.

Um das Importieren hierarchischer Daten mithilfe der Profildatenservices-API zu entwickeln, müssen Sie eine Java-Klasse schreiben, die Informationen von allen Datenquellen abruft und einem `ISessionDataRootNode`-Objekt zuordnet. Danach müssen Sie mithilfe des Makros EXTERNALCALLOUT in einem Prozess "Auswählen" für ein interaktives Ablaufdiagramm auf die zugeordneten Daten verweisen.

Sie müssen Ihre Implementierung anhand von `interact_externalcallout.jar` im `lib`-Verzeichnis Ihrer Unica Interact-Laufzeitumgebungsinstallation kompilieren.

Eine vollständige JavaDoc-Dokumentation zur Verwendung dieser Benutzeroberfläche finden Sie in den Dateien *`Interact_home`*`/docs/externalCalloutJavaDoc`, auf die Sie in jedem Web-Browser zugreifen können.

Eine Beispielimplementierung für die Verwendung des Profile Data Service, einschließlich kommentierter Beschreibungen der Implementierung des Beispiels, finden Sie unter *`Interact_home`*`/samples/externalcallout/XMLProfileDataService.java`.

Anmerkung: Die Beispielimplementierung soll nur als Beispiel dienen. Sie sollten dieses Beispiel nicht in Ihrer Implementierung verwenden.

Hinzufügen einer Datenquelle zur Verwendung mit Profildatenservices

Mit diesem Verfahren fügen Sie eine Datenquelle zur Verwendung mit den Profildatenservices hinzu.

Das Makro EXTERNALCALLOUT erkennt eine Datenquelle für den Import hierarchischer Daten mithilfe der Profildatenservices nur, wenn Sie die entsprechenden Konfigurationseigenschaften definiert haben.

Fügen Sie in Unica Platform für die Laufzeitumgebung die folgenden Konfigurationseigenschaften in der Kategorie `Interact > profile >Audience > Levels [AudienceLevelName] Profile Data Services` hinzu oder definieren Sie sie.

Konfigurationseigenschaft	Einstellung
Kategorie `Neuer Kategoriename`	Der Name der von Ihnen definierten Datenquelle. Der Name, den Sie hier eingeben, muss innerhalb der Datenquellen einer Zielgruppenebene eindeutig sein.
`enabled`	Gibt an, ob die Datenquelle für die Zielgruppenebene aktiviert ist, in der sie definiert ist.
`className`	Der vollqualifizierte Name der Datenquellenklasse, die implementiert `IInteractProfileDataService`
`classPath`	Der Klassenpfad zu Ihren Klassendateien in den Profildatenservices. Wenn Sie die Einstellung weglassen, wird standardmäßig der Klassenpfad des übergeordneten Anwendungsservers verwendet.
Kategorie `Priorität`	Die Priorität dieser Datenquelle in dieser Zielgruppenebene. Der Wert muss für jede Datenquelle in einer Zielgruppenebene eindeutig sein. (Wenn also für eine Datenquelle die Priorität 100 festgelegt ist, kann keine weitere Datenquelle in der Zielgruppenebene eine Priorität von 100 haben.)

Benutzeroberfläche IParameterizableCallout

Die parametrisierbare Aufruf-API ist in der Benutzeroberfläche `IParameterizableCallout` enthalten.

Diese Benutzeroberfläche ist die Basisbenutzeroberfläche der zugänglichen APIs, die Parameter von der Konfiguration über Unica Platform akzeptieren können. Da dies eine Basisbenutzeroberfläche ist, sollte sie nicht direkt implementiert werden. Die Parameter werden von den untergeordneten Knoten des Knotens `Parameter Data` in der Kategorie abgerufen, in der auf diese Implementierung verwiesen wird. Im folgenden Beispiel stellt ESB eine angepasste Implementierung des Profildatenservice dar, über den wiederum die Benutzeroberfläche `IParameterizableCallout` implementiert wird. Die Parameter endPoint und login werden zusammen mit den zugehörigen Werten in diese Implementierungsklasse übergeben, wenn die Unica Interact-Engine versucht, diese zu initialisieren und zu beenden.

```
Profile Data Services
...ESB
   ...Parameter Data
      ...endPoint
      ...login
```

Die Benutzeroberfläche umfasst zwei Methoden:

- `initialize`
- `shutdown`

Initialisieren

Mit der Methode `initialize` wird diese Implementierungsklasse initialisiert.

```
void initialize(java.util.Map<java.lang.String,java.lang.String>
 configurationData)
               throws CalloutException
```

Die Methode `initialize` erfordert die folgenden Parameter.

- **configurationData** - Eine Zuordnung mit Name/Wert-Paaren der von Benutzern konfigurierten Parameter

Löst aus

```
CalloutException
```

shutdown

Bei der Methode `shutdown` wird diese Implementierungsklasse beendet.

```
void shutdown(java.util.Map<java.lang.String,java.lang.String>
 configurationData)
              throws CalloutException
```

Die Methode `shutdown` erfordert die folgenden Parameter.

- **configurationData** - Eine Zuordnung mit Name/Wert-Paaren der von Benutzern konfigurierten Parameter

Löst aus

```
CalloutException
```

Benutzeroberfläche ITriggeredMessageAction

Die API der Aktion für ausgelöste Nachrichten ist in der Benutzeroberfläche `ITriggeredMessageAction` enthalten. Diese Benutzeroberfläche ermöglicht es Ihnen, den Namen dieser Instanz abzurufen und festzulegen.

Die Benutzeroberfläche `ITriggeredMessageAction` dient als Basisbenutzerschnittstelle für andere Benutzeroberflächen und sollte niemals direkt implementiert werden.

Die Benutzeroberfläche umfasst zwei Methoden:

- `getName`
- `setName`

getName

Mit der Methode `getName` wird der Name der Instanz `ITriggeredMessageAction` zurückgegeben.

```
java.lang.String getName()
```

setName

Mit der Methode `setName` wird der Name der Instanz `ITriggeredMessageAction` festgelegt.

```
void setName(java.lang.String name)
```

Während Sie die Implementierungsklasse dieser Benutzeroberfläche initialisieren, legt Unica Interact den Namen der Benutzeroberfläche mit dem in der Konfigurations-UI angegebenen Namen fest.

Im folgenden Beispiel lautet der Name dieses Gateways `InteractLog`.

```
triggeredMessage
           ...gateways
              ...InteractLog
```

Die Methode `setName` erfordert die folgenden Parameter.

- name - Der Name, den Sie für die Instanz `ITriggeredMessageAction` festlegen möchten.

Benutzeroberfläche IChannelSelector

Die Channel Selector-API ist in der Benutzeroberfläche `IChannelSelector` enthalten. Diese Benutzeroberfläche ermöglicht es Ihnen, die Kanäle für abgehende Nachrichten basierend auf dem zu sendenden Angebot und den Sitzungsattributen auszuwählen.

Eine Beispielimplementierung zur Verwendung der Aktion für ausgelöste Nachrichten, einschließlich kommentierter Beschreibungen zur Vorgehensweise bei der Implementierung

des Beispiels, finden Sie unter *Interact_home*`/samples/triggeredmessage/SampleChannelSelector.java`.

Anmerkung: Die Beispielimplementierung soll nur als Beispiel dienen. Sie sollten dieses Beispiel nicht in Ihrer Implementierung verwenden.

Sie sollten versuchen, diese Implementierung zu verwenden, statt Ihre eigene Implementierung zu schreiben.

Die Benutzeroberfläche umfasst eine Methode:

- `selectChannels`

selectChannels

Bei der Methode `selectChannels` werden die Kanäle für abgehende Nachrichten ausgewählt, an die das übergebene Angebot über die Benutzeroberfläche `IChannelSelector` gesendet werden sollte.

```
java.util.List<java.lang.String> selectChannels

 (java.util.Map<java.lang.String,java.util.Map<java.lang.String,
                                    java.lang.Object>> availableChannels,
                                    com.unicacorp.interact.api.Offer offer,
                                    com.unicacorp.interact.treatment.
                                    optimization.IInteractSessionData
 sessionData)
```

Unica Interact versucht, dieses Angebot an alle diese zurückgegebenen Kanäle zu senden.

Die `selectChannels`-Methode erfordert die folgenden Parameter:

- **availableChannels** - Eine Übersicht über die verfügbaren Kanäle für abgehende Nachrichten, die in der UI für ausgelöste Nachrichten in den Einstellungen für die Entwurfszeit von Unica Interact konfiguriert werden. In jedem Eintrag der Übersicht umfasst der Schlüssel den Namen des Kanals und der Wert die

konfigurierten Parameter dieses Kanals in der Designzeit von Unica Interact. Die Iterationsreihenfolge in dieser Übersicht entspricht der in dieser UI definierten Reihenfolge. Wenn der vom Profil bevorzugte Kanal in der UI für ausgelöste Nachrichten verwendet wird, wird er durch den tatsächlichen Kanal ersetzt, bevor diese Methode aufgerufen wird. Zusätzlich bleibt nur das Vorkommen mit der höchsten Priorität erhalten und alle Duplikate werden entfernt, wenn in der UI der gleiche Kanal mehrmals auftaucht.

- **offer** - Das Angebot, das bereitgestellt werden soll
- **sessionData** - die Attribute, die derzeit in der zugehörigen Unica Interact-Sitzung gespeichert sind

Benutzeroberfläche IDispatcher

Die Dispatcher-API ist in der Benutzeroberfläche `IDispatcher` enthalten. Diese Benutzeroberfläche sendet Angebote an zielgruppenspezifische Gateways.

Da es für jeden konfigurierten Dispatcher nur eine Instanz dieser Klasse gibt, muss die Implementierung dieser Benutzeroberfläche statusunabhängig aus der Perspektive von Unica Interact erfolgen.

Eine Beispielimplementierung zur Verwendung der Aktion für ausgelöste Nachrichten, einschließlich kommentierter Beschreibungen zur Vorgehensweise bei der Implementierung des Beispiels, finden Sie unter *`Interact_home`*`/samples/triggeredmessage/SampleDispatcher.java`.

Anmerkung: Die Beispielimplementierung soll nur als Beispiel dienen. Sie sollten dieses Beispiel nicht in Ihrer Implementierung verwenden.

Sie sollten versuchen, diese Implementierung zu verwenden, statt Ihre eigene Implementierung zu schreiben.

Die Benutzeroberfläche umfasst eine Methode:

- `dispatch`

dispatch

Bei der Methode `dispatch` werden Angebote an die Zielgateways der Benutzeroberfläche `IDispatcher` gesendet.

```
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
```

Sobald Kanäle für abgehende Nachrichten für ein mögliches Angebot ausgewählt wurden, versucht Unica Interact, das mögliche Angebot an die Handler zu senden, die dem Kanal zugeordnet sind. Die Handler werden basierend auf ihren definierten Prioritäten von hoch zu niedrig festgelegt. Unica Interact ruft diese Methode des konfigurierten Dispatchers für jeden Handler auf. Es hängt von der Implementierung dieser Dispatcherinstanz ab, wie das Angebot an das Zielgateway weitergeleitet wird, das bei demselben Handler konfiguriert ist. Wenn demselben Handler aufgrund der gleichen Bewertung ausgelöster Nachrichten mehrere Angebote gesendet werden, versucht Unica Interact, alle diese Angebote in einem Batch zu senden.

Die `dispatch`-Methode erfordert die folgenden Parameter:

- **channel** - der Kanal für abgehende Nachrichten, an den diese Angebote gesendet werden
- **gatewayName** - Der Name des Zielgateways
- **offers** - Die Angebote, die in einem Stapel an das Gateway gesendet werden sollen
- **profileData** - Von `IGateway.validate` aufgefüllte Profilattribute, die an `IGateway.deliver` übergeben werden

Rückgabewert

Die Methode `dispatch` gibt den Rückgabewert zurück, wenn das Senden erfolgreich war oder fehlgeschlagen ist

Löst aus

```
com.unicacorp.interact.exceptions.InteractException
```

Benutzeroberfläche IGateway

Die Gateway-API ist in der Benutzeroberfläche `IGateway` enthalten. Diese Benutzeroberfläche erhält Angebote von Unica Interact und sendet die Angebote an ihr Ziel.

Jede Implementierung dieser Benutzeroberfläche kommuniziert mit einem bestimmten Ziel. Am Ziel müssen die notwendige Datentransformation, Attributauffüllung und ähnliche zielbezogene Aufgaben ausgeführt werden.

Eine Beispielimplementierung zur Verwendung der Aktion für ausgelöste Nachrichten, einschließlich kommentierter Beschreibungen zur Vorgehensweise bei der Implementierung des Beispiels, finden Sie unter *`Interact_home`*`/samples/triggeredmessage/SampleOutboundGateway.java`.

Anmerkung: Die Beispielimplementierung soll nur als Beispiel dienen. Sie sollten dieses Beispiel nicht in Ihrer Implementierung verwenden. Beispiel: SampleOutboundGateway ist unter Beispielverzeichnis als Referenz für die Implementierung enthalten.

Die Benutzeroberfläche umfasst zwei Methoden:

- `deliver`
- `validate`

Deliver

Die Methode `deliver` wird aufgerufen, wenn das Angebot bzw. die Angebote an ein Ziel in der Benutzeroberfläche `IGateway` gesendet werden soll(en).

```
void deliver(java.util.Collection<com.unicacorp.interact.api.Offer> offers,
                     com.unicacorp.interact.api.NameValuePair[]
 profileData,
                     java.lang.String channel)
```

Die `deliver`-Methode erfordert die folgenden Parameter:

- **offers** - Das Angebot, das gesendet werden soll
- **profileData** - Die mit der Methode "Validate" in `parameterMap` aufgefüllten Profilattribute
- **channel** - der Kanal für abgehende Nachrichten, an den diese Angebote gesendet werden

Bestätigen

Mit der Methode `validate` werden mögliche Angebote in der Benutzeroberfläche `IGateway` validiert.

```
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)
```

Die Unica Interact-Engine ruft diese Methode zur Validierung der möglichen Angebote auf. Bei der Implementierung dieser Methode sollten die Angebote, Angebotsattribute und Sitzungsattribute auf die Voraussetzungen des Ziels geprüft werden, um zu ermitteln, welches Angebot bzw. welche Angebote über dieses Gateway gesendet werden kann/können. Zusätzlich könnten notwendige Parameter zur übergebenen Übersicht hinzugefügt werden, die wieder an die Methode zur Bereitstellung übergeben wird.

Die `validate`-Methode erfordert die folgenden Parameter:

- **sessionData** - die Attribute, die derzeit in der zugehörigen Unica Interact-Sitzung gespeichert sind
- **candidateOffers** - die Angebote, die basierend auf der Methode zur Angebotsauswahl, den zugehörigen Parametern und weiteren Faktoren von Unica Interact ausgewählt wurden. Diese Angebote kommen aus der Perspektive von Unica Interact für die Bereitstellung infrage, allerdings vorbehaltlich des Gateways.
- **parameterMap** - eine Übersicht, die bei der Implementierung dieser Methode zur Übergabe von Parametern an die Methode zur Bereitstellung verwendet werden sollte
- **channel** - der Kanal für abgehende Nachrichten, an den diese Angebote gesendet werden

Kapitel 11. Unica Interact-Dienstprogramme

In diesem Abschnitt werden die Administrationsdienstprogramme beschrieben, die mit Unica Interact verfügbar sind.

Dienstprogramm RunDeployment (runDeployment.sh/.bat)

Mithilfe des Befehlszeilentools `runDeployment` können Sie von der Befehlszeile aus einen interaktiven Kanal für eine bestimmte Servergruppe implementieren. Verwenden Sie dazu die Einstellungen in der Datei `deployment.properties`, die alle möglichen Parameter beschreibt. Sie befindet sich im selben Verzeichnis wie das Tool `runDeployment`. Die Möglichkeit, von der Befehlszeile aus einen interaktiven Kanal zu implementieren, ist besonders nützlich, wenn Sie die Funktion O`OffersBySQL` verwenden. Sie können z. B. ein Unica Campaign-Batch-Flowchart konfigurieren, das regelmäßig ausgeführt wird. Wenn die Ausführung des Flowcharts abgeschlossen ist, kann ein Trigger aufgerufen werden, der die Implementierung der Angebote in der `OffersBySQL`-Tabelle mithilfe dieses Befehlszeilentools initialisiert.

Syntax

Sie finden das Befehlszeilentool `runDeployment`, das automatisch auf dem Unica Interact-Entwicklungszeitserver installiert wird, im folgenden Verzeichnis:

`Interact_home``/interactDT/tools/deployment/runDeployment.sh` (oder `runDeployment.bat` auf einem Windows™-Server)

Das einzige Argument, das dem Befehl übergeben wird, ist die Position der Datei `deployment.properties`, die alle möglichen Parameter beschreibt, die zum Implementieren der Kombination von interaktivem Kanal und Laufzeitservergruppe erforderlich sind. Zu Referenzzwecken ist eine Beispieldatei verfügbar.

Anmerkung: Bevor Sie das Dienstprogramm `runDeployment` verwenden, müssen Sie es zunächst in einem beliebigen Texteditor bearbeiten, damit es die Position der Java™-Laufzeitumgebung auf dem Server angibt. Sie können z. B. den Pfad

Interact_home/jre oder *Platform_home/jre* angeben, wenn eines dieser Verzeichnisse die Java™-Laufzeit enthält, die vom Dienstprogramm verwendet werden soll. Stattdessen können Sie auch den Pfad zu jeder beliebigen Java™-Laufzeitumgebung angeben, deren Verwendung mit diesem Release der Produkte unterstützt wird.

Verwenden des Dienstprogramms runDeployment in einer sicheren SSL-Umgebung

Damit Sie das Dienstprogramm „runDeployment“ verwenden können, wenn auf dem Unica Interact-Server Sicherheitsfunktionen aktiviert wurden (und Verbindungen daher über einen SSL-Port hergestellt werden), müssen Sie die folgenden Schritte ausführen, um die Java-TrustStore-Eigenschaft hinzuzufügen:

1. Wenn Sie die Datei `deployment.properties` für Ihre Implementierung des interaktiven Kanals bearbeiten, ändern Sie die Eigenschaft `deploymentURL` so, dass die sichere SSL-URL verwendet wird, so wie im folgenden Beispiel:

   `deploymentURL=https://<HOST>.<DOMAIN>:<PORT>/Campaign/interact/InvokeDeploymentServlet`
2. Bearbeiten Sie das Script `runDeployment.sh` bzw. `runDeployment.bat` mithilfe eines beliebigen Texteditors, um das folgende Argument zu der Zeile hinzuzufügen, die mit ${{JAVA_HOME}} beginnt:

   `-Djavax.net.ssl.trustStore=<TrustStorePath>`

   Die Zeile könnte z. B. so aussehen, nachdem Sie das TrustStore-Argument hinzugefügt haben:

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

   Ersetzen Sie <TrustStorePath> durch den Pfad zum tatsächlichen SSL-geschützten Truststore.

Ausführen des Dienstprogramms

Nachdem Sie das Dienstprogramm so bearbeitet haben, dass es die Java™-Laufzeitumgebung angibt, und eine Kopie der Datei `deployment.properties` an Ihre Umgebung angepasst haben, können Sie das Dienstprogramm mit dem folgenden Befehl ausführen:

Interact_home`/interactDT/tools/deployment/runDeployment.sh` *`deployment.properties`*

Ersetzen Sie Interact_home durch den tatsächlichen Wert der Unica Interact-Entwicklungszeitinstallation und ersetzen Sie deployment.properties durch den tatsächlichen Pfad und den Namen der Eigenschaftendatei, die Sie für diese Implementierung angepasst haben.

Beispiel für eine `deployment.properties`-Datei

Die Beispieldatei `deployment.properties` enthält eine kommentierte Liste aller Parameter, die Sie anpassen müssen, damit sie mit Ihrer Umgebung übereinstimmen. Die Beispieldatei enthält darüber hinaus Anmerkungen, die die einzelnen Parameter erklären und angeben, warum Sie einen bestimmten Wert möglicherweise anpassen müssen.

```
##############################################################################
####
#
# 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/InvokeDeploymentServl
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
```

Bereinigung abgelaufenes Token-Dienstprogramm

Um den Code und die Konfiguration in der Interact-Laufzeit zu implementieren, müssen die abgelaufenen Token aus der UACI_RTToken-Tabelle entfernt werden. Dies muss automatisch über einen Hintergrund-Thread mit einem konfigurierbaren Intervall erfolgen. Dieser Hintergrund muss generisch sein, damit er für die Bereinigung anderer Tabellen verwendet werden kann.

Konfigurationsänderungen

Ein Knoten "Bereinigung" wird unter dem Pfad "Affinium|interact|services" hinzugefügt, um bereinigungsbezogene Aktivitäten durchzuführen. Ein Unterknoten "expiredTokens" wird unter dem Pfad "Affinium|interact|services|Cleanup" hinzugefügt, um die Bereinigungsoperation für die abgelaufenen Token durchzuführen. Sie müssen das Feld "enable" wie `true` unter dem Pfad "Affinium|interact|services|Cleanup|expiredTokens" setzen. Der Standardwert des "enable"-Feldes von "expiredTokens" ist auf `True` gesetzt. Gültige Werte sind `True` und `False`.

Anmerkung: Benutzer können die Token in die UACI_RTToken-Tabelle einfügen, indem sie eine Sitzung starten und das Feld "tokenAuthentication" unter dem Pfad "Affinium|interact|general|API" aktivieren.

Optionale JVM-Parameter.

Benutzer können die folgenden JVM-Parameter unter den JVM-Optionen einstellen.

- -Dcom.unica.interact.cleanupThreadPoolCoreSize=1
- -Dcom.unica.interact.cleanupInitialDelay=300
- -Dcom.unica.interact.cleanupDelay=300
- -Dcom.unica.interact.cleanupBatchSize=5000

Dabei gilt Folgendes:

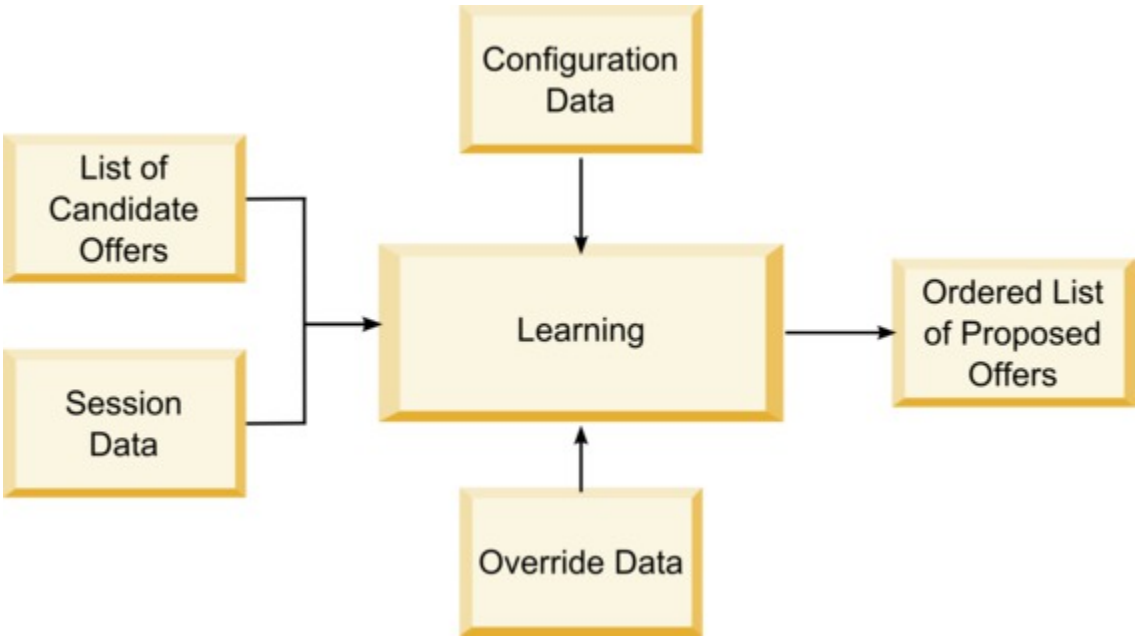
- cleanupThreadPoolCoreSize ist die Anzahl der Threads, die im Pool gehalten werden müssen. Wenn nicht gesetzt, nimmt das System den Standardwert 1 an.
- cleanupInitialDelay ist die Zeit für die Verzögerung der ersten Ausführung in Sekunden. Wenn nicht gesetzt, nimmt das System den Standardwert 300 an.
- cleanupDelay ist die in Sekunden ausgedrückte Verzögerung zwischen der Beendigung einer Ausführung und dem Beginn der darauf folgenden. Wenn nicht gesetzt, nimmt das System den Standardwert 300 an.
- cleanupBatchSize ist die Anzahl der abgelaufenen Token, die die Benutzer in einem Versuch löschen möchten. Wenn nicht gesetzt, nimmt das System den Standardwert 5000 an.

Kapitel 12. Informationen zur Lern-API

Unica Interact stellt ein Lernmodul bereit, das einen naiven bayesschen Algorithmus verwendet, um Besucheraktionen zu überwachen und um optimale Angebote (in Bezug auf Annahme) vorzuschlagen. Sie können dieselbe Java™-Benutzeroberfläche mit Ihren eigenen Algorithmen unter Verwendung der Lern-API implementieren, um Ihr eigenes Lernmodul zu erstellen.

Anmerkung: Wenn Sie externes Lernen verwenden, geben die Beispielberichte in Bezug auf die Lernfunktion (Berichte "Lerndetails des interaktiven Angebots" und " Steigerungsanalyse von interaktiven Segmenten") keine gültigen Daten zurück.

Auf der einfachsten Ebene stellt die Lern-API Methoden bereit, um Daten von der Laufzeitumgebung zu erfassen und eine geordnete Liste von empfohlenen Angeboten zurückzugeben.



Sie können die folgenden Daten von Unica Interact erfassen:

- Angebotskontaktdaten
- Angebotsannahmedaten
- Alle Sitzungsdaten
- Unica Campaign spezifische Angebotsdaten
- Konfigurationseigenschaften, die in der Kategorie `learning` für die Designumgebung und in der Kategorie `offerserving` für die Laufzeitumgebung definiert sind

Sie können diese Daten in Ihren Algorithmen verwenden, um eine Liste von vorgeschlagenen Angeboten zu erstellen. Sie geben dann eine Liste von empfohlenen Angeboten in der Reihenfolge von höchster zu niedrigster Empfehlung zurück.

Sie können auch die Lern-API verwenden, um Daten für Ihre Lernimplementierung zu erfassen (im Diagramm nicht angezeigt). Sie können diese Daten im Speicher ablegen oder sie in einer Datei oder Datenbank für spätere Analyse protokollieren.

Nach der Erstellung Ihrer Java™-Klassen können Sie sie in eine JAR-Datei konvertieren. Nachdem Sie JAR-Dateien erstellt haben, müssen Sie die Laufzeitumgebung auch konfigurieren, Ihr externes Lernmodul zu erkennen, indem Sie Konfigurationseigenschaften bearbeiten. Sie müssen Ihre Java™-Klassen oder JAR-Dateien auf jeden Laufzeitserver kopieren, der Ihr externes Lernmodul verwendet.

Zusätzlich zu den Informationen in diesem Abschnitt steht die JavaDoc für die Lernoptimierungsprogramm-API auf jedem Laufzeitserver im Verzeichnis `Interact/docs/learningOptimizerJavaDoc` zur Verfügung.

Sie müssen Ihre Implementierung anhand von `interact_learning.jar` im `lib`-Verzeichnis Ihrer Unica Interact-Laufzeitumgebungsinstallation kompilieren.

Wenn Sie Ihre benutzerdefinierte Lernimplementierung schreiben, sollten Sie die folgenden Richtlinien berücksichtigen.

- Die Leistung ist entscheidend.
- Muss mit Multithreading funktionieren und Thread-sicher sein.

- Muss alle externen Ressourcen steuern, unter Berücksichtigung von Fehlermodi und Leistung.
- Verwenden Sie entsprechende Ausnahmebedingungen, entsprechende Protokollierung (log4j) und entsprechenden Speicher.

Konfigurieren der Laufzeitumgebung für die Erkennung von externen Lernmodulen

Sie können die Lern-Java™-API verwenden, um Ihr eigenes Lernmodul zu schreiben. Sie müssen die Laufzeitumgebung konfigurieren, um Ihr Lerndienstprogramm in Unica Platform zu erkennen.

Sie müssen den Unica Interact-Laufzeitserver erneut starten, damit diese Änderungen wirksam werden.

1. Bearbeiten Sie in Unica Platform für die Laufzeitumgebung folgende Konfigurationseigenschaften in der Kategorie `Interact > offerserving`. Die Konfigurationseigenschaften für die Lernoptimierungsprogramm-API befinden sich in der Kategorie `Interact > offerserving > External Learning Config`.

Konfigurationseigenschaft	Einstellung
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Klassenname für das externe Lernen
`externalLearningClassPath`	Der Pfad zu den Klassen- oder JAR-Dateien auf dem Laufzeitserver für das externe Lernen. Wenn Sie eine Servergruppe verwenden und alle Laufzeitserver dieselbe Instanz von Unica Platform referenzieren, muss jeder Server über eine Kopie der Klassen- oder JAR-Dateien an derselben Position verfügen.

2. Starten Sie den Unica Interact-Laufzeitserver erneut, damit diese Änderungen wirksam werden.

Benutzeroberfläche ILearning

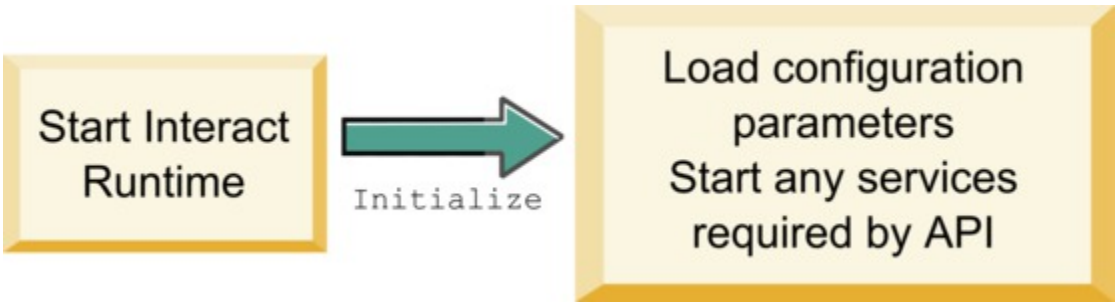
Die Lern-API wird um die Benutzeroberfläche `ILearning` aufgebaut. Sie müssen die Benutzeroberfläche `ILearning` implementieren, um die angepasste Logik Ihres Lernmoduls zu unterstützen.

Unter Anderem ermöglicht es Ihnen die Benutzeroberfläche `ILearning`, Daten aus der Laufzeitumgebung für Ihre Java™-Klasse zu erfassen und Angebote zurück an den Laufzeitserver zu senden.

Initialisieren

Die Methode `initialize` wird beim Start des Laufzeitservers einmal aufgerufen. Wenn es Operationen gibt, die nicht wiederholt werden müssen, aber möglicherweise die Leistung während der Laufzeit einschränken, wie z. B. das Laden von statischen Daten aus einer Datenbanktabelle, dann sollten sie durch diese Methode ausgeführt werden.

```
initialize(ILearningConfig config, boolean debug)
```



- **config** - ein `ILearningConfig`-Objekt definiert alle für die Lernfunktion relevanten Konfigurationseigenschaften.
- **debug** - ein boolescher Ausdruck. Wenn `true` gibt er an, dass die Ausführlichkeit der Protokollstufe für das Laufzeitumgebungssystem auf "debug" festgelegt ist. Es wird empfohlen, dass Sie diesen Wert auswählen, bevor in ein Protokoll geschrieben wird.

Wenn die Methode `initialize` aus irgendeinem Grund fehlschlägt, wird eine `LearningException` ausgelöst.

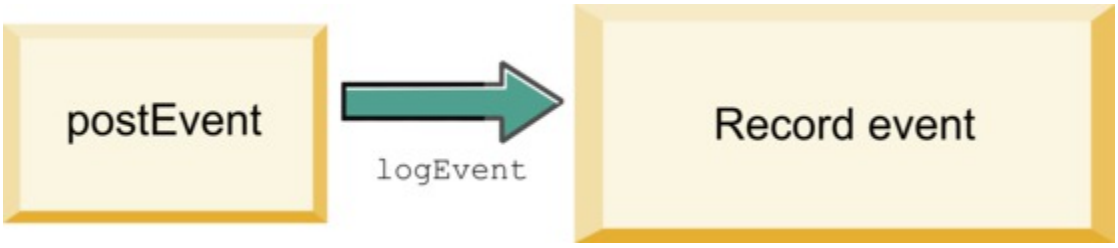
Rückgabewert

Keine.

logEvent

Die Methode `logEvent` wird vom Laufzeitserver aufgerufen, wenn die Unica Interact-API ein Ereignis bereitstellt, das konfiguriert ist, als ein Kontakt oder als eine Antwort protokolliert zu werden. Verwenden Sie diese Methode, um Kontakt- und Antwortdaten in eine Datenbank oder Datei für Berichterstellungs- oder Lernzwecke zu protokollieren. Beispiel: Wenn Sie algorithmisch die Wahrscheinlichkeit, dass ein Kunde ein Angebot annimmt, basierend auf Kriterien bestimmen wollen, verwenden Sie diese Methode, um die Daten zu protokollieren.

```
logEvent(ILearningContext context,
              IOffer offer,
              IClientArgs clientArgs,
              IInteractSession session,
              boolean debug)
```



- **context** - ein `ILearningContext`-Objekt, das den Lernkontext des Ereignisses definiert, z. B. Kontakt, Annehmen oder Ablehnen.
- **offer** - ein `IOffer`-Objekt, das das Angebot definiert, zu dem dieses Ereignis protokolliert wird.

- **clientArgs** - ein `IClientArgs`-Objekt, das Parameter definiert. Derzeit erfordert `logEvent` keine `clientArgs`, daher ist dieser Parameter möglicherweise leer.
- **session** - ein `IInteractSession`-Objekt, das alle Sitzungsdaten definiert.
- **debug** - ein boolescher Ausdruck. Wenn `true` gibt er an, dass die Ausführlichkeit der Protokollstufe für das Laufzeitumgebungssystem auf "debug" festgelegt ist. Es wird empfohlen, dass Sie diesen Wert auswählen, bevor in ein Protokoll geschrieben wird.

Wenn die Methode `logEvent` fehlschlägt, wird eine `LearningException` ausgelöst.

Rückgabewert

Keine.

optimizeRecommendList

Die Methode `optimizeRecommendList` sollte eine Liste von empfohlenen Angeboten und die Sitzungsdaten nehmen und eine Liste zurückgeben, die die angeforderte Anzahl von Angeboten enthält. Die Methode `optimizeRecommendList` sollte die Angebote auf eine bestimmte Art mit Ihrem eigenen Lernalgorithmus sortieren. Die Liste der Angebote muss so sortiert sein, dass die Angebote, die Sie zuerst anbieten wollen, sich am Anfang der Liste befinden. Beispiel: Wenn Ihr Lernalgorithmus den besten Angeboten eine niedrige Bewertung gibt, sollten die Angebote 1, 2, 3 sortiert sein. Wenn Ihr Lernalgorithmus den besten Angeboten einee hohe Bewertung gibt, sollten die Angebote 100, 99, 98 sortiert sein.

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

Die Methode `optimizeRecommendList` erfordert die folgenden Parameter:

- **recList** - eine Liste der Verfahrensobjekte (Angebote), die von der Laufzeitumgebung empfohlen werden.
- **clientArg** - ein IClientArgs-Objekt, das zumindest die Anzahl der Angebote enthält, die von der Laufzeitumgebung angefordert werden.
- **session** - ein IInteractSession-Objekt, das alle Sitzungsdaten enthält.
- **debug** - ein boolescher Ausdruck. Wenn `true` gibt er an, dass die Ausführlichkeit der Protokollstufe für das Laufzeitumgebungssystem auf "debug" festgelegt ist. Es wird empfohlen, dass Sie diesen Wert auswählen, bevor in ein Protokoll geschrieben wird.

Wenn die Methode `optimizeRecommendList` fehlschlägt, wird eine `LearningException` ausgelöst.

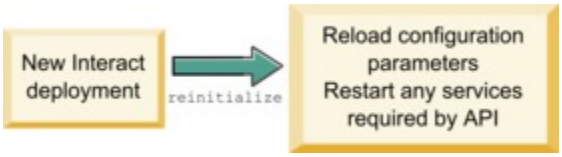
Rückgabewert

Die Methode `optimizeRecommendList` gibt eine Liste von `ITreatment`-Objekten zurück.

reinitialize

Die Laufzeitumgebung ruft die Methode`reinitialize` jedes Mal auf, wenn es eine neue Implementierung gibt. Diese Methode übergibt die gesamten Lernkonfigurationsdaten. Wenn Sie Services haben, die von der Lern-API erfordert werden und Konfigurationseigenschaften lesen, sollte diese Benutzeroberfläche sie erneut starten.

```
reinitialize(ILearningConfig config,
                  boolean debug)
```



- **config** - ein `ILearningConfig`-Objekt definiert alle für die Lernfunktion relevanten Konfigurationseigenschaften.
- **debug** - ein boolescher Ausdruck. Wenn `true` gibt er an, dass die Ausführlichkeit der Protokollstufe für das Laufzeitumgebungssystem auf "debug" festgelegt ist. Es wird empfohlen, dass Sie diesen Wert auswählen, bevor in ein Protokoll geschrieben wird.

Wenn die Methode `logEvent` fehlschlägt, wird eine `LearningException` ausgelöst.

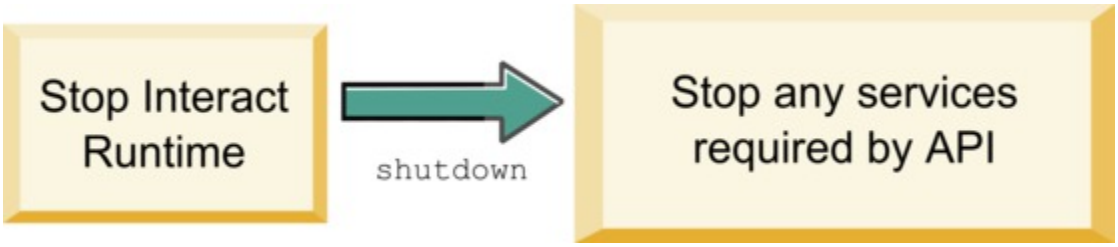
Rückgabewert

Keine.

shutdown

Die `shutdown`-Methode wird beim Beenden des Laufzeitservers einmal aufgerufen. Wenn es Bereinigungsaufgaben gibt, die von Ihrem Lernmodul erfordert werden, sollten sie zu diesem Zeitpunkt ausgeführt werden.

```
shutdown(ILearningConfig config, boolean debug)
```



Die Methode `shutdown` erfordert die folgenden Parameter.

- **config** - ein `ILearningConfig`-Objekt, das alle Konfigurationseigenschaften definiert.
- **debug** - ein boolescher Ausdruck. Wenn `true` gibt er an, dass die Ausführlichkeit der Protokollstufe für das Laufzeitumgebungssystem auf "debug" festgelegt ist. Es wird empfohlen, dass Sie diesen Wert auswählen, bevor in ein Protokoll geschrieben wird.

Wenn die Methode `logEvent` fehlschlägt, wird eine `LearningException` ausgelöst.

Rückgabewert

Keine.

Benutzeroberfläche IAudienceID

Die Benutzeroberfläche `IAudienceID` unterstützt die Benutzeroberfläche `IInteractSession`. Dies ist eine Benutzeroberfläche für die Zielgruppen-ID. Da Ihre Zielgruppen-ID aus mehreren Abschnitten bestehen kann, ermöglicht es Ihnen diese Benutzeroberfläche, auf alle Elemente der Zielgruppen-ID sowie auf den Zielgruppenebenennamen zuzugreifen.

getAudienceLevel

Die `getAudienceLevel`-Methode gibt die Zielgruppenebene zurück.

```
getAudienceLevel()
```

Rückgabewert

Die `getAudienceLevel`-Methode gibt die Zielgruppenebene zurück.

getComponentNames

Die `getComponentNames`-Methode ruft einen Satz mit den Namen der Komponenten ab, aus denen sich die Zielgruppen-ID zusammensetzt. Wenn Ihre Zielgruppen-ID zum Beispiel die Werte `customerName` und `accountID` umfasst, gibt `getComponentNames` einen Satz zurück, der die Zeichenfolgen customerName und accountID enthält.

```
getComponentNames()
```

Rückgabewert

Ein Satz aus Zeichenfolgen, die die Namen der Komponenten der Zielgruppen-ID enthalten.

getComponentValue

Die `getComponentValue`-Methode gibt den Wert der angegebenen Komponente zurück.

```
getComponentValue(String componentName)
```

- **componentName** - eine Zeichenfolge, die den Namen der Komponente definiert, für die der Wert abgerufen werden soll. Diese Zeichenfolge unterscheidet nicht zwischen Groß- und Kleinschreibung.

Rückgabewert

Die `getComponentValue`-Methode gibt ein Objekt zurück, das den Wert der Komponente definiert.

IClientArgs

Die Benutzeroberfläche `IClientArgs` unterstützt die Benutzeroberfläche `ILearning`. Diese Benutzeroberfläche ist eine Abstraktion, um Daten zu erfassen, die an den Server vom Touchpoint übergeben werden, die noch nicht von den Sitzungsdaten erfasst sind. Beispiel: Die Anzahl der Angebote, die von der Unica Interact-API-Methode `getOffers` angefordert werden. Diese Daten werden in einer Zuordnung gespeichert.

getValue

Die `getValue`-Methode gibt den Wert des angeforderten Zuordnungselements zurück.

```
getValue(int clientArgKey)
```

Die folgenden Elemente sind in der Zuordnung erforderlich.

- **1** - NUMBER_OF_OFFERS_REQUESTED. Die Anzahl der Angebote, die von der Unica Interact-API-Methode `getOffers` angefordert werden. Diese Konstante gibt eine Ganzzahl zurück.

Rückgabewert

Die `getValue`-Methode gibt ein Objekt zurück, das den Wert der angeforderten Zuordnungskonstante definiert.

IInteractSession

Die Benutzeroberfläche `IInteractSession` unterstützt die Benutzeroberfläche `ILearning`. Dies ist eine Benutzeroberfläche zur aktuellen Sitzung in der Laufzeitumgebung.

getAudienceId

Die `getAudienceId`-Methode gibt ein `AudienceID`-Objekt zurück. Verwenden Sie die `IAudienceID`-Benutzeroberfläche, um die Werte zu extrahieren.

```
getAudienceId()
```

Rückgabewert

Die `getAudienceId`-Methode gibt ein `AudienceID`-Objekt zurück.

getSessionData

Die `getSessionData`-Methode gibt eine nicht modifizierbare Zuordnung von Sitzungsdaten zurück, wobei der Name der Sitzungsvariablen als Schlüssel verwendet wird. Der Name der Sitzungsvariablen wird immer großgeschrieben. Verwenden Sie die `IInteractSessionData`-Benutzeroberfläche, um die Werte zu extrahieren.

```
getSessionData()
```

Rückgabewert

Die `getSessionData`-Methode gibt ein `IInteractSessionData`-Objekt zurück.

Benutzeroberfläche IInteractSessionData

Die Benutzeroberfläche `IInteractSessionData` unterstützt die Benutzeroberfläche `ILearning`. Dies ist eine Benutzeroberfläche zu den Laufzeit-Sitzungsdaten für den aktuellen Besucher. Sitzungsdaten werden in einer Liste von Name/Wert-Paaren gespeichert. Sie können diese Benutzeroberfläche auch verwenden, um den Wert von Daten in der Laufzeitsitzung zu ändern.

getDataType

Die `getDataType`-Methode gibt den Datentyp für den angegebenen Parameternamen zurück.

```
getDataType(string parameterName)
```

Rückgabewert

Die `getDataType`-Methode gibt ein InteractDataType-Objekt zurück. InteractDataType ist eine Java™-Aufzählung, die als unbekannt, Zeichenfolge, Doppelzeichen, Datum oder Liste dargestellt wird.

getParameterNames

Die `getParameterNames`-Methode gibt einen Satz mit allen Namen der Daten in der aktuellen Sitzung zurück.

```
getParameterNames()
```

Rückgabewert

Die `getParameterNames`-Methode gibt einen Satz mit allen Namen zurück, für die Werte festgelegt wurden. Jeder Name im Satz kann in getValue(String) übergeben werden, um einen Wert zurückzugeben.

getValue

Die `getValue`-Methode gibt den Objektwert zurück, der dem angegebenen `parameterName` entspricht. Das Objekt kann eine Zeichenfolge, ein Doppelzeichen oder ein Datum sein.

```
getValue(parameterName)
```

Die Methode `getValue` erfordert die folgenden Parameter.

- **parameterName** - eine Zeichenfolge, die den Namen des Name/Wert-Paares der Sitzungsdaten definiert.

Rückgabewert

Die `getValue`-Methode gibt ein Objekt zurück, das den Wert des angegebenen Parameters enthält.

setValue

Mit der `setValue`-Methode kann ein Wert für den angegebenen `parameterName` festgelegt werden. Der Wert kann eine Zeichenfolge, ein Doppelzeichen oder ein Datum sein.

```
setValue(string parameterName, object value)
```

Die `setValue`-Methode benötigt die folgenden Parameter:

- **parameterName** - eine Zeichenfolge, die den Namen des Name/Wert-Paares der Sitzungsdaten definiert.
- **value** - ein Objekt, das den Wert des angegebenen Parameters definiert.

Rückgabewert

Keine.

ILearningAttribute

Die Benutzeroberfläche `ILearningAttribute` unterstützt die Benutzeroberfläche `ILearningConfig`. Dies ist eine Benutzeroberfläche zu den Lernattributen, die in Konfigurationseigenschaften in der Kategorie `learningAttributes` definiert sind.

getName

Die `getName`-Methode gibt den Namen des Lernattributs zurück.

```
getName()
```

Rückgabewert

Die `getName`-Methode gibt eine Zeichenfolge zurück, die den Namen des Lernattributs definiert.

ILearningConfig

Die Benutzeroberfläche `ILearningConfig` unterstützt die Benutzeroberfläche `ILearning`. Dies ist eine Benutzeroberfläche zu den Konfigurationseigenschaften der Lernfunktion. Alle diese Methoden geben den Eigenschaftswert zurück.

Die Benutzeroberfläche besteht aus 15 Methoden:

- **getAdditionalParameters** - gibt eine Zuordnung von zusätzlichen Eigenschaften zurück, die in der Kategorie `External Learning Config` definiert sind
- **getAggregateStatsIntervalInMinutes** - gibt eine Ganzzahl zurück
- **getConfidenceLevel** - gibt eine Ganzzahl zurück
- **getDataSourceName** - gibt eine Zeichenfolge zurück
- **getDataSourceType** - gibt eine Zeichenfolge zurück
- **getInsertRawStatsIntervalInMinutes** - gibt eine Ganzzahl zurück
- **getLearningAttributes** - gibt eine Liste von `ILearningAttribute`-Objekten zurück
- **getMaxAttributeNames** - gibt eine Ganzzahl zurück
- **getMaxAttributeValues** - gibt eine Ganzzahl zurück
- **getMinPresentCountThreshold** - gibt eine Ganzzahl zurück

- **getOtherAttributeValue** - gibt eine Zeichenfolge zurück
- **getPercentRandomSelection** - gibt eine Ganzzahl zurück
- **getRecencyWeightingFactor** - gibt einen Gleitkommawert zurück
- **getRecencyWeightingPeriod** - gibt eine Ganzzahl zurück
- **isPruningEnabled** - gibt einen booleschen Ausdruck zurück

ILearningContext

Die Benutzeroberfläche `ILearningContext` unterstützt die Benutzeroberfläche `ILearning`.

getLearningContext

Die `getLearningContext`-Methode gibt die Konstante zurück, die festlegt, ob es sich bei diesem Szenario um einen Kontakt, eine Annahme oder eine Ablehnung handelt.

```
getLearningContext()
```

- **1**-LOG_AS_CONTACT
- **2**-LOG_AS_ACCEPT
- **3**-LOG_AS_REJECT

4 und 5 sind für zukünftige Verwendung reserviert.

Rückgabewert

Die `getLearningContext`-Methode gibt eine Ganzzahl zurück.

getResponseCode

Die `getResponseCode`-Methode gibt den Antwortcode zurück, der diesem Angebot zugeordnet ist. Alle diese Antworttypen müssen in der Tabelle `UA_UsrResponseType` in den Unica Campaign-Systemtabellen vorhanden sein.

```
getResponseCode()
```

Rückgabewert

Die `getResponseCode`-Methode gibt eine Zeichenfolge zurück, die den Antwortcode definiert.

IOffer

Die Benutzeroberfläche `IOffer` unterstützt die Benutzeroberfläche `ITreatment`. Dies ist eine Benutzeroberfläche zum Angebotsobjekt, das in der Designumgebung definiert ist. Verwenden Sie die Benutzeroberfläche `IOffer`, um die Angebotsdetails aus der Laufzeitumgebung zu erfassen.

getCreateDate

Die `getCreateDate`-Methode gibt das Datum zurück, an dem das Angebot erstellt wurde.

```
getCreateDate()
```

Rückgabewert

Die `getCreateDate`-Methode gibt ein Datum zurück, das das Datum definiert, an dem das Angebot erstellt wurde.

getEffectiveDateFlag

Die `getEffectiveDateFlag`-Methode gibt eine Zahl zurück, die das Aktivierungsdatum des Angebots definiert.

```
getEffectiveDateFlag()
```

- **0** - das Aktivierungsdatum ist ein absolutes Datum, z. B. 15. März 2010.
- **1** - das Aktivierungsdatum ist das Datum der Empfehlung.

Rückgabewert

Die `getEffectiveDateFlag`-Methode gibt eine Ganzzahl zurück, die das gültige Datum des Angebots definiert.

getExpirationDateFlag

Die `getExpirationDateFlag`-Methode gibt einen Ganzzahlwert zurück, der das Ablaufdatum des Angebots beschreibt.

```
getExpirationDateFlag()
```

- **0** - ein absolutes Datum, z. B. 15. März 2010.
- **1** - eine Anzahl an Tagen nach der Empfehlung, z. B. 14.
- **2** - Monatsende nach Empfehlung. Ein Angebot am 31. März läuft noch am gleichen Tag ab.

Rückgabewert

Die `getExpirationDateFlag`-Methode gibt einen Ganzzahlwert zurück, der das Ablaufdatum des Angebots beschreibt.

getOfferAttributes

Die `getOfferAttributes`-Methode gibt Angebotsattribute zurück, die als ein `IOfferAttributes`-Objekt für das Angebot definiert sind.

```
getOfferAttributes()
```

Rückgabewert

Die `getOfferAttributes`-Methode gibt Angebotsattribute zurück, die als ein `IOfferAttributes`-Objekt für das Angebot definiert sind.

getOfferCode

Die `getTreatmentCode`-Methode gibt den Verfahrenscode des Angebots zurück, wie in Unica Campaign definiert.

```
getOfferCode()
```

Rückgabewert

Die `getOfferCode`-Methode gibt ein `IOfferCode`-Objekt zurück.

getOfferDescription

Die `getOfferDescription`-Methode gibt die Beschreibung des in Unica Campaign definierten Angebots zurück.

```
getOfferDescription()
```

Rückgabewert

Die `getOfferDescription`-Methode gibt eine Zeichenfolge zurück.

getOfferID

Die `getOfferID`-Methode gibt die in Unica Campaign definierte Angebots-ID zurück.

```
getOfferID()
```

Rückgabewert

Die `getOfferID`-Methode gibt die in definierte Angebots-ID zurück.

getOfferName

Die `getOfferName`-Methode gibt den in Unica Campaign definierten Namen des Angebots zurück.

```
getOfferName()
```

Rückgabewert

Die `getOfferName`-Methode gibt eine Zeichenfolge zurück.

getUpdateDate

Die `getUpdateDate`-Methode gibt das Datum der letzten Aktualisierung des Angebots zurück.

```
getUpdateDate()
```

Rückgabewert

Die `getCreateDate`-Methode gibt ein Datum zurück, das das Datum definiert, an dem das Angebot erstellt wurde.

IOfferAttributes

Die Benutzeroberfläche `IOfferAttributes` unterstützt die Benutzeroberfläche `IOffer`. Dies ist eine Benutzeroberfläche zu den Angebotsattributen, die für ein Angebot in der Designumgebung definiert sind. Verwenden Sie die Benutzeroberfläche `IOffer`, um die Angebotsdetails aus der Laufzeitumgebung zu erfassen.

getParameterNames

Die `getParameterNames`-Methode gibt eine Liste mit den Parameternamen des Angebots zurück.

```
getParameterNames()
```

Rückgabewert

Die `getParameterNames`-Methode gibt einen Satz zurück, der die Liste mit den Parameternamen des Angebots definiert.

getValue

Die `getValue`-Methode gibt ein Objekt zurück, das den Wert des Angebotsattributs definiert.

```
getValue(String parameterName)
```

Die `getValue`-Methode gibt einen Wert für das gegebene Angebotsattribut zurück.

Rückgabewert

Benutzeroberfläche IOfferCode

Die Benutzeroberfläche `IOfferCode` unterstützt die Benutzeroberfläche `ILearning`. Dies ist eine Benutzeroberfläche zum Angebotscode, der für ein Angebot in der Designumgebung definiert wurde. Ein Angebotscode kann aus einer oder vielen Zeichenfolgen erstellt werden. Verwenden Sie die Benutzeroberfläche `IOfferCode`, um den Angebotscode aus der Laufzeitumgebung zu erfassen.

getPartCount

Die `getPartCount`-Methode gibt die Anzahl der Teile zurück, aus denen der Angebotscode besteht.

```
getPartCount()
```

Rückgabewert

Die `getPartCount`-Methode gibt eine Ganzzahl zurück, die die Anzahl der Teile des Angebotscodes definiert.

getParts

Die `getParts`-Methode gibt eine nicht modifizierbare Liste mit den Teilen des Angebotscodes zurück.

```
getParts()
```

Rückgabewert

Die `getParts`-Methode gibt eine nicht modifizierbare Liste mit den Teilen des Angebotscodes zurück.

LearningException

Die Klasse `LearningException` unterstützt die Benutzeroberfläche `ILearning`. Einige Methoden innerhalb der Benutzeroberfläche erfordern Implementierungen, um eine `LearningException` auszulösen, was eine einfache Unterklasse von `java.lang.Exception` ist. Es wird dringend für Debugging-Zwecke empfohlen, dass die `LearningException` mit der verursachenden Ausnahmebedingung erstellt wird, falls eine verursachende Ausnahmebedingung vorhanden ist.

IScoreOverride

Die Benutzeroberfläche `IScoreOverride` unterstützt die Benutzeroberfläche `ITreatment`. Diese Benutzeroberfläche ermöglicht es Ihnen, die Daten zu lesen, die in der Bewertungsüberschreibungs- oder Standardangebotstabelle definiert sind.

getOfferCode

Die `getOfferCode`-Methode gibt den Wert der Spalten mit dem Angebotscode in der Tabelle für die Bewertungsüberschreibung dieses Zielgruppenmitglieds zurück.

```
getOfferCode()
```

Rückgabewert

Die `getOfferCode`-Methode gibt ein `IOfferCode`-Objekt zurück, das den Wert der Angebotscodespalten in der Tabelle für die Bewertungsüberschreibung definiert.

getParameterNames

Die `getParameterNames`-Methode gibt die Liste der Parameter zurück.

```
getParameterNames()
```

Rückgabewert

Die `getParameterNames`-Methode gibt einen Satz zurück, der die Parameterliste definiert.

Die `IScoreOverride`-Methode enthält die folgenden Parameter. Sofern nicht anders angegeben, sind diese Parameter mit denen in der Tabelle für die Bewertungsüberschreibung identisch.

- ADJ_EXPLORE_SCORE_COLUMN
- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT - Zum Überschreiben der geschätzten Anzahl der Vorkommen (während die Angebotsgewichtung berechnet wird)
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN - Zum Erstellen eines booleschen Ausdrucks, um die Eignung des Angebots zu bestimmen
- PREDICATE_SCORE - Zum Erstellen eines Ausdrucks für die Berechnung eines numerischen Werts
- SCORE_COLUMN
- ZONE_COLUMN

Sie können auch auf jede andere Spalte verweisen, indem Sie den Namen einer beliebigen Spalte verwenden, die Sie der Tabelle mit den Standardwerten oder der Tabelle für die Bewertungsüberschreibung hinzugefügt haben.

getValue

Die `getValue`-Methode gibt den Wert der Spalte für die Zone in der Tabelle für die Bewertungsüberschreibung dieses Zielgruppenmitglieds zurück.

```
getValue(String parameterName)
```

- **parameterName** - eine Zeichenfolge, die den Namen des Parameters definiert, für den der Wert gelten soll.

Rückgabewert

Die `getValue`-Methode gibt ein Objekt zurück, das den Wert des angeforderten Parameters definiert.

ISelectionMethod

Die Benutzeroberfläche `ISelection` gibt die Methode an, mit der die Empfehlungsliste bereitgestellt wird. Der Standardwert für das Verfahrensobjekt ist EXTERNAL_LEARNING, daher müssen Sie diesen Wert nicht festlegen. Der Wert wird schließlich in "Detaillierter Kontaktverlauf" für Berichterstellungszwecke gespeichert.

Sie können diese Benutzeroberfläche über die bestehenden Konstanten hinaus erweitern, wenn Sie die Daten für die Analyse später speichern wollen. Sie könnten beispielsweise zwei verschiedene Lernmodule erstellen und sie auf separaten Servergruppen implementieren. Sie könnten die Benutzeroberfläche `ISelection` erweitern, um SERVER_GROUP_1 und SERVER_GROUP_2 zu umfassen. Sie könnten dann die Ergebnisse Ihrer zwei Lernmodule vergleichen.

Benutzeroberfläche ITreatment

Die Benutzeroberfläche `ITreatment` unterstützt die Benutzeroberfläche `ILearning` als eine Benutzeroberfläche zu Verfahrensinformationen. Ein Verfahren stellt ein Angebot dar, das einer bestimmten Zelle zugeordnet ist, wie in der Designumgebung definiert. Von dieser Benutzeroberfläche aus können Sie Zellen- und Angebotsinformationen sowie den zugewiesenen Marketing-Score abrufen.

getCellCode

Die `getCellCode`-Methode gibt den in Unica Campaign definierten Zellencode zurück. Die Zelle ist die Zelle, die dem mit diesem Angebot verknüpften Smart-Segment zugeordnet ist.

```
getCellCode()
```

Rückgabewert

Die `getCellCode`-Methode gibt eine Zeichenfolge zurück, die den Zellencode definiert.

getCellId

Die `getCellId-` Methode gibt die interne ID der in Unica Campaign definierten Zelle zurück. Die Zelle ist die Zelle, die dem mit diesem Angebot verknüpften Smart-Segment zugeordnet ist.

```
getOfferName()
```

Rückgabewert

Die `getCellId`-Methode gibt einen Langwert zurück, der die Zellen-ID definiert.

getCellName

Die `getCellCode`-Methode gibt den in Unica Campaign definierten Zellencode zurück. Die Zelle ist die Zelle, die dem mit diesem Angebot verknüpften Smart-Segment zugeordnet ist.

```
getCellName()
```

Rückgabewert

Die `getCellName`-Methode gibt eine Zeichenfolge zurück, die den Zellennamen definiert.

getLearningScore

Die `getLearningScore`-Methode gibt die Punktzahl für dieses Verfahren zurück.

Rückgabewert

Die `getLearningScore`-Methode gibt eine Ganzzahl zurück, die die Punktzahl definiert, die vom Lernalgorithmus ermittelt wurde. Für den Vorrang gilt Folgendes.

1. Rückgabe des neuen Werts, wenn in der von `IScoreoveride.PREDICATE_SCORE_COLUMN` verschlüsselten Zuordnung neue Werte vorhanden sind
2. Rückgabe der berechneten Punktzahl, wenn der Wert nicht null ist.
3. Rückgabe der Punktzahl der Marketiers, wenn in der von `IScoreoveride.SCORE` verschlüsselten Zuordnung neue Werte vorhanden sind
4. Rückgabe der Punktzahl der Marketiers

getMarketerScore

Die `getMarketerScore`-Methode gibt die vom Regler auf der Registerkarte "Interaktionsstrategie" für das Angebot definierte Punktzahl des Marketiers zurück.

```
getMarketerScore()
```

Verwenden Sie `getPredicateScore`, um die mit den erweiterten Optionen auf der Registerkarte "Interaktionsstrategie" definierte Punktzahl eines Marketiers abzurufen.

Verwenden Sie `getLearningScore`, um die tatsächlich im Verfahren verwendete Punktzahl eines Marketiers abzurufen.

Rückgabewert

Die `getMarketerScore`-Methode gibt eine Ganzzahl zurück, die die Punktzahl des Marketiers definiert.

getOffer

Die `getOffer`-Methode gibt das Angebot für das Verfahren zurück.

```
getOffer()
```

Rückgabewert

Die `getOffer`-Methode gibt ein `IOffer`-Objekt zurück, das das Angebot für dieses Verfahren definiert.

getOverrideValues

Die `getOverrideValues`-Methode gibt die in der Tabelle für die Bewertungsüberschreibung oder die in der Tabelle mit den Standardangeboten definierten Überschreibungen zurück.

```
getOverrideValues()
```

Rückgabewert

Die `getOverrideValues`-Methode gibt ein `IScoreOverride`-Objekt zurück.

getPredicate

Die `getPredicate`-Methode gibt das Vergleichselement zurück, das in der Vergleichselementspalte der Tabelle mit den Standardangeboten, in der Tabelle für die Bewertungsüberschreibung oder in den erweiterten Optionen der Verfahrensregeln definiert ist.

```
getPredicate()
```

Rückgabewert

Die `getPredicate`-Methode gibt eine Zeichenfolge zurück, die das Vergleichselement definiert, das in der Vergleichselementspalte der Tabelle mit den Standardangeboten, in der Tabelle für die Bewertungsüberschreibung oder in den erweiterten Optionen der Verfahrensregeln definiert ist.

getPredicateScore

Die `getPredicateScore`-Methode gibt die Punktzahl zurück, die in der Vergleichselementspalte der Tabelle mit den Standardangeboten, in der Tabelle für die Bewertungsüberschreibung oder in den erweiterten Optionen der Verfahrensregeln definiert ist.

```
getPredicateScore()
```

Rückgabewert

Die `getPredicateScore`-Methode gibt ein Doppelzeichen zurück, das die Punktzahl definiert, die in der Vergleichselementspalte der Tabelle mit den Standardangeboten, in der Tabelle für die Bewertungsüberschreibung oder in den erweiterten Optionen der Verfahrensregeln eingestellt ist.

getScore

Die Methode `getScore` gibt den Marketing-Score zurück, der entweder durch die Interaktionsstrategie in Unica Campaign oder durch die Bewertungsüberschreibungstabelle definiert ist.

```
getScore()
```

Die `getScore`-Methode gibt eine der folgenden Optionen zurück:

- Der Marketing-Score des Angebots, der auf der Registerkarte "Interaktionsstrategie" in Unica Campaign definiert ist, wenn die `enableScoreOverrideLookup`-Eigenschaft auf "false" gesetzt ist.
- Die Bewertung (Score) des Angebots, wie durch das `scoreOverrideTable` definiert, wenn die `enableScoreOverrideLookup`-Eigenschaft auf "true" gesetzt ist.

Rückgabewert

Die Methode `getScore` gibt eine Ganzzahl zurück, die die Bewertung des Angebots darstellt.

getTreatmentCode

Die `getTreatmentCode`-Methode gibt den Verfahrenscode zurück.

```
getTreatmentCode()
```

Rückgabewert

Die`getTreatmentCode`-Methode gibt eine Zeichenfolge zurück, die den Verfahrenscode definiert.

setActualValueUsed

Verwenden Sie die `setActualValueUsed`-Methode, um zu definieren, welche Werte in den verschiedenen Stadien bei der Ausführung des Lernalgorithmus verwendet werden.

```
setActualValueUsed(string parmName, object value)
```

Beispiel: Wenn Sie mit dieser Methode in den Kontakt- und Antwortverlaufstabellen schreiben und die vorhandenen Beispielberichte ändern, können Sie Daten aus dem Lernalgorithmus in die Berichte einbinden.

- **parmName** - eine Zeichenfolge, die den Namen des angegebenen Parameters definiert.
- **value** ein Objekt, das den Wert des Parameters definiert, den Sie einstellen.

Rückgabewert

Keine.

Beispiel für eine Lern-API

Dieser Abschnitt enthält eine Beispielimplementierung von ILearningInterface. Beachten Sie, dass diese Implementierung nur ein Beispiel ist und nicht für die Verwendung in einer Produktionsumgebung geeignet ist.

Dieses Beispiel protokolliert Annahme- und Kontaktzählungen und verwendet das Verhältnis von Annahme zu Kontakten für ein bestimmtes Angebot als die Wahrscheinlichkeitsrate für das Angebot. Nicht präsentierte Angebote erhalten eine höhere Priorität für Empfehlungen. Angebote mit zumindest einem Kontakt werden basierend auf absteigender Annahmewahrscheinlichkeitsrate sortiert.

In diesem Beispiel werden alle Zählungen im Speicher abgelegt. Das ist kein realistisches Szenario, weil der Speicherplatz des Laufzeitservers nicht ausreichen wird. In einem realen Produktionsszenario sollten die Zählungen in einer Datenbank persistiert werden.

```
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.unicacorp.interact.treatment.optimization.v2.ILearning#initialize
    *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig,
boolean)
    */
   public void initialize(ILearningConfig config, boolean debug) throws
LearningException
   {
       // If any remote connections are required, this is a good place to
initialize those connections as this
       // method is called once at the start of the interact runtime
webapp.
       // This example does not have any remote connections and prints for
debugging purposes that this method will
       // be called
       System.out.println("Calling initialize for SampleLearning");
   }

   /* (non-Javadoc)
    * @see
com.unicacorp.interact.treatment.optimization.v2.ILearning#reinitialize
    *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig,
boolean)
    */
   public void reinitialize(ILearningConfig config, boolean debug) throws
LearningException
   {
       // If an IC is deployed, this reinitialize method is called to
allow the implementation to
       // refresh any updated configuration settings
       System.out.println("Calling reinitialize for SampleLearning");
   }
```

```
   /* (non-Javadoc)
    * @see
com.unicacorp.interact.treatment.optimization.v2.ILearning#logEvent
    *
(com.unicacorp.interact.treatment.optimization.v2.ILearningContext,
    *   com.unicacorp.interact.treatment.optimization.v2.IOffer,
    *   com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
    *   com.unicacorp.interact.treatment.optimization.IInteractSession,
boolean)
    */
   public void logEvent(ILearningContext context, IOffer offer,
IClientArgs clientArgs,
  IInteractSession session, boolean debug) throws LearningException
   {
       System.out.println("Calling logEvent for SampleLearning");


       if(context.getLearningContext()==ILearningContext.LOG_AS_CONTACT)
       {
           System.out.println("adding contact");

           // Keep track of all contacts in memory
           synchronized(_offerToAcceptCount)
           {
              Integer count = _offerToAcceptCount.get(offer.getOfferId());
              if(count == null)
                  count = new Integer(1);
              else
                  count++;
              _offerToAcceptCount.put(offer.getOfferId(), ++count);
```

```
            }

        }
        else
 if(context.getLearningContext()==ILearningContext.LOG_AS_ACCEPT)
        {
            System.out.println("adding accept");
            // Keep track of all accept counts in memory by adding to the
 map
            synchronized(_offerToAcceptCount)
            {
               Integer count = _offerToAcceptCount.get(offer.getOfferId());
               if(count == null)
                   count = new Integer(1);
               else
                   count++;
               _offerToAcceptCount.put(offer.getOfferId(), ++count);
            }
            }

    }

    /* (non-Javadoc)
     * @see
 com.unicacorp.interact.treatment.optimization.v2.ILearning#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);

        }
    }

}
```

Kapitel 13. Unica Interact-WSDL

Die Unica Interact-Installation enthält zwei WDSL-XML-Dateien (WSDL = Web Services Description Language), die die verfügbaren Web-Services und die Zugriffsmöglichkeiten darauf beschreiben. Diese Dateien können Sie im Ausgangsverzeichnis von Unica Interact anzeigen. Ein Beispiel wird hier dargestellt.

Nachdem Sie Unica Interact installiert haben, finden Sie die WSDL-Dateien von Unica Interact an den folgenden Positionen:

- `<Interact_home>/conf/InteractService.wsdl`
- `<Interact_home>/conf/InteractAdminService.wsdl`

Die Unica Interact-WDSL kann mit jedem Software-Release und jedem Fixpack geändert werden. Informationen dazu finden Sie in den Unica Interact-Releaseinformationen oder in den Readme-Dateien des Release.

Eine Kopie der `InteractService.wsdl` wird hier als Referenz gezeigt. Wenn Sie sicherstellen wollen, dass Sie die aktuellsten Informationen verwenden, überprüfen Sie die WDSL-Dateien, die mit Unica Interact installiert werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://soap.api.interact.unicacorp.com"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 bloop="http://api.interact.unicacorp.com/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 targetNamespace="http://soap.api.interact.unicacorp.com">
 <wsdl:types>
  <xs:schema xmlns:ns="http://soap.api.interact.unicacorp.com"
 attributeFormDefault="qualified"
```

```
  elementFormDefault="qualified"
targetNamespace="http://soap.api.interact.unicacorp.com">
  <xs:element name="executeBatch">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="1" name="commands"
nillable="false" type="ns1:CommandImpl"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="executeBatchResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:BatchResponse"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="endSession">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="endSessionResponse">
   <xs:complexType>
    <xs:sequence>
```

```
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="getOffers">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
     <xs:element minOccurs="1" name="iPoint" nillable="false"
type="xs:string"/>
     <xs:element minOccurs="1" name="numberRequested" type="xs:int"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="getOffersResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="getProfile">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
```

```
  <xs:element name="getProfileResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="getVersionResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="postEvent">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
     <xs:element minOccurs="1" name="eventName" nillable="false"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="1"
name="eventParameters"
      nillable="true" type="ns1:NameValuePairImpl"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="postEventResponse">
   <xs:complexType>
    <xs:sequence>
```

```
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="setAudience">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID"
nillable="false" type="ns1:NameValuePairImpl"/>
     <xs:element minOccurs="1" name="audienceLevel" nillable="false"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="1" name="parameters"
nillable="true" type="ns1:NameValuePairImpl"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="setAudienceResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="setDebug">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
```

```
     <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="setDebugResponse">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="startSession">
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="sessionID" nillable="false"
type="xs:string"/>
     <xs:element minOccurs="1" name="relyOnExistingSession"
type="xs:boolean"/>
     <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
     <xs:element minOccurs="1" name="interactiveChannel" nillable="false"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID"
nillable="false" type="ns1:NameValuePairImpl"/>
     <xs:element minOccurs="1" name="audienceLevel" nillable="false"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="1" name="parameters"
nillable="true" type="ns1:NameValuePairImpl"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="startSessionResponse">
```

```
   <xs:complexType>
    <xs:sequence>
     <xs:element minOccurs="1" name="return" nillable="false"
type="ns1:Response"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:schema>
 <xs:schema xmlns:ax21="http://api.interact.unicacorp.com/xsd"
attributeFormDefault="qualified"
  elementFormDefault="qualified"
targetNamespace="http://api.interact.unicacorp.com/xsd">
  <xs:complexType name="Command">
   <xs:sequence>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID"
nillable="true" type="ax21:NameValuePair"/>
    <xs:element minOccurs="1" name="audienceLevel" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
    <xs:element minOccurs="1" name="event" nillable="true"
type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="eventParameters"
nillable="true" type="ax21:NameValuePair"/>
    <xs:element minOccurs="1" name="interactionPoint" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="interactiveChannel" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="methodIdentifier" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="numberRequested" type="xs:int"/>
    <xs:element minOccurs="1" name="relyOnExistingSession"
type="xs:boolean"/>
```

```
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="NameValuePair">
   <xs:sequence>
    <xs:element minOccurs="1" name="name" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="valueAsDate" nillable="true"
type="xs:dateTime"/>
    <xs:element minOccurs="1" name="valueAsNumeric" nillable="true"
type="xs:double"/>
    <xs:element minOccurs="1" name="valueAsString" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="valueDataType" nillable="true"
type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CommandImpl">
   <xs:sequence>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="audienceID"
nillable="true" type="ax21:NameValuePairImpl"/>
    <xs:element minOccurs="1" name="audienceLevel" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="debug" type="xs:boolean"/>
    <xs:element minOccurs="1" name="event" nillable="true"
type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="1" name="eventParameters"
nillable="true" type="ax21:NameValuePairImpl"/>
    <xs:element minOccurs="1" name="interactionPoint" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="interactiveChannel" nillable="true"
type="xs:string"/>
```

```
    <xs:element minOccurs="1" name="methodIdentifier" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="numberRequested" type="xs:int"/>
    <xs:element minOccurs="1" name="relyOnExistingSession"
type="xs:boolean"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="NameValuePairImpl">
   <xs:sequence>
    <xs:element minOccurs="1" name="name" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="valueAsDate" nillable="true"
type="xs:dateTime"/>
    <xs:element minOccurs="1" name="valueAsNumeric" nillable="true"
type="xs:double"/>
    <xs:element minOccurs="1" name="valueAsString" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="1" name="valueDataType" nillable="true"
type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="BatchResponse">
   <xs:sequence>
    <xs:element minOccurs="0" name="batchStatusCode" type="xs:int"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="responses"
nillable="false" type="ax21:Response"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Response">
   <xs:sequence>
    <xs:element maxOccurs="unbounded" minOccurs="0"
name="advisoryMessages" nillable="true" type="ax21:AdvisoryMessage"/>
```

```
    <xs:element minOccurs="0" name="apiVersion" nillable="false"
type="xs:string"/>
    <xs:element minOccurs="0" name="offerList" nillable="true"
type="ax21:OfferList"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="profileRecord"
nillable="true" type="ax21:NameValuePair"/>
    <xs:element minOccurs="0" name="sessionID" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="0" name="statusCode" type="xs:int"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AdvisoryMessage">
   <xs:sequence>
    <xs:element minOccurs="0" name="detailMessage" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="0" name="message" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="0" name="messageCode" type="xs:int"/>
    <xs:element minOccurs="0" name="statusLevel" type="xs:int"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OfferList">
   <xs:sequence>
    <xs:element minOccurs="0" name="defaultString" nillable="true"
type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="0"
name="recommendedOffers" nillable="true" type="ax21:Offer"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Offer">
   <xs:sequence>
```

```
    <xs:element maxOccurs="unbounded" minOccurs="0"
name="additionalAttributes" nillable="true" type="ax21:NameValuePair"/>
    <xs:element minOccurs="0" name="description" nillable="true"
type="xs:string"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="offerCode"
nillable="true" type="xs:string"/>
    <xs:element minOccurs="0" name="offerName" nillable="true"
type="xs:string"/>
    <xs:element minOccurs="0" name="score" type="xs:int"/>
    <xs:element minOccurs="0" name="treatmentCode" nillable="true"
type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:schema>
</wsdl:types>
<wsdl:message name="setAudienceRequest">
 <wsdl:part name="parameters" element="ns0:setAudience"/>
</wsdl:message>
<wsdl:message name="setAudienceResponse">
 <wsdl:part name="parameters" element="ns0:setAudienceResponse"/>
</wsdl:message>
<wsdl:message name="postEventRequest">
 <wsdl:part name="parameters" element="ns0:postEvent"/>
</wsdl:message>
<wsdl:message name="postEventResponse">
 <wsdl:part name="parameters" element="ns0:postEventResponse"/>
</wsdl:message>
<wsdl:message name="getOffersRequest">
 <wsdl:part name="parameters" element="ns0:getOffers"/>
</wsdl:message>
<wsdl:message name="getOffersResponse">
 <wsdl:part name="parameters" element="ns0:getOffersResponse"/>
```

```
</wsdl:message>
<wsdl:message name="startSessionRequest">
 <wsdl:part name="parameters" element="ns0:startSession"/>
</wsdl:message>
<wsdl:message name="startSessionResponse">
 <wsdl:part name="parameters" element="ns0:startSessionResponse"/>
</wsdl:message>
<wsdl:message name="getVersionRequest"/>
<wsdl:message name="getVersionResponse">
 <wsdl:part name="parameters" element="ns0:getVersionResponse"/>
</wsdl:message>
<wsdl:message name="setDebugRequest">
 <wsdl:part name="parameters" element="ns0:setDebug"/>
</wsdl:message>
<wsdl:message name="setDebugResponse">
 <wsdl:part name="parameters" element="ns0:setDebugResponse"/>
</wsdl:message>
<wsdl:message name="executeBatchRequest">
 <wsdl:part name="parameters" element="ns0:executeBatch"/>
</wsdl:message>
<wsdl:message name="executeBatchResponse">
 <wsdl:part name="parameters" element="ns0:executeBatchResponse"/>
</wsdl:message>
<wsdl:message name="getProfileRequest">
 <wsdl:part name="parameters" element="ns0:getProfile"/>
</wsdl:message>
<wsdl:message name="getProfileResponse">
 <wsdl:part name="parameters" element="ns0:getProfileResponse"/>
</wsdl:message>
<wsdl:message name="endSessionRequest">
 <wsdl:part name="parameters" element="ns0:endSession"/>
</wsdl:message>
```

```
<wsdl:message name="endSessionResponse">
 <wsdl:part name="parameters" element="ns0:endSessionResponse"/>
</wsdl:message>
<wsdl:portType name="InteractServicePortType">
 <wsdl:operation name="setAudience">
  <wsdl:input message="ns0:setAudienceRequest"
wsaw:Action="urn:setAudience"/>
  <wsdl:output message="ns0:setAudienceResponse"
wsaw:Action="urn:setAudienceResponse"/>
 </wsdl:operation>
 <wsdl:operation name="postEvent">
  <wsdl:input message="ns0:postEventRequest" wsaw:Action="urn:postEvent"/>
  <wsdl:output message="ns0:postEventResponse"
wsaw:Action="urn:postEventResponse"/>
 </wsdl:operation>
 <wsdl:operation name="getOffers">
  <wsdl:input message="ns0:getOffersRequest" wsaw:Action="urn:getOffers"/>
  <wsdl:output message="ns0:getOffersResponse"
wsaw:Action="urn:getOffersResponse"/>
 </wsdl:operation>
 <wsdl:operation name="startSession">
  <wsdl:input message="ns0:startSessionRequest"
wsaw:Action="urn:startSession"/>
  <wsdl:output message="ns0:startSessionResponse"
wsaw:Action="urn:startSessionResponse"/>
 </wsdl:operation>
 <wsdl:operation name="getVersion">
  <wsdl:input message="ns0:getVersionRequest"
wsaw:Action="urn:getVersion"/>
  <wsdl:output message="ns0:getVersionResponse"
wsaw:Action="urn:getVersionResponse"/>
 </wsdl:operation>
```

```
 <wsdl:operation name="setDebug">
  <wsdl:input message="ns0:setDebugRequest" wsaw:Action="urn:setDebug"/>
  <wsdl:output message="ns0:setDebugResponse"
wsaw:Action="urn:setDebugResponse"/>
 </wsdl:operation>
 <wsdl:operation name="executeBatch">
  <wsdl:input message="ns0:executeBatchRequest"
wsaw:Action="urn:executeBatch"/>
  <wsdl:output message="ns0:executeBatchResponse"
wsaw:Action="urn:executeBatchResponse"/>
 </wsdl:operation>
 <wsdl:operation name="getProfile">
  <wsdl:input message="ns0:getProfileRequest"
wsaw:Action="urn:getProfile"/>
  <wsdl:output message="ns0:getProfileResponse"
wsaw:Action="urn:getProfileResponse"/>
 </wsdl:operation>
 <wsdl:operation name="endSession">
  <wsdl:input message="ns0:endSessionRequest"
wsaw:Action="urn:endSession"/>
  <wsdl:output message="ns0:endSessionResponse"
wsaw:Action="urn:endSessionResponse"/>
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="InteractServiceSOAP11Binding"
type="ns0:InteractServicePortType">
 <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
 <wsdl:operation name="setAudience">
  <soap:operation soapAction="urn:setAudience" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
```

```
 </wsdl:input>
 <wsdl:output>
  <soap:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="postEvent">
 <soap:operation soapAction="urn:postEvent" style="document"/>
 <wsdl:input>
  <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getOffers">
 <soap:operation soapAction="urn:getOffers" style="document"/>
 <wsdl:input>
  <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="startSession">
 <soap:operation soapAction="urn:startSession" style="document"/>
 <wsdl:input>
  <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
```

```
<wsdl:operation name="getVersion">
 <soap:operation soapAction="urn:getVersion" style="document"/>
 <wsdl:input>
  <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="setDebug">
 <soap:operation soapAction="urn:setDebug" style="document"/>
 <wsdl:input>
  <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="executeBatch">
 <soap:operation soapAction="urn:executeBatch" style="document"/>
 <wsdl:input>
  <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getProfile">
 <soap:operation soapAction="urn:getProfile" style="document"/>
 <wsdl:input>
  <soap:body use="literal"/>
 </wsdl:input>
```

```
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="endSession">
  <soap:operation soapAction="urn:endSession" style="document"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="InteractServiceSOAP12Binding"
type="ns0:InteractServicePortType">
 <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
 <wsdl:operation name="setAudience">
  <soap12:operation soapAction="urn:setAudience" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap12:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="postEvent">
  <soap12:operation soapAction="urn:postEvent" style="document"/>
  <wsdl:input>
   <soap12:body use="literal"/>
  </wsdl:input>
```

```
 <wsdl:output>
  <soap12:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getOffers">
 <soap12:operation soapAction="urn:getOffers" style="document"/>
 <wsdl:input>
  <soap12:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap12:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="startSession">
 <soap12:operation soapAction="urn:startSession" style="document"/>
 <wsdl:input>
  <soap12:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap12:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getVersion">
 <soap12:operation soapAction="urn:getVersion" style="document"/>
 <wsdl:input>
  <soap12:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap12:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="setDebug">
```

```
 <soap12:operation soapAction="urn:setDebug" style="document"/>
 <wsdl:input>
  <soap12:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap12:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="executeBatch">
 <soap12:operation soapAction="urn:executeBatch" style="document"/>
 <wsdl:input>
  <soap12:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap12:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getProfile">
 <soap12:operation soapAction="urn:getProfile" style="document"/>
 <wsdl:input>
  <soap12:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
  <soap12:body use="literal"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="endSession">
 <soap12:operation soapAction="urn:endSession" style="document"/>
 <wsdl:input>
  <soap12:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
```

```
    <soap12:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="InteractServiceHttpBinding"
type="ns0:InteractServicePortType">
 <http:binding verb="POST"/>
 <wsdl:operation name="setAudience">
  <http:operation location="InteractService/setAudience"/>
  <wsdl:input>
   <mime:content part="setAudience" type="text/xml"/>
  </wsdl:input>
  <wsdl:output>
   <mime:content part="setAudience" type="text/xml"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="postEvent">
  <http:operation location="InteractService/postEvent"/>
  <wsdl:input>
   <mime:content part="postEvent" type="text/xml"/>
  </wsdl:input>
  <wsdl:output>
   <mime:content part="postEvent" type="text/xml"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="getOffers">
  <http:operation location="InteractService/getOffers"/>
  <wsdl:input>
   <mime:content part="getOffers" type="text/xml"/>
  </wsdl:input>
  <wsdl:output>
   <mime:content part="getOffers" type="text/xml"/>
```

```
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="startSession">
 <http:operation location="InteractService/startSession"/>
 <wsdl:input>
  <mime:content part="startSession" type="text/xml"/>
 </wsdl:input>
 <wsdl:output>
  <mime:content part="startSession" type="text/xml"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getVersion">
 <http:operation location="InteractService/getVersion"/>
 <wsdl:input>
  <mime:content part="getVersion" type="text/xml"/>
 </wsdl:input>
 <wsdl:output>
  <mime:content part="getVersion" type="text/xml"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="setDebug">
 <http:operation location="InteractService/setDebug"/>
 <wsdl:input>
  <mime:content part="setDebug" type="text/xml"/>
 </wsdl:input>
 <wsdl:output>
  <mime:content part="setDebug" type="text/xml"/>
 </wsdl:output>
</wsdl:operation>
<wsdl:operation name="executeBatch">
 <http:operation location="InteractService/executeBatch"/>
 <wsdl:input>
```

```
    <mime:content part="executeBatch" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="executeBatch" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getProfile">
   <http:operation location="InteractService/getProfile"/>
   <wsdl:input>
    <mime:content part="getProfile" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="getProfile" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="endSession">
   <http:operation location="InteractService/endSession"/>
   <wsdl:input>
    <mime:content part="endSession" type="text/xml"/>
   </wsdl:input>
   <wsdl:output>
    <mime:content part="endSession" type="text/xml"/>
   </wsdl:output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="InteractService">
  <wsdl:port name="InteractServiceSOAP11port_http"
 binding="ns0:InteractServiceSOAP11Binding">
   <soap:address
 location="http://localhost:7001/interact/services/InteractService"/>
  </wsdl:port>
```

```
  <wsdl:port name="InteractServiceSOAP12port_http"
 binding="ns0:InteractServiceSOAP12Binding">
   <soap12:address
 location="http://localhost:7001/interact/services/InteractService"/>
  </wsdl:port>
  <wsdl:port name="InteractServiceHttpport"
 binding="ns0:InteractServiceHttpBinding">
   <http:address
 location="http://localhost:7001/interact/services/InteractService"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Kapitel 14. Unica Interact Laufzeitumgebung - Konfigurationseigenschaften

In diesem Abschnitt werden alle Konfigurationseigenschaften für die Unica Interact-Laufzeitumgebung beschrieben.

Interact | Allgemein

Diese Konfigurationseigenschaften definieren allgemeine Einstellungen für Ihre Laufzeitumgebung, einschließlich der Standardprotokollebene und Ländereinstellung.

log4jConfig

Syntax

Die Position der Datei, die die log4j-Eigenschaften enthält. Dieser Pfad kann entweder absolut oder relativ zur Umgebungsvariablen `INTERACT_HOME` sein. `INTERACT_HOME` ist die Position des Unica Interact-Installationsverzeichnisses.

Standardwert

`./conf/interact_log4j2.xml`

asmUserForDefaultLocale

Syntax

Die Eigenschaft `asmUserForDefaultLocale` legt den Unica-Benutzer fest, von dem Unica Interact die Ländereinstellungen ableitet.

Die Ländereinstellungen definieren, welche Sprache in der Design- und Laufzeit angezeigt wird und in welcher Sprache nützliche Hinweise von der Unica Interact-API erstellt werden. Wenn die Ländereinstellung nicht mit den Einstellungen des Betriebssystems Ihres Computers übereinstimmt, funktioniert Unica Interact trotzdem, aber möglicherweise werden nützliche Hinweise in einer anderen Sprache erstellt, als in der Designumgebung verwendet wird.

Standardwert

`asm_admin`

configurationRefreshInMins

Syntax

Der Adminbenutzer mit Adminrolle (Platform) kann die Funktion zur dynamischen Aktualisierung der Konfigurationsdaten ein- oder ausschalten, indem er den Wert für die Aktualisierung der Konfiguration in Minuten <=0 (aus) oder >0 (ein) einstellt. Der aktuelle Wert der Konfigurationsaktualisierung in Minuten beträgt beispielsweise 5 Minuten. Es kann bis zu 5 Minuten dauern, bis die Konfigurationsänderungen wirksam werden. Der neu eingestellte Wert der Konfigurationsaktualisierung wird dann in Minutenschnelle wirksam. Das System aktualisiert danach einige der Konfigurationsdaten nach dem neuen geänderten Wert.

Anmerkung: Die automatische Aktualisierung wird bei den folgenden Konfigurationsparametern nicht unterstützt:

1. Allgemein – keine Unterstützung für alle Datenquellenkonfigurationen, nur Unterstützung für die oberste Konfigurationsebene.
2. Überwachung
3. Profil
4. cacheManagement
5. triggedMessage
6. activityOrchestrator
7. Simulator
8. ETL

Interact | Allgemein | API

Diese Konfigurationseigenschaft definiert die Einstellung für die Authentifizierung der Unica Interact-APIs.

tokenAuthentication

Beschreibung

Mit dieser tokenAuthentication-Eigenschaft können Sie die Authentifizierung für die Unica Interact-APIs aktivieren oder inaktivieren.

Standardwert

FALSE

Gültige Werte

TRUE | FALSE

enabledLogging

Beschreibung

Verwenden Sie diese Eigenschaft, um zu bestimmen, ob die API-Protokollierung global aktiviert werden soll.

Standardwert

FALSE

Gültige Werte

TRUE | FALSE

Interact | Allgemein | centralizedLogger

Diese Konfigurationseigenschaft definiert die zentralisierte Protokollierung für Unica Interact.

Aktiviert

Syntax

Definiert, ob der zentralisierte Logger in Unica Interact aktiviert oder inaktiviert ist.

Standardwert

FALSE

Gültige Werte

TRUE | FALSE

maxBatchSize

Syntax

Die maximale Anzahl von Protokolleinträgen, die in einem einzigen Batch in der Datenbank persistiert werden.

Standardwert

1000

Gültige Werte

Nicht-negative und positive Ganzzahlen.

maxDelayInSec

Syntax

Die maximale Wartezeit für einen Protokolleintrag, bevor er in der Datenbank persistiert wird.

Standardwert

15

Gültige Werte

Nicht-negative und positive Ganzzahlen.

Selbst in Fällen, in denen die centralizedLogger-Funktion für die gesamte Servergruppe aktiviert ist, kann sie auf einem einzelnen Server durch Einfügen eines Kommentars in die interact_log4j2.xml dieses Servers inaktiviert werden:

```
<AppenderRef ref="db"/>
```

Interact | Allgemein | learningTablesDataSource

Diese Konfigurationseigenschaften definieren die Datenquelleneinstellungen für die integrierten Lerntabellen. Sie müssen diese Datenquelle definieren, wenn Sie das integrierte Lernmodul von Unica Interact verwenden.

Wenn Sie mit der Lern-API eine eigene Implementierung des Lernmoduls erstellen, können Sie Ihr benutzerdefiniertes Lernmodul so konfigurieren, dass diese Werte mithilfe der `ILearningConfig`-Benutzeroberfläche gelesen werden.

jndiName

Beschreibung

Verwenden Sie diese `jndiName`-Eigenschaft, um die Java™ Naming and Directory Interface (JNDI) Datenquelle zu identifizieren, die auf dem Anwendungsserver (Websphere oder WebLogic) für die Lerntabellen definiert ist, auf die die Laufzeitserver von Unica Interact zugreifen.

Die Lerntabellen werden von der DLL-Datei `aci_lrntab` erstellt und enthalten (u.a.) die Tabellen: `UACI_AttributeValue` und `UACI_OfferStats`.

Standardwert

Es ist kein Standardwert definiert.

Typ

Beschreibung

Der Datenbanktyp für die Datenquelle, die von den Lerntabellen verwendet wird, auf die die Laufzeitserver von Unica Interact zugreifen.

Die Lerntabellen werden von der DLL-Datei `aci_lrntab` erstellt und enthalten (u.a.) die Tabellen: `UACI_AttributeValue` und `UACI_OfferStats`.

Standardwert

SQL-Server

Gültige Werte

SQLServer | DB2® | ORACLE | MARIADB

connectionRetryPeriod

Beschreibung

Die Eigenschaft `ConnectionRetryPeriod` gibt die Zeitspanne in Sekunden an, in der Unica Interact versucht, die Datenbankverbindungsanforderung erneut zu stellen, wenn ein Fehler bei den Lerntabellen aufgetreten ist. Unica Interact versucht automatisch, die Verbindung zur Datenbank für diese Zeitspanne erneut herzustellen, bevor ein Datenbankfehler oder -ausfall gemeldet wird. Wird der Wert auf 0 gesetzt, versucht Unica Interact unbegrenzt, die Verbindung wiederherzustellen. Wenn -1 festgelegt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Die Lerntabellen werden von der DLL-Datei `aci_lrntab` erstellt und enthalten (u.a.) die Tabellen: `UACI_AttributeValue` und `UACI_OfferStats`.

Standardwert

-1

connectionRetryDelay

Beschreibung

Die Eigenschaft `ConnectionRetryDelay` gibt in Sekunden an, wie lange Unica Interact wartet, bevor versucht wird, die Verbindung zur Datenbank wiederherzustellen, wenn ein Fehler bei den Lerntabellen aufgetreten ist. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Die Lerntabellen werden von der DLL-Datei `aci_lrntab` erstellt und enthalten (u.a.) die Tabellen: `UACI_AttributeValue` und `UACI_OfferStats`.

Standardwert

-1

Schema

Beschreibung

Der Name des Schemas, das die Tabellen für das eingebaute Lernmodul enthält. Unica Interact fügt den Wert für diese Eigenschaft vor

allen Tabellennamen ein. Beispielsweise wird `UACI_IntChannel` zu `schema.UACI_IntChannel`.

Sie müssen kein Schema definieren. Wenn Sie kein Schema angeben, geht Unica Interact davon aus, dass der Eigner der Tabellen mit dem Schema übereinstimmt. Sie sollten diesen Wert festlegen, um Mehrdeutigkeit zu vermeiden.

Standardwert

Es ist kein Standardwert definiert.

Interact | Allgemein | prodUserDataSource

Diese Konfigurationseigenschaften definieren die Datenquelleneinstellungen für die Produktionsprofiltabellen. Sie müssen diese Datenquelle definieren. Auf diese Datenquelle verweist die Laufzeitumgebung beim Ausführen der interaktiven Ablaufdiagramme nach der Bereitstellung.

jndiName

Beschreibung

Verwenden Sie diese `jndiName`-Eigenschaft, um die Java™ Naming and Directory Interface (JNDI) Datenquelle zu identifizieren, die auf dem Anwendungsserver (Websphere oder WebLogic) für die Kundentabellen definiert ist, auf die die Laufzeitserver von Unica Interact zugreifen.

Standardwert

Es ist kein Standardwert definiert.

Typ

Beschreibung

Der Datenbanktyp für die Kundentabellen, auf die Laufzeitserver in Unica Interact zugreifen.

Standardwert

SQL-Server

Gültige Werte

SQLServer | DB2® | ORACLE | MARIADB

aliasPrefix

Beschreibung

Die Eigenschaft `AliasPrefix` gibt an, wie Unica Interact den neuen Aliasnamen bildet, der automatisch von Unica Interact erstellt wird, wenn eine Dimensionstabelle verwendet in eine neue Tabelle in den Kundentabellen geschrieben wird, auf die die Unica Interact-Laufzeitserver zugreifen.

Für jede Datenbank gilt eine maximale ID-Länge. Lesen Sie die Dokumentation für die von Ihnen verwendete Datenbank, um sicherzustellen, dass Sie keinen Wert festlegen, der die maximale ID-Länge für Ihre Datenbank überschreitet.

Standardwert

Z

connectionRetryPeriod

Beschreibung

Die Eigenschaft `ConnectionRetryPeriod` gibt die Zeitspanne in Sekunden an, in der Unica Interact versucht, die Datenbankverbindungsanforderung erneut zu stellen, wenn ein Fehler bei Laufzeit-Kundentabellen aufgetreten ist. Unica Interact versucht automatisch, die Verbindung zur Datenbank für diese Zeitspanne erneut herzustellen, bevor ein Datenbankfehler oder -ausfall gemeldet wird. Wird der Wert auf 0 gesetzt, versucht Unica Interact unbegrenzt, die Verbindung wiederherzustellen. Wenn -1 festgelegt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

connectionRetryDelay

Beschreibung

Die Eigenschaft `ConnectionRetryDelay` gibt in Sekunden an, wie lange Unica Interact wartet, bevor versucht wird, die Verbindung zur Datenbank wiederherzustellen, wenn ein Fehler bei den Unica Interact-Laufzeit-Kundentabellen aufgetreten ist. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

Schema

Beschreibung

Der Name des Schemas, das Ihre Profildatentabellen enthält. Unica Interact fügt den Wert für diese Eigenschaft vor allen Tabellennamen ein. Beispielsweise wird `UACI_IntChannel` zu `schema.UACI_IntChannel`.

Sie müssen kein Schema definieren. Wenn Sie kein Schema angeben, geht Unica Interact davon aus, dass der Eigner der Tabellen mit dem Schema übereinstimmt. Sie sollten diesen Wert festlegen, um Mehrdeutigkeit zu vermeiden.

Wenn Sie eine DB2-Datenbank verwenden, muss der Schemaname in Großbuchstaben geschrieben sein.

Standardwert

Es ist kein Standardwert definiert.

Interact | general | API | requestThreadPool

Diese Konfigurationseigenschaft konfiguriert den Thread Pool, der zusätzliche Verfahren von anderen Zielgruppen-IDs abruft, die in UACISupplementAudience angegeben sind.

corePoolSize

Beschreibung

Die Mindestanzahl von Threads in diesem Pool.

Standardwert

Keine

Gültige Werte

Numerisch

maxPoolSize

Die maximale Anzahl von Threads in diesem Pool.

Standardwert

Keine

Gültige Werte

Numerisch

keepAliveTimeSecs

Die maximale Wartezeit in Sekunden, bevor das System einen inaktiven Thread aus diesem Pool entfernt.

Standardwert

Keine

Gültige Werte

Numerisch

termWaitSecs

Die maximale Wartezeit in Sekunden, bevor das System einen Thread in diesem Pool stoppt, wenn das System heruntergefahren wird.

Standardwert

Keine

Gültige Werte

Numerisch

Interact | Allgemein | systemTablesDataSource

Diese Konfigurationseigenschaften definieren die Datenquelleneinstellungen für die Systemtabellen für die Laufzeitumgebung. Sie müssen diese Datenquelle definieren.

jndiName

Beschreibung

Verwenden Sie diese `jndiName`-Eigenschaft, um die Java™-Datenquelle "Naming and Directory Interface" (JNDI) zu identifizieren, die auf dem Anwendungsserver (Websphere oder WebLogic) für die Laufzeitumgebungstabellen definiert ist.

Die Laufzeitdatenbank ist die mit den DLL-Scripts `aci_runtime` und `aci_populate_runtime` gefüllte Datenbank und enthält beispielsweise (u.a.) die folgenden Tabellen: `UACI_CHOfferAttrib` und `UACI_DefaultedStat`.

Standardwert

Es ist kein Standardwert definiert.

Typ

Beschreibung

Der Typ der Datenbank für die Systemtabellen für die Laufzeitumgebung.

Die Laufzeitdatenbank ist die mit den DLL-Scripts `aci_runtime` und `aci_populate_runtime` gefüllte Datenbank und enthält beispielsweise (u.a.) die folgenden Tabellen: `UACI_CHOfferAttrib` und `UACI_DefaultedStat`.

Standardwert

SQL-Server

Gültige Werte

SQLServer | DB2® | ORACLE | MARIADB

connectionRetryPeriod

Beschreibung

Die Eigenschaft `ConnectionRetryPeriod` gibt die Zeitspanne in Sekunden an, in der Unica Interact versucht, die Datenbankverbindungsanforderung erneut zu stellen, wenn ein Fehler bei bei den Laufzeitsystemtabellen aufgetreten ist. Unica Interact versucht automatisch, die Verbindung zur Datenbank für diese Zeitspanne erneut herzustellen, bevor ein Datenbankfehler oder -ausfall gemeldet wird. Wird der Wert auf 0 gesetzt, versucht Unica Interact unbegrenzt, die Verbindung wiederherzustellen. Wenn -1 festgelegt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Die Laufzeitdatenbank ist die mit den DLL-Scripts `aci_runtime` und `aci_populate_runtime` gefüllte Datenbank und enthält beispielsweise (u.a.) die folgenden Tabellen: `UACI_CHOfferAttrib` und `UACI_DefaultedStat`.

Standardwert

-1

connectionRetryDelay

Beschreibung

Die Eigenschaft `ConnectionRetryDelay` gibt in Sekunden an, wie lange Unica Interact wartet, bevor versucht wird, die Verbindung zur Datenbank wiederherzustellen, wenn ein Fehler bei den Unica Interact-Laufzeitsystemtabellen aufgetreten ist. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Die Laufzeitdatenbank ist die mit den DLL-Scripts `aci_runtime` und `aci_populate_runtime` gefüllte Datenbank und enthält beispielsweise (u.a.) die folgenden Tabellen: `UACI_CHOfferAttrib` und `UACI_DefaultedStat`.

Standardwert

-1

Schema

Beschreibung

Der Name des Schemas, das die Tabellen für die Laufzeitumgebung enthält. Unica Interact fügt den Wert für diese Eigenschaft vor allen Tabellennamen ein. Beispielsweise wird `UACI_IntChannel` zu `schema.UACI_IntChannel`.

Sie müssen kein Schema definieren. Wenn Sie kein Schema angeben, geht Unica Interact davon aus, dass der Eigner der Tabellen mit dem Schema übereinstimmt. Sie sollten diesen Wert festlegen, um Mehrdeutigkeit zu vermeiden.

Standardwert

Es ist kein Standardwert definiert.

Interact | Allgemein | systemTablesDataSource | loaderProperties

Diese Konfigurationseigenschaften definieren die Einstellungen des Datenbankladeprogramms für die Systemtabellen für die Laufzeitumgebung. Sie müssen diese Eigenschaften nur definieren, wenn Sie ein Datenbankladeprogramm verwenden.

databaseName

Syntax

Der Name der Datenbank, mit der das Datenbankladeprogramm verbunden ist.

Standardwert

Es ist kein Standardwert definiert.

LoaderCommandForAppend

Syntax

Der Parameter `LoaderCommandForAppend` gibt den Befehl an, der zum Aufrufen des Datenbankladeprogramms für das Anhängen von Datensätzen zu den Staging-Datenbanktabellen für den Kontakt- und Antwortverlauf in Unica Interact dient. Sie müssen diesen Parameter festlegen, um das Datenbankladeprogramm für die Kontakt- und Antwortverlaufsdaten zu aktivieren.

Dieser Parameter wird als vollständiger Pfadname zur ausführbaren Datei des Datenbankladeprogramms oder zu einem Script, das das Datenbankladeprogramm startet, angegeben. Durch die Verwendung eines Scripts können Sie zusätzliche Konfigurationsvorgänge ausführen, bevor Sie das Ladedienstprogramm starten.

Für den Start der meisten Datenbankladeprogramme sind mehrere Argumente erforderlich. Diese können u. a. die Daten- und Steuerdatei, aus der geladen werden soll, und die Datenbank und Tabelle, in die geladen werden soll, angeben. Die Token werden bei der Ausführung des Befehls durch die festgelegten Elemente ersetzt.

Informieren Sie sich in der Dokumentation zu Ihrem Datenbankladeprogramm über die korrekte Syntax, die Sie für den Start des Ladedienstprogramms verwenden müssen.

Dieser Parameter ist standardmäßig nicht definiert.

Die für `LoaderCommandForAppend` verfügbaren Token werden in der folgenden Tabelle beschrieben.

Token	Syntax
<CONTROLFILE>	Dieses Token wird durch den vollständigen Pfad und Dateinamen der temporären Steuerdatei ersetzt, die von Unica Interact gemäß der im Parameter `LoaderControlFileTemplate` angegebenen Vorlage generiert wird.l
<DATABASE>	Dieses Token wird durch den Namen der Datenquelle ersetzt, in die Unica Interact Daten lädt. Dies ist derselbe Datenquellenname, der im Kategorienamen für diese Datenquelle verwendet wird.
<DATAFILE>	Dieses Token wird durch den vollständigen Pfad und Dateinamen der temporären Datendatei ersetzt, die von Unica Interact während des Ladevorgangs erstellt wird. Diese Datei

Token	Syntax
	befindet sich im Temp-Verzeichnis von Unica Interact: `UNICA_ACTMPDIR`.
<DBCOLUMNNUMBER>	Dieses Token wird durch die Spaltenordnungszahl in der Datenbank ersetzt.
<FIELDLENGTH>	Dieses Token wird durch die Länge des in die Datenbank geladenen Felds ersetzt.
<FIELDNAME>	Dieses Token wird durch den Namen des in die Datenbank geladenen Felds ersetzt.
<FIELDNUMBER>	Dieses Token wird durch die Nummer des in die Datenbank geladenen Felds ersetzt.
<FIELDTYPE>	Dieses Token wird durch den Literalwert "CHAR( )" ersetzt. Die Länge des Felds wird in den Klammern () angegeben. Wenn die Datenbank den Feldtyp CHAR nicht verstehen kann, können Sie den entsprechenden Text für den Feldtyp manuell angeben und das Token <FIELDLENGTH> verwenden. Für SQLSVR und SQL2000 würden Sie beispielsweis SQLCHAR(<FIELDLENGTH>) verwenden.
<NATIVETYPE>	Dieses Token wird durch den Typ der Datenbank ersetzt, in die das Feld geladen wird.
<NUMFIELDS>	Dieses Token wird durch die Anzahl der Felder in der Tabelle ersetzt.
<KENNWORT>	Dieses Token wird mit dem Datenbankkennwort von der aktuellen Ablaufdiagrammverbindung zur Datenquelle ersetzt.
<TABLENAME>	Dieses Token wird durch den Namen der Datenbanktabelle ersetzt, in die Unica Interact Daten lädt.

Token	Syntax
<USER>	Dieses Token wird mit dem Datenbankbenutzer der aktuellen Ablaufdiagrammverbindung zur Datenquelle ersetzt.

Standardwert

Es ist kein Standardwert definiert.

LoaderControlFileTemplateForAppend

Syntax

Die Eigenschaft `LoaderControlFileTemplateForAppend` gibt den vollständigen Pfad und Dateinamen der Steuerdateivorlage an, die zuvor in Unica Interact konfiguriert wurde. Wenn dieser Parameter festgelegt ist, erstellt Unica Interact basierend auf der hier angegebenen Vorlage dynamisch eine temporäre Steuerdatei. Der Pfad und Name dieser temporären Steuerdatei stehen dem Token `<CONTROLFILE>` zur Verfügung, das für die Eigenschaft `LoaderCommandForAppend` verfügbar ist.

Vor der Verwendung von Unica Interact im Datenbankladeprogrammmodus müssen Sie die Steuerdateivorlage konfigurieren, die durch diesen Parameter festgelegt wird. Die Steuerdateivorlage unterstützt die folgenden Token, die dynamisch ersetzt werden, wenn die temporäre Steuerdatei von Unica Interact erstellt wird.

Informationen über die richtige Syntax für Ihre Steuerdatei finden Sie in der Dokumentation zu Ihrem Datenbankladeprogramm. Die für die Steuerdateivorlage zur Verfügung stehenden Token sind dieselben wie die für die Eigenschaft `LoaderControlFileTemplate`.

Dieser Parameter ist standardmäßig nicht definiert.

Standardwert

Es ist kein Standardwert definiert.

LoaderDelimiterForAppend

Syntax

Die Eigenschaft `LoaderDelimiterForAppend` gibt an, ob die temporäre Unica Interact-Datendatei eine Flatfile mit fester Breite oder mit Trennzeichen ist. Bei einer Flatfile mit Trennzeichen werden außerdem die Zeichen festgelegt, die als Trennzeichen verwendet werden.

Ist der Wert nicht definiert, erstellt Unica Interact die temporäre Datendatei als Flatfile mit fester Breite.

Wenn Sie einen Wert angeben, wird dieser verwendet, wenn das Ladeprogramm aufgerufen wird, um eine Tabelle zu füllen, die nicht bekanntermaßen leer ist. Unica Interact legt die temporäre Datendatei als abgegrenzte Flachdatei an, wobei der Wert dieser Eigenschaft als Trennzeichen verwendet wird.

Diese Eigenschaft ist standardmäßig nicht definiert.

Standardwert

Gültige Werte

Zeichen, die Sie auf Wunsch in doppelten Anführungszeichen angeben können.

LoaderDelimiterAtEndForAppend

Syntax

Einige externe Ladeprogramme erfordern, dass die Datendatei durch Trennzeichen getrennt ist und jede Zeile mit dem Trennzeichen endet. Um diese Anforderung zu erfüllen, setzen Sie den Wert für `LoaderDelimiterAtEndForAppend` auf `TRUE`. Wenn das Ladeprogramm zum Füllen einer Tabelle aufgerufen wird, von der bekannt ist, dass sie leer ist, verwendet Unica Interact Trennzeichen am Ende jeder Zeile.

Standardwert

FALSE

Gültige Werte

TRUE | FALSE

LoaderUseLocaleDP

Syntax

Die Eigenschaft `LoaderUseLocaleDP` legt fest, ob das länderspezifische Symbol als Dezimalzeichen verwendet wird, wenn Unica Interact numerische Werte in Dateien schreibt, die über ein Datenbankladeprogramm geladen werden sollen.

Setzen Sie diesen Wert auf `FALSE`, um anzugeben, dass der Punkt (.) als Dezimalzeichen verwendet werden soll.

Geben Sie `TRUE` an, um festzulegen, dass das länderspezifische Symbol als Dezimalzeichen verwendet werden soll.

Standardwert

FALSE

Gültige Werte

TRUE | FALSE

Interact | Allgemein | testRunDataSource

Diese Konfigurationseigenschaften definieren die Datenquelleneinstellungen für die Testlauftabellen für die Unica Interact-Designumgebung. Sie müssen diese Datenquelle für mindestens eine der Laufzeitumgebungen definieren. Diese Tabellen werden verwendet, wenn Sie einen Testlauf Ihres interaktiven Ablaufdiagramms durchführen.

jndiName

Beschreibung

Verwenden Sie diese `jndiName`-Eigenschaft, um die Java™ Datenquelle "Naming and Directory Interface" (JNDI) zu identifizieren, die auf dem Anwendungsserver (Websphere oder WebLogic) für die Kundentabellen

definiert ist, auf die die Designumgebung beim Durchführen von interaktiven Ablaufdiagramm-Testläufen zugreift.

Standardwert

Es ist kein Standardwert definiert.

Typ

Beschreibung

Der Datenbanktyp für die Kundentabellen, auf die die Designumgebung beim Durchführen von interaktiven Ablaufdiagramm-Testläufen zugreift.

Standardwert

SQL-Server

Gültige Werte

SQLServer | DB2® | ORACLE | MARIADB

aliasPrefix

Beschreibung

Die Eigenschaft `AliasPrefix` gibt an, wie Unica Interact den neuen Aliasnamen bildet, der automatisch von Unica Interact erstellt wird, wenn eine Dimensionstabelle verwendet wird und in eine neue Tabelle für die Kundentabellen geschrieben wird, auf die die Designumgebung beim Durchführen von interaktiven Ablaufdiagramm-Testläufen zugreift.

Für jede Datenbank gilt eine maximale ID-Länge. Lesen Sie die Dokumentation für die von Ihnen verwendete Datenbank, um sicherzustellen, dass Sie keinen Wert festlegen, der die maximale ID-Länge für Ihre Datenbank überschreitet.

Standardwert

Z

connectionRetryPeriod

Beschreibung

Die Eigenschaft `ConnectionRetryPeriod` gibt die Zeitspanne in Sekunden an, in der Unica Interact versucht, die Datenbankverbindungsanforderung erneut zu stellen, wenn ein Fehler bei den Testlauftabellen aufgetreten ist. Unica Interact versucht automatisch, die Verbindung zur Datenbank für diese Zeitspanne erneut herzustellen, bevor ein Datenbankfehler oder -ausfall gemeldet wird. Wird der Wert auf 0 gesetzt, versucht Unica Interact unbegrenzt, die Verbindung wiederherzustellen. Wenn -1 festgelegt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

connectionRetryDelay

Beschreibung

Die Eigenschaft `ConnectionRetryDelay` gibt in Sekunden an, wie lange Unica Interact wartet, bevor versucht wird, die Verbindung zur Datenbank wiederherzustellen, wenn ein Fehler bei den Testlauftabellen aufgetreten ist. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

Schema

Beschreibung

Der Name des Schemas, das die Tabellen für die interaktiven Ablaufdiagramm-Testläufe der Laufzeitumgebung enthält. Unica Interact fügt den Wert für diese Eigenschaft vor allen Tabellennamen ein, beispielsweise wird `UACI_IntChannel` zu `schema.UACI_IntChannel`.

Sie müssen kein Schema definieren. Wenn Sie kein Schema angeben, geht Unica Interact davon aus, dass der Eigner der Tabellen mit dem Schema

übereinstimmt. Sie sollten diesen Wert festlegen, um Mehrdeutigkeit zu vermeiden.

Standardwert

Es ist kein Standardwert definiert.

Interact | Allgemein | contactAndResponseHistoryDataSource

Diese Konfigurationseigenschaften definieren die Verbindungseinstellungen für die Kontakt- und Antwortverlaufsdatenquelle, die für die Antwortverfolgung in Unica Interact erforderlich ist. Zwischen diesen Einstellungen und dem Kontakt- und Antwortverlaufsmodul besteht keine Verbindung.

jndiName

Beschreibung

Verwenden Sie diese `jndiName`-Eigenschaft, um die Java™ Naming and Directory Interface (JNDI) Datenquelle zu identifizieren, die auf dem Anwendungsserver (WebSphere® oder WebLogic) für die Kontakt- und Antwortverlaufsdatenquelle definiert ist, die für die Antwortverfolgung in Unica Interact erforderlich ist.

Standardwert

Typ

Beschreibung

Der Datenbanktyp für die Datenquelle, die von der Kontakt- und Antwortverlaufsdatenquelle verwendet wird, die für die Antwortverfolgung in Unica Interact erforderlich ist.

Standardwert

SQL-Server

Gültige Werte

SQLServer | DB2® | ORACLE | MARIADB

connectionRetryPeriod

Beschreibung

Die Eigenschaft `ConnectionRetryPeriod` gibt die Zeitspanne in Sekunden an, in der Unica Interact versucht, die Datenbankverbindungsanforderung erneut zu stellen, wenn ein Fehler bei der sitzungsübergreifenden Antwortverfolgung von Unica Interact aufgetreten ist. Unica Interact versucht automatisch, die Verbindung zur Datenbank für diese Zeitspanne erneut herzustellen, bevor ein Datenbankfehler oder -ausfall gemeldet wird. Wird der Wert auf 0 gesetzt, versucht Unica Interact unbegrenzt, die Verbindung wiederherzustellen. Wenn -1 festgelegt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

connectionRetryDelay

Beschreibung

Die Eigenschaft `ConnectionRetryDelay` gibt in Sekunden an, wie lange Unica Interact wartet, bevor versucht wird, die Verbindung zur Datenbank wiederherzustellen, wenn ein Fehler bei der sitzungsübergreifenden Antwortverfolgung von Unica Interact aufgetreten ist. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

Schema

Beschreibung

Der Name des Schemas, das die Tabellen für die sitzungsübergreifende Antwortverfolgung von Unica Interact enthält. Unica Interact fügt den Wert für diese Eigenschaft vor allen Tabellennamen ein, beispielsweise wird `UACI_IntChannel` zu `schema.UACI_IntChannel`.

Sie müssen kein Schema definieren. Wenn Sie kein Schema angeben, geht Unica Interact davon aus, dass der Eigner der Tabellen mit dem Schema übereinstimmt. Sie sollten diesen Wert festlegen, um Mehrdeutigkeit zu vermeiden.

Standardwert

Es ist kein Standardwert definiert.

Interact | Allgemein | idsByType

Diese Konfigurationseigenschaften definieren Einstellungen für ID-Nummern, die vom Kontakt- und Antwortverlaufsmodul verwendet werden.

initialValue

Syntax

Der ursprüngliche ID-Wert, der bei der Erstellung von IDs mit der UACI_IDsByType-Tabelle verwendet wird.

Standardwert

1

Gültige Werte

Ein beliebiger Wert größer 0.

Wiederholungsversuche

Syntax

Die Anzahl der Wiederholungen, bevor eine Ausnahme ausgelöst wird, wenn IDs mit der UACI_IDsByType-Tabelle erstellt werden.

Standardwert

20

Gültige Werte

Eine beliebige Ganzzahl größer 0.

Interact | Ablaufdiagramm

In diesem Abschnitt werden die Konfigurationseinstellungen für interaktive Ablaufdiagramme definiert.

Die Formate DT_DELIM_XXX können nicht mit Ablaufdiagrammen für interaktive Sitzungen verwendet werden.

defaultDateFormat

Syntax

Das Standarddatumsformat, das von Unica Interact zum Konvertieren eines Datums in eine Zeichenfolge bzw. einer Zeichenfolge in ein Datum verwendet wird.

Standardwert

dd/MM/y

idleFlowchartThreadTimeoutInMinutes

Syntax

Die Anzahl von Minuten, die ein Thread, der einem interaktiven Ablaufdiagramm zugewiesen ist, in Unica Interact im Leerlauf sein kann, bevor der Thread freigegeben wird.

Standardwert

5

idleProcessBoxThreadTimeoutInMinutes

Syntax

Die Anzahl von Minuten, die ein Thread, der einem interaktiven Ablaufdiagrammprozess zugewiesen ist, in Unica Interact im Leerlauf sein kann, bevor der Thread freigegeben wird.

Standardwert

5

maxSizeOfFlowchartEngineInboundQueue

Syntax

Die maximale Anzahl der Aufforderungen zum Ausführen eines Ablaufdiagramms, die in Unica Interact in einer Warteschlange gehalten werden. Wenn diese Anzahl erreicht wird, hört Unica Interact auf, Anfragen anzunehmen.

Standardwert

1000

maxNumberOfFlowchartThreads

Syntax

Die maximale Anzahl an Threads, die Anforderungen für interaktive Ablaufdiagramme zugewiesen sind.

Standardwert

25

maxNumberOfProcessBoxThreads

Syntax

Die maximale Anzahl an Threads, die interaktiven Ablaufdiagrammprozessen zugewiesen sind.

Standardwert

50

maxNumberOfProcessBoxThreadsPerFlowchart

Syntax

Die maximale Anzahl an Threads, die interaktiven Ablaufdiagrammprozessen pro Ablaufdiagramminstanz zugewiesen sind.

Standardwert

3

minNumberOfFlowchartThreads

Syntax

Die Mindestanzahl an Threads, die Anforderungen für interaktive Ablaufdiagramme zugewiesen sind.

Standardwert

10

minNumberOfProcessBoxThreads

Syntax

Die Mindestanzahl an Threads, die interaktiven Ablaufdiagrammprozessen zugewiesen sind.

Standardwert

20

sessionVarPrefix

Syntax

Das Präfix für Sitzungsvariablen.

Standardwert

SessionVar

Interact | Ablaufdiagramm| ExternalCallouts | [ExternalCalloutName]

In diesem Abschnitt werden die Klasseneinstellungen für benutzerdefinierte externe Aufrufe definiert, die Sie mit der externen Callout-API geschrieben haben.

Klasse

Syntax

Der Name der Java™-Klasse, die durch diesen externen Aufruf repräsentiert wird.

Dies ist die Java™-Klasse, auf die Sie mit dem -Makro EXTERNALCALLOUT zugreifen können.

Standardwert

Es ist kein Standardwert definiert.

classpath

Syntax

Der Klassenpfad für die Java™-Klasse, die durch diesen externen Aufruf repräsentiert wird. Der Klassenpfad muss auf jar-Dateien auf dem Laufzeitserver verweisen. Wenn Sie eine Servergruppe verwenden und alle Laufzeitserver dieselbe Unica Platform verwenden, muss jeder Server über eine Kopie der jar-Datei an demselben Datenträger verfügen. Der Klassenpfad muss absolute Datenträgere der jar-Dateien enthalten, die durch das Pfadtrennzeichen des Betriebssystems des Servers für die Laufzeitumgebung getrennt sind, z. B. Semikolon (;) bei Windows™-Systemen und Doppelpunkt (:) bei UNIX™-Systemen. Verzeichnisse, die Klassendateien enthalten, sind nicht zulässig. Beispielsweise unter einem Unix-Betriebssystem: `/path1/file1.jar:/path2/file2.jar`.

Dieser Klassenpfad kann maximal 1024Zeichen enthalten. Mit der Manifestdatei in einer jar-Datei können Sie andere jar-Dateien angeben, sodass im Klassenpfad nur eine jar-Datei enthalten sein muss.

Dies ist die Java™-Klasse, auf die Sie mit dem -Makro EXTERNALCALLOUT zugreifen können.

Wenn mehrere JARs in dieser Einstellung bereitgestellt werden, müssen sie mit dem Pfad-Trennzeichen ihrer Plattform getrennt werden, z.B. das Semikolon ";" bei Windows und der Doppelpunkt ":" bei Linux).

Standardwert

Es ist kein Standardwert definiert.

Hinweise zum Aufruf von Web Service APIs von interaktiven externen Aufrufen

- Interact empfiehlt zu vermeiden, dass externe Aufrufmodule zusätzliche SOAP-Aufrufe durchführen, um ihre Verarbeitung abzuschließen. Dies kann zwar bei richtiger Konfiguration funktionieren, kann aber auch zu schwer zu behebenden Classloading und internen SOAP-Konfigurationsfehlern führen. Falls Sie andere Systeme aus deren externen Aufrufe heraus aufrufen müssen, wird empfohlen, REST oder alternative Webservice-API-Frameworks zu verwenden.
- Wenn SOAP-APIs innerhalb der externen Aufrufe von Interact verwendet werden sollen, müssen sie mit der Version von SOAP/Axis2 kompiliert werden, die in Interact enthalten ist. Ab dieser Veröffentlichung ist die Version Axis2 Version 1.52. Sie können die genaue Version, die von Interact verwendet wird, ermitteln, indem Sie nach den *axis2*.jar-Dateien im Verzeichnis InteractRT.war/web-inf/lib suchen und die am Ende ihres Dateinamens aufgelistete Version vermerken (Beispiel, axis2-kernel-1.5.2.jar). Die Version von Axis2, die zum Kompilieren Ihrer SOAP-Stubs verwendet wurde, wird in der Regel als Kommentar am Anfang der generierten oder Ihnen zur Verfügung gestellten SOAP *Stub.java-Datei aufgeführt.
- Wenn Ihr externer Aufruf auf SOAP-Fehler stößt, müssen Sie die Klassenladeprotokollierung auf Ihrem Anwendungsserver aktivieren, um sicherzustellen, dass nur die von Interact bereitgestellten Axis2- und Axiom-Bibliotheken geladen und sowohl von Interact als auch von Ihrem externen Aufrufcode verwendet werden.
- Fehler bei der Verarbeitung externer Aufrufe können in der Datei interact.log gefunden werden.

Interact | Ablaufdiagramm | ExternalCallouts | [ExternalCalloutName] | Parameterdaten | [parameterName]

In diesem Abschnitt werden die Parametereinstellungen für einen benutzerdefinierten externen Aufruf definiert, den Sie mit der externen Callout-API geschrieben haben.

Wert

Syntax

Der Wert für einen beliebigen Parameter, der für die Klasse für den externen Aufruf erforderlich ist.

Standardwert

Es ist kein Standardwert definiert.

Beispiel

Wenn der externe Aufruf den Hostnamen eines externen Servers erfordert, erstellen Sie eine Parameterkategorie mit der Bezeichnung `host`, und definieren Sie die `value`-Eigenschaft als Servernamen.

Interact | Überwachung

Diese Gruppe von Konfigurationseigenschaften ermöglicht das Definieren von JMX-Überwachungseinstellungen. Sie müssen diese Eigenschaften nur konfigurieren, wenn Sie die JMX-Überwachung verwenden. In den Konfigurationseigenschaften für die Unica Interact-Designumgebung müssen für das Kontaktverlaufs- und Antwortverlaufsmodul separate JMX-Überwachungseigenschaften definiert werden.

protocol

Syntax

Definieren Sie das Protokoll für den Unica Interact-Nachrichtenservice.

Bei der Auswahl von JMXMP müssen die folgenden JAR-Dateien im Klassenpfad in der richtigen Reihenfolge enthalten sein:

```
Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar
```

Standardwert

JMXMP

Gültige Werte

JMXMP | RMI

Port

Syntax

Die Portnummer für den Nachrichtenservice.

Standardwert

9998

enableSecurity

Syntax

Ein boolescher Operator, der die Sicherheit für den JMXMP-Nachrichtenservice für den Unica Interact-Laufzeitserver aktiviert oder inaktiviert. Wenn der Wert auf `true` festgelegt ist, müssen Sie einen Benutzernamen und ein Kennwort angeben, um auf den Unica Interact-Laufzeit-JMX-Service zugreifen zu können. Diese Benutzerberechtigungsnachweise werden von Unica Platform für den Laufzeitserver authentifiziert. Jconsole erfordert, dass bei der Anmeldung ein Kennwort angegeben werden muss.

Bei einem RMI-Protokoll hat diese Eigenschaft keine Auswirkung. Diese Eigenschaft hat keine Auswirkung auf JMX für Unica Campaign (die Unica Interact-Entwicklungszeit).

Standardwert

True

Gültige Werte

True | False

Interact | Überwachung| activitySubscribers

Diese Gruppe von Konfigurationseigenschaften aktiviert den Stammknoten für die Einstellungen, die in Zusammenhang mit fernen Abonnenten stehen, die regelmäßige Aktualisierungen zu den grundlegenden Leistungsdaten in der Unica Interact-Laufzeitumgebung empfangen können.

heartbeatPeriodInSecs

Syntax

Das Intervall in Sekunden, in dem die einzelnen Laufzeitinstanzen eine Aktualisierung an die Abonnenten senden.

Standardwert

60

Interact | Überwachung | activitySubscribers | (Ziel)

(target)

Syntax

Der Stammknoten für die Einstellungen eines Abonnenten.

URL

Syntax

Die URL dieses Abonnenten. Dieser Endpunkt muss in der Lage sein, JSON-Nachrichten zu akzeptieren, die via HTTP übertragen wurden.

continuousErrorsForAbort

Syntax

Die Anzahl der aufeinanderfolgenden fehlgeschlagenen Aktualisierungen, die erreicht werden muss, bevor die Laufzeitinstanz das Senden weiterer Aktualisierungen an diesen Abonnenten einstellt.

Standardwert

5

timeoutInMillis

Syntax

Das Zeitlimit in Millisekunden, nach dem beim Sendeprozess eine Zeitlimitüberschreitung beim Senden einer Aktualisierung an diesen Abonnenten auftritt.

Standardwert

`1000`

Gültige Werte

Aktiviert

Syntax

Gibt an, ob dieser Abonnent aktiviert oder inaktiviert ist.

Standardwert

`True`

Gültige Werte

`True` oder `False`

Typ

Syntax

Der Typ dieses Datenspeichers. Wenn diese Option ausgewählt wird, dann muss der Parameter **className** hinzugefügt werden. Dabei muss der Wert dem vollständig qualifizierten Namen dieser Implementierungsklasse entsprechen. **classPath** muss mit der URI der JAR-Datei hinzugefügt werden, wenn sie sich nicht im Klassenpfad der Interact-Laufzeit befindet.

Standardwert

`InteractLog`

Gültige Werte

`InteractLog, RelationalDB und Custom`

jmxInclusionCycles

Syntax

Das Intervall als Vielfaches des Werts von **heartbeatPeriodInSecs**, in dem detaillierte JMX-Statistiken an diesen Abonnenten gesendet werden.

Standardwert

`5`

Gültige Werte

Interact | Profil

Diese Gruppe von Konfigurationseigenschaften steuert mehrere optionale Funktionen für Angebotservices, einschließlich der Angebotsunterdrückung und Bewertungsüberschreibung.

enableScoreOverrideLookup

Syntax

Wenn der Wert auf `True` festgelegt wird, lädt Unica Interact die Bewertungsüberschreibungsdaten aus der `scoreOverrideTable`, wenn eine Sitzung erstellt wird. Wenn `False` festgelegt wird, lädt Unica Interact die Marketing-Bewertungsüberschreibungsdaten nicht, wenn eine Sitzung erstellt wird.

Wenn Sie "true" wählen, müssen Sie auch die Eigenschaft `Interact | Profil | Zielgruppen | (Zielgruppe) | scoreOverrideTable` konfigurieren. Sie müssen nur die `scoreOverrideTable`-Eigenschaft für die erforderlichen Zielgruppenebenen definieren. Wenn `scoreOverrideTable` für eine Zielgruppenebene leer gelassen wird, wird die Tabelle für Bewertungsüberschreibung für die Zielgruppenebene inaktiviert.

Standardwert

`False`

Gültige Werte

True | False

enableOfferSuppressionLookup

Syntax

Wenn der Wert auf `True` festgelegt wird, lädt Unica Interact die Angebotsunterdrückungsdaten aus der `offerSuppressionTable`, wenn eine Sitzung erstellt wird. Wenn `False` festgelegt wird, lädt Unica Interact die Marketing Angebotsunterdrückungsdaten nicht, wenn eine Sitzung erstellt wird.

Wenn Sie "true" wählen, müssen Sie auch die Eigenschaft `Interact | Profil | Zielgruppen | (Zielgruppe) | offerSuppressionTable` konfigurieren. Sie müssen nur die `enableOfferSuppressionLookup`-Eigenschaft für die erforderlichen Zielgruppenebenen definieren.

Standardwert

`False`

Gültige Werte

True | False

enableProfileLookup

Syntax

In einer Neuinstallation von Unica Interact wird diese Eigenschaft nicht weiter unterstützt. In einer Unica Interact-Upgradeinstallation ist diese Eigenschaft bis zur ersten Implementierung gültig.

Das Ladeverhalten für eine Tabelle, die in einem interaktiven Ablaufdiagramm verwendet wird, aber nicht im interaktiven Kanal zugeordnet ist.
Wenn der Wert auf `True` festgelegt wird, lädt Unica Interact die Angebotsunterdrückungsdaten aus der `profileTable`, wenn eine Sitzung erstellt wird.

Wenn Sie "true" wählen, müssen Sie auch die Eigenschaft `Interact | Profil | Zielgruppen | (Zielgruppe) | profileTable` konfigurieren.

Die Einstellung **Diese Daten in den Speicher laden, wenn eine Besuchssession startet** im Assistenten für die Zuweisung der interaktiven Kanaltabelle überschreibt diese Konfigurationseinstellung.

Standardwert

`False`

Gültige Werte

True | False

defaultOfferUpdatePollPeriod

Syntax

Die Anzahl der Sekunden, die das System wartet, bevor es die Standardangebote im Cache mit den Werten aus der Standardangebotstabelle aktualisiert. Wenn der Wert auf -1 gesetzt ist, aktualisiert das System die Standardangebote im Cache nicht, nachdem die ursprüngliche Liste in den Cache geladen wurde, wenn der Laufzeitserver startet.

Standardwert

-1

Interact | Profil | Zielgruppen | [AudienceLevelName]

Diese Gruppe von Konfigurationseigenschaften ermöglicht das Definieren der Tabellennamen, die für zusätzliche Unica Interact-Funktionen erforderlich sind. Sie müssen einen Tabellennamen nur definieren, wenn Sie die entsprechende Funktion verwenden.

Neuer Kategoriename

Syntax

Der Name Ihrer Zielgruppenebene.

scoreOverrideTable

Syntax

Der Name der Tabelle, die die Informationen zur Bewertungsüberschreibung für diese Zielgruppenebene enthält. Diese Eigenschaft ist anwendbar, wenn Sie `enableScoreOverrideLookup` auf true gesetzt haben. Sie müssen diese Eigenschaft für die Zielgruppenebenen definieren, für die Sie eine Tabelle für die Bewertungsüberschreibung aktivieren möchten. Wenn für diese Zielgruppenebene keine Tabelle für die Bewertungsüberschreibung vorhanden ist, muss diese Eigenschaft nicht definiert werden, selbst wenn `enableScoreOverrideLookup` auf true gesetzt ist.

Unica Interact sucht diese Tabelle in den Kundentabellen, auf die die Unica Interact-Laufzeitserver zugreifen und die durch die `prodUserDataSource`-Eigenschaften definiert sind.

Wenn Sie die Eigenschaft `schema` für diese Datenquelle definiert haben, fügt Unica Interact vor diesem Tabellennamen das Schema ein, z. B. `schema.UACI_ScoreOverride`. Wenn Sie einen vollständig qualifizierten Namen eingeben, z. B. `mySchema.UACI_ScoreOverride`, fügt Unica Interact den Schemanamen nicht ein.

Standardwert

UACI_ScoreOverride

offerSuppressionTable

Syntax

Der Name der Tabelle, die die Informationen zur Angebotsunterdrückung für diese Zielgruppenebene enthält. Sie müssen diese Eigenschaft für die Zielgruppenebenen definieren, für die Sie eine Tabelle für Angebotsunterdrückung aktivieren möchten. Wenn für diese Zielgruppenebene keine Angebotsunterdrückungstabelle vorhanden ist, muss diese Eigenschaft nicht definiert werden. Steht `enableOfferSuppressionLookup` auf true, muss diese Eigenschaft auf eine gültige Tabelle gesetzt werden.

Unica Interact sucht diese Tabelle in den Kundentabellen, auf die die -Laufzeitserver zugreifen und die durch die `prodUserDataSource`-Eigenschaften definiert sind.

Standardwert

UACI_BlackList

contactHistoryTable

Syntax

Der Name der Staging-Tabelle für die Kontaktverlaufsdaten für diese Zielgruppenebene.

Diese Tabelle wird in den Tabellen der Laufzeitumgebung gespeichert (`systemTablesDataSource`).

Wenn Sie die Eigenschaft `schema` für diese Datenquelle definiert haben, fügt Unica Interact vor diesem Tabellennamen das Schema ein, z. B. `schema.UACI_CHStaging`. Wenn Sie einen vollständig qualifizierten Namen eingeben, z. B. `mySchema.UACI_CHStaging`, fügt Unica Interact den Schemanamen nicht ein.

Ist die Kontaktverlaufsprotokollierung inaktiviert, muss diese Eigenschaft nicht festgelegt werden.

Standardwert

UACI_CHStaging

chOfferAttribTable

Syntax

Der Name der Tabelle für die Angebotsattribute des Kontaktverlaufs für diese Zielgruppenebene.

Diese Tabelle wird in den Tabellen der Laufzeitumgebung gespeichert (`systemTablesDataSource`).

Wenn Sie die Eigenschaft `schema` für diese Datenquelle definiert haben, fügt Unica Interact vor diesem Tabellennamen das Schema ein, z. B. `schema.UACI_CHOfferAttrib`. Wenn Sie einen vollständig qualifizierten Namen eingeben, z. B. `mySchema.UACI_CHOfferAttrib`, fügt Unica Interact den Schemanamen nicht ein.

Ist die Kontaktverlaufsprotokollierung inaktiviert, muss diese Eigenschaft nicht festgelegt werden.

Standardwert

UACI_CHOfferAttrib

responseHistoryTable

Syntax

Der Name der Staging-Tabelle für den Antwortverlauf für diese Zielgruppenebene.

Diese Tabelle wird in den Tabellen der Laufzeitumgebung gespeichert (`systemTablesDataSource`).

Wenn Sie die Eigenschaft `schema` für diese Datenquelle definiert haben, fügt Unica Interact vor diesem Tabellennamen das Schema ein, z. B. `schema.UACI_RHStaging`. Wenn Sie einen vollständig qualifizierten Namen eingeben, z. B. `mySchema.UACI_RHStaging`, fügt Unica Interact den Schemanamen nicht ein.

Ist die Antwortverlaufsprotokollierung inaktiviert, muss diese Eigenschaft nicht festgelegt werden.

Standardwert

UACI_RHStaging

crossSessionResponseTable

Syntax

Der Name der Tabelle für diese Zielgruppenebene, die für die Antwortverfolgung in den Kontakt- und Antwortverlaufstabellen erforderlich ist, auf die die Funktion für die Antwortverfolgung zugreifen kann.

Wenn Sie die Eigenschaft `schema` für diese Datenquelle definiert haben, fügt Unica Interact vor diesem Tabellennamen das Schema ein, z. B. `schema.UACI_XSessResponse`. Wenn Sie einen vollständig qualifizierten Namen eingeben, z. B. `mySchema.UACI_XSessResponse`, fügt Unica Interact den Schemanamen nicht ein.

Ist die sitzungsübergreifende Antwortprotokollierung inaktiviert, muss diese Eigenschaft nicht festgelegt werden.

Standardwert

`UACI_XSessResponse`

userEventLoggingTable

Syntax

Gibt den Namen der Datenbanktabelle an, die für die Protokollierung benutzerdefinierter Ereignisaktivitäten verwendet wird. Benutzer können Ereignisse auf der Registerkarte "Ereignisse" der Übersichtsseite "Interaktiver Kanal" in der Unica Interact-Schnittstelle definieren. Die Datenbanktabelle, die Sie hier angeben, dient zur Speicherung von Informationen wie z.B. der Ereignis-ID, des Namens und der Anzahl der Vorkommen dieses Ereignisses für die aktuelle Zielgruppenebene seit der letzten Löschung des Ereignisaktivitätscaches.

Wenn Sie die Eigenschaft `schema` für diese Datenquelle definiert haben, fügt Unica Interact vor diesem Tabellennamen das Schema ein, z. B. `schema.UACI_UserEventActivity`. Wenn Sie einen vollständig qualifizierten Namen eingeben, z. B. `mySchema.UACI_UserEventActivity`, fügt Unica Interact den Schemanamen nicht ein.

Standardwert

`UACI_UserEventActivity`

patternStateTable

Syntax

Gibt den Namen der Datenbanktabelle an, die für die Protokollierung von Ereignismusterstatus verwendet wird (beispielsweise ob die Musterbedingung erfüllt wurde, ob das Muster abgelaufen oder inaktiviert ist usw.).

Wenn Sie die Eigenschaft `schema` für diese Datenquelle definiert haben, fügt Unica Interact vor diesem Tabellennamen das Schema ein, z. B. `schema.UACI_EventPatternState`. Wenn Sie einen vollständig qualifizierten Namen eingeben, z. B. `mySchema.UACI_EventPatternState`, fügt Unica Interact den Schemanamen nicht ein.

Für jede Zielgruppenebene ist eine Eigenschaft patternStateTable erforderlich, auch wenn keine Ereignismuster verwendet werden. patternStateTable basiert auf der DLL des integrierten `UACI_EventPatternState`. Nachfolgend wird ein Beispiel gezeigt, bei dem die Zielgruppen-ID die beiden Komponenten ComponentNum und ComponentStr hat.

```
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)
)
```

Standardwert

`UACI_EventPatternState`

Interact | Profil | Zielgruppen | [AudienceLevelName] | Angebote von Raw SQL

Diese Gruppe von Konfigurationseigenschaften ermöglicht das Definieren der Tabellennamen, die für zusätzliche Unica Interact-Funktionen Sie müssen einen Tabellennamen nur definieren, wenn Sie die entsprechende Funktion verwenden.

enableOffersByRawSQL

Syntax

Wenn diese Eigenschaft auf `True` festgelegt ist, aktiviert Unica Interact die Funktion offersBySQL für diese Zielgruppenebene. Dies ermöglicht Ihnen, SQL-Code zu konfigurieren, um einen gewünschten Satz von potenziellen Angeboten zur Laufzeit zu erstellen. Bei Festlegung auf `False` verwendet Unica Interact die Funktion offersBySQL nicht.

Wenn Sie diese Eigenschaft auf "True" festlegen, können Sie auch die Eigenschaft `Interact | Profil | Zielgruppen | (Zielgruppe) | Angebot von Raw SQL | SQL Template` konfigurieren, um eine oder mehrere SQL-Vorlagen zu definieren.

Standardwert

`False`

Gültige Werte

True | False

cacheSize

Syntax

Größe des Cache zum Speichern von Ergebnissen der OfferBySQL-Abfragen. Beachten Sie, dass die Verwendung eines Cache negative Auswirkungen auf die Leistung haben kann, wenn die Abfrageergebnisse für die meisten Sitzungen eindeutig sind.

Standardwert

`-1 (off)`

Gültige Werte

-1 | Wert

cacheLifeInMinutes

Syntax

Wenn der Cache aktiviert ist, gibt dies die Anzahl von Minuten an, die gewartet wird, bevor das System den Cache löscht, um die Aktualität zu gewährleisten.

Standardwert

`-1 (off)`

Gültige Werte

-1 | Wert

defaultSQLTemplate

Syntax

Der Name der zu verwendenden SQL-Vorlage, wenn keine über die API-Aufrufe angegeben wird.

Standardwert

`Keine`

Gültige Werte

Name der SQL-Vorlage

Name

Konfigurationskategorie

`Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL | (SQL Templates)`

Syntax

Der Name, den Sie dieser SQL-Abfragenvorlage zuweisen möchten. Geben Sie einen beschreibenden Namen ein, der aussagekräftig ist, wenn Sie diese SQL-Vorlage in API-Aufrufen verwenden. Beachten Sie: Wenn Sie hier einen Namen verwenden, der mit einem im Interact-Listenprozessfeld für ein offerBySQL-Verfahren definierten Namen identisch ist, wird statt der hier eingegebenen SQL die SQL im Prozessfeld verwendet.

Standardwert

Keine

SQL

Konfigurationskategorie

`Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL | (SQL Templates)`

Syntax

Enthält die SQL-Abfrage, die durch diese Vorlage aufgerufen wird. Die SQL-Abfrage kann Verweise auf Variablennamen enthalten, die Teil der Sitzungsdaten (Profil) des Besuchers sind. Beispielsweise wäre `select * from MyOffers where category = ${preferredCategory}` auf die Sitzung angewiesen, die eine Variable namens `preferredCategory` enthält.

Sie müssen die SQL so konfigurieren, dass die speziellen Angebotstabellen abgefragt werden, die Sie während der Entwicklungszeit zur Verwendung durch diese Funktion erstellt haben. Beachten Sie, dass gespeicherte Prozeduren hier nicht unterstützt werden.

Standardwert

Keine

Interact | Profil | Zielgruppen | [AudienceLevelName] | SQL-Vorlage

Mit diesen Konfigurationseigenschaften können Sie eine oder mehrere SQL-Abfragenvorlagen für die Verwendung durch die Funktion offersBySQL von Unica Interact definieren.

Name

Syntax

Der Name, den Sie dieser SQL-Abfragenvorlage zuweisen möchten. Geben Sie einen beschreibenden Namen ein, der aussagekräftig ist, wenn Sie diese SQL-Vorlage in API-Aufrufen verwenden. Beachten Sie: Wenn Sie hier einen Namen verwenden, der mit einem im Interact-Listenprozessfeld für ein offerBySQL-Verfahren definierten Namen identisch ist, wird statt der hier eingegebenen SQL die SQL im Prozessfeld verwendet.

Standardwert

Keine

SQL

Syntax

Enthält die SQL-Abfrage, die durch diese Vorlage aufgerufen wird. Die SQL-Abfrage kann Verweise auf Variablennamen enthalten, die Teil der Sitzungsdaten (Profil) des Besuchers sind. Beispielsweise wäre `select * from MyOffers where category = ${preferredCategory}` auf die Sitzung angewiesen, die eine Variable namens `preferredCategory` enthält.

Sie müssen die SQL so konfigurieren, dass die speziellen Angebotstabellen abgefragt werden, die Sie während der Entwicklungszeit zur Verwendung durch diese Funktion erstellt haben. Beachten Sie, dass gespeicherte Prozeduren hier nicht unterstützt werden.

Standardwert

Keine

Interact | profile | Zielgruppe | [AudienceLevelName | Profildatendienste | [DataSource]

Diese Gruppe von Konfigurationseigenschaften ermöglicht das Definieren der Tabellennamen, die für zusätzliche Unica Interact-Funktionen Sie müssen einen Tabellennamen nur definieren, wenn Sie die entsprechende Funktion verwenden. In der

Kategorie "Profile Data Services" werden Informationen über eine integrierte Datenquelle (Datenbank) angegeben, die für alle Zielgruppenebenen erstellt wird und mit einer Priorität von 100 vorkonfiguriert ist. Sie können die Datenquelle jedoch auch ändern oder inaktivieren. In dieser Kategorie ist außerdem eine Vorlage für zusätzliche externe Datenquellen enthalten. Wenn Sie auf die Vorlage mit dem Namen **Externe Datenservices** klicken, können Sie die hier beschriebenen Konfigurationseinstellungen vornehmen.

Neuer Kategoriename

Syntax

(Nicht verfügbar für den Standard-Datenbankeintrag.) Der Name der Datenquelle, die Sie definieren. Der hier eingegebene Name muss in allen Datenquellen einer Zielgruppenebene eindeutig sein.

Standardwert

Keine

Gültige Werte

Jede Textzeichenfolge ist zulässig.

Aktiviert

Syntax

Wenn der Wert auf `True` festgelegt ist, wird diese Datenquelle für die Zielgruppenebene aktiviert, der sie zugeordnet ist. Wenn er auf `False` festgelegt ist, verwendet Unica Interact diese Datenquelle nicht für diese Zielgruppenebene.

Standardwert

`True`

Gültige Werte

True | False

className

Syntax

(Nicht verfügbar für den Standard-Datenbankeintrag.) Der vollqualifizierte Name der Datenquellenklasse, die `IInteractProfileDataService` implementiert.

Standardwert

Keine.

Gültige Werte

Eine Zeichenfolge, die einen vollständig qualifizierten Klassennamen angibt.

classPath

Syntax

(Nicht verfügbar für den Standard-Datenbankeintrag.) Eine optionale Konfigurationseinstellung, die den Pfad zum Laden dieser Datenquellen-Implementierungsklasse angibt. Wenn Sie die Einstellung weglassen, wird standardmäßig der Klassenpfad des übergeordneten Anwendungsservers verwendet.

Standardwert

Wird nicht angezeigt, aber wenn hier kein Wert angegeben ist, wird standardmäßig der Klassenpfad des übergeordneten Anwendungsservers verwendet.

Gültige Werte

Eine Zeichenfolge, die den Klassenpfad angibt.

Priorität

Syntax

Die Priorität dieser Datenquelle in dieser Zielgruppenebene. Der Wert muss in allen Datenquellen jeder Zielgruppenebene eindeutig sein. (Das heißt, wenn für eine Datenquelle die Priorität 100 festgelegt ist, kann keine weitere Datenquelle in der Zielgruppenebene die Priorität 100 haben).

Standardwert

`100` für die Standarddatenbank, `200` für eine benutzerdefinierte Datenquelle

Gültige Werte

Jede nicht negative Ganzzahl ist zulässig.

Affinium|interact|profile|Audience Levels|[Audience Levels]requestLogTable

Mit dieser Konfigurationseigenschaft können Sie den Namen der Tabelle zur Erfassung von API Anfragen bestimmen, die auf diese bestimmte Zielgruppenebene ausgerichtet sind. Wird das Feld leer gelassen, findet auf dieser Zielgruppenebene keine API Protokollierung statt. Diese Funktion muss global aktiviert werden, um API Anfragen für diese Zielgruppenebene zu protokollieren.

Standardwert

leere Zeichenfolge

Interact | offerserving

Mit diesen Konfigurationseigenschaften werden die generischen Lernkonfigurationseigenschaften definiert. Verwenden Sie bei einem integrierten Lernmodul die Konfigurationseigenschaften für die Designumgebung, um Ihre Implementierung des Lernmoduls zu optimieren.

offerTieBreakMethod

Beschreibung

Die Eigenschaft `offerTieBreakMethod` definiert das Verhalten der Angebotsbereitstellung, wenn zwei Angebote über gleichwertige (Gleichstand) Bewertungen verfügen. Wenn Sie diese Eigenschaft auf ihren Standardwert "Wahlfrei" (Random) setzen, gibt Unica Interact eine zufällige Auswahl aus den Angeboten wieder, die über eine gleichwertige Bewertung verfügen. Wenn Sie dieser Konfiguration auf ein neueres Angebot (Newer Offer) setzen, stellt Unica Interact das neuere Angebot (basierend auf der höheren Angebots-ID)

vor dem älteren Angebot (mit der niedrigeren Angebots-ID) bereit, wenn die Bewertungen der Angebote gleich sind.

Note:

Unica Interact verfügt über eine Zusatzfunktion, die dem Administrator die Systemkonfiguration für die beliebige Angebotsrückgabe unabhängig von der Bewertung ermöglicht, indem die Option `percentRandomSelection` (`Campaign | partitions | [partition_number] | Interact | learning | percentRandomSelection`) festgelegt wird. Die hier beschriebene Eigenschaft `offerTieBreakMethod` wird verwendet nur wenn `percentRandomSelection` auf Null gesetzt wird (deaktiviert).

Standardwert

Zufall

Gültige Werte

Zufall | Neueres Angebot

optimizationType

Beschreibung

Die Eigenschaft `optimizationType` definiert, ob Unica Interact eine Lernplattform zur Unterstützung bei Angebotszuweisungen verwendet. Wird der Wert auf `NoLearning` eingestellt, verwendet Unica Interact kein Lernmodul. Wird der Wert auf `BuiltInLearning` eingestellt, verwendet Unica Interact das in Unica Interact integrierte Bayessches Lernmodul. Wird der Wert auf `ExternalLearning` eingestellt, verwendet Unica Interact eine von Ihnen bereitgestellte Lernplattform. Wird die Option `ExternalLearning` ausgewählt, müssen Sie die Eigenschaften `externalLearningClass` und `externalLearningClassPath` definieren.

Standardwert

NoLearning

Gültige Werte

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

Beschreibung

Die maximale Dauer in Millisekunden, die der Laufzeitserver wartet, bis ein interaktives Ablaufdiagramm abgeschlossen ist, bevor Angebote angenommen werden.

Standardwert

`5000`

treatmentCodePrefix

Beschreibung

Das Präfix, das in Verfahrenscodes eingefügt wird.

Standardwert

Es ist kein Standardwert definiert.

effectiveDateBehavior

Beschreibung

Bestimmt, ob Unica Interact beim Herausfiltern der einem Besucher präsentierten Angebote das Aktivierungsdatum eines Angebots berücksichtigen soll. Mögliche Werte:

- -1 weist Unica Interact an, das Aktivierungsdatum des Angebots zu ignorieren.

  0 weist Unica Interact an, beim Filtern des Angebots das Aktivierungsdatum zu berücksichtigen. Wenn nun das Aktivierungsdatum des Angebots vor dem aktuellen Datum liegt oder mit diesem identisch ist, wird das Angebot dem Besucher präsentiert.

Wird ein Wert **für effectiveDateGracePeriod** schon eingestellt, wird die Nachfrist auch bei der Entscheidung über die Angebotsbereitstellung berücksichtigt.

- Eine positive Ganzzahl weist Unica Interact an, das aktuelle Datum plus den Wert dieser Eigenschaft zu verwenden, um zu bestimmen, ob das Angebot Besuchern präsentiert werden soll. Wenn nun das Aktivierungsdatum des Angebots vor dem aktuellen Datum plus dem Wert dieser Eigenschaft liegt, wird das Angebot den Besuchern präsentiert.

  Wird ein Wert **für effectiveDateGracePeriod** schon eingestellt, wird die Nachfrist auch bei der Entscheidung über die Angebotsbereitstellung berücksichtigt.

Standardwert

-1

effectiveDateGracePeriodOfferAttr

Beschreibung

Gibt den Namen des angepassten Attributs in einer Angebotsdefinition an, das die Karenzzeit für das Aktivierungsdatum definiert. Z.B. können Sie diese Eigenschaft mit dem Wert `AltGracePeriod` konfigurieren. Danach definieren Sie die Angebote mit einem benutzerdefinierten Attribut `AltGracePeriod`, das die Anzahl der Tage angibt, die als Nachfrist zusammen mit der Eigenschaft **effectiveDateBehavior** verwendet werden sollen.

Nehmen Sie an, dass Sie eine neue Angebotsvorlage mit einem Gültigkeitsdatum von 10 Tagen nach dem aktuellen Datum erstellen und fügen Sie ein benutzerdefiniertes Attribut `AltGracePeriod` hinzu. Wenn Sie ein Angebot mit dieser Vorlage erstellen und den Wert von `AltGracePeriod` auf 14 Tage setzen, wird das Angebot den Besuchern bereitgestellt, da das aktuelle Datum innerhalb der Nachfrist des Angebotsdatums liegt.

Standardwert

Leer

alwaysLogLearningAttributes

Beschreibung

Gibt an, ob Unica Interact Informationen zu Besucherattributen, die vom Lernmodul verwendet werden, in die Protokolldateien schreiben soll. Beachten Sie, dass das Festlegen dieses Werts auf "True" sich auf die Lernleistung und die Protokolldateigrößen auswirken kann.

Standardwert

False

Beachten Sie dabei die folgenden Punkte:

- Wenn das Lernen für Version 2 mit BeispielMethode1 und für Version 1 mit BeispielMethode1 aktiviert ist
- Während des Interaktionskonfigurationsprozesses überprüft das System, ob die Einstellungen übereinstimmen.
- Während des Optimierungsprozesses für die interaktive Angebotsbehandlung in Lernversion 2 werden die auf den Einstellungen BeispielMethode1 und BeispielMethode2 basierenden Angebotslernmethoden behandelt. Die Angebots-RWA wird basierend auf den Einstellungen BeispielMethode1 und BeispielMethode2 ebenfalls berechnet.

includeArbitrationInfo

Diese Konfigurationseigenschaft bestimmt, ob die Zusammenfassung der Angebotsentscheidung in die Antwort von den Anfragen getOffers und getOffersForMultipleInteractionPoints aufgenommen werden soll.

Standardwert

False

Gültige Werte

True | False

Interact | Offerserving | integrierte Lernmodul Konfiguration

Diese Konfigurationseigenschaften definieren die Schreibeinstellungen der Datenbank für das integrierte Lernmodul. Verwenden Sie die Konfigurationseigenschaften für die Designumgebung, um Ihre Implementierung des Lernmoduls zu optimieren.

version

Syntax

Sie können `1` oder `2` auswählen. Version 1 ist die Basiskonfigurationsversion, die keine Parameter für Grenzwerte von Threads und Datensätzen verwendet. Version 2 ist die erweiterte Konfigurationsversion, die das Festlegen von Parametern für Grenzwerte von Threads und Datensätzen ermöglicht, um die Leistung zu verbessern. Diese Parameter führen Aggregationen und Löschungen aus, wenn diese Parametergrenzen erreicht werden.

Standardwert

1

insertRawStatsIntervalInMinutes

Syntax

Die Anzahl von Minuten, die Unica Interact wartet, bevor weitere Zeilen in die Staging-Lerntabellen eingefügt werden. Abhängig von der Datenmenge, die das Lernmodul in Ihrer Umgebung verarbeitet, muss diese Dauer u.U. geändert werden.

Standardwert

5

Gültige Werte

Positive ganze Zahl

aggregateStatsIntervalInMinutes

Syntax

Die Anzahl an Minuten, die Unica Interact zwischen dem Aggregieren von Daten in den Staging-Lerntabellen wartet. Abhängig von der Datenmenge, die das Lernmodul in Ihrer Umgebung verarbeitet, muss diese Dauer u.U. geändert werden.

Standardwert

15

Gültige Werte

Eine Ganzzahl größer 0.

autoAdjustPercentage

Syntax

Der Wert, mit dem der Prozentsatz von Daten festgelegt wird, der bei der Ausführung von Versuchen zum Aggregieren auf der Basis der Metriken der vorherigen Ausführung verarbeitet werden soll. Standardmäßig wird dieser Wert auf 0 gesetzt, was bedeutet, dass bei der Aggregation alle Staging-Sätze verarbeitet werden sollen und die Funktion zur automatischen Anpassung inaktiviert wird.

Standardwert

0

Gültige Werte

Eine Zahl von 0 bis 100.

excludeAbnormalAttribute

Syntax

Mit dieser Einstellung wird festgelegt, ob diese Attribute als ungültig markiert werden sollen. Bei dem Wert `IncludeAttribute` werden abnormale Attribute aufgenommen und nicht als ungültig markiert. Bei dem Wert

`ExcludeAttribute` werden abnormale Attribute ausgeschlossen und als ungültig markiert.

Standardwert

IncludeAttribute

Gültige Werte

IncludeAttribute | **ExcludeAttribute**

saveOriginalValues

Syntax

Sie können die Werte als " Alle Werte", "Binned Values" oder "Keine" einstellen. Dadurch wird gesteuert, welche Werte in der Tabelle UACI_LearningAttributeHist protokolliert werden.

Wenn "All Values" ausgewählt wird, werden alle Lernattribute in der Tabelle protokolliert. Wenn dieser Parameter auf "Attributwerte" gesetzt ist, werden in der Tabelle, für die unter "Interact-> Global Learning" Bins angelegt werden, nur die Attribute protokolliert.

Bei der Einstellung "Keine" werden keine Werte in UACI_LearningAttributeHist protokolliert.

Standardmäßig ist dies auf "None" eingestellt.

Standardwert

Keine

Gültige Werte

Alle Werte | **Attributwerte** | **Keine**

Interact | offerserving | integrierte Lernmodul Konfiguration | Datenparameter | [parameterName]

Diese Konfigurationseigenschaften definieren alle Parameter für das externe Lernmodul.

numberOfThreads

Syntax

Die maximale Anzahl der Threads, die der Lernaggregator für die Verarbeitung der Daten verwendet. Ein gültiger Wert ist eine positive Ganzzahl. Der Wert sollte die maximale Anzahl der Verbindungen, die in der Lerndatenquelle konfiguriert sind, nicht überschreiten. Dieser Parameter wird nur von der Version 2 des Aggregators verwendet.

Standardwert

10

maxLogTimeSpanInMin

Syntax

Wenn die Version 1 des Aggregators ausgewählt ist, können Sie die Staging-Datensätze in Iterationen verarbeiten, um extrem umfangreiche Datenbankstapel zu vermeiden. In diesem Fall werden die betreffenden Staging-Datensätze nach Blöcken verarbeitet, und zwar Iteration für Iteration in einem einzelnen Aggregationszyklus. Der Wert dieses Parameters gibt die maximale Zeitspanne für Staging-Datensätze an, deren Verarbeitung der Aggregator in jeder Iteration versucht. Diese Zeitspanne basiert auf dem Feld "LogTime", das jedem Staging-Datensatz zugeordnet ist. Nur die Datensätze, deren LogTime in das früheste Zeitfenster fallen, werden verarbeitet. Ein gültiger Wert ist eine Ganzzahl, die nicht negativ ist. Bei dem Wert 0 ist kein Grenzwert festgelegt, sodass alle Staging-Datensätze in einer einzelnen Iteration verarbeitet werden.

Standardwert

0

maxRecords

Syntax

Wenn die Version 2 des Aggregators ausgewählt ist, können Sie die Staging-Datensätze in Iterationen verarbeiten, um extrem umfangreiche Datenbankstapel zu vermeiden. In diesem Fall werden die betreffenden Staging-Datensätze in Blöcken verarbeitet, und zwar Iteration für Iteration in einem einzelnen Aggregationszyklus. Der Wert dieses Parameters gibt die maximale Anzahl der Staging-Datensätze an, deren Verarbeitung der Aggregator in jeder Iteration versucht. Ein gültiger Wert ist eine Ganzzahl, die nicht negativ ist. Bei dem Wert 0 ist kein Grenzwert festgelegt, sodass alle Staging-Datensätze in einer einzelnen Iteration verarbeitet werden.

Standardwert

0

Wert

Syntax

Der Wert für jeden Parameter, der von der Klasse für ein integriertes Lernmodul benötigt wird.

Standardwert

Es ist kein Standardwert definiert.

Interact | Offerserving | External Learning Config

Diese Konfigurationseigenschaften definieren die Klasseneinstellungen für ein externes Lernmodul, das Sie mit der Lern-API geschrieben haben.

Klasse

Syntax

Wenn `optimizationType` auf `ExternalLearning` gesetzt ist, legen Sie `externalLearningClass` auf den Klassennamen für die externe Lernengine fest.

Standardwert

Es ist kein Standardwert definiert.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn `optimizationType` auf `ExternalLearning` festgelegt ist.

classPath

Syntax

Wenn `optimizationType` auf `ExternalLearning` gesetzt ist, legen Sie `externalLearningClass` auf den Klassenpfad für die externe Lernengine fest.

Der Klassenpfad muss auf jar-Dateien auf dem Laufzeitserver verweisen. Wenn Sie eine Servergruppe verwenden und alle Laufzeitserver dieselbe Unica Platform verwenden, muss jeder Server über eine Kopie der jar-Datei an demselben Datenträger verfügen. Der Klassenpfad muss absolute Datenträgere der jar-Dateien enthalten, die durch das Pfadtrennzeichen des Betriebssystems des Servers für die Laufzeitumgebung getrennt sind, z. B. Semikolon (;) bei Windows™-Systemen und Doppelpunkt (:) bei UNIX™-Systemen. Verzeichnisse, die Klassendateien enthalten, sind nicht zulässig. Beispielsweise unter einem Unix-Betriebssystem: `/path1/file1.jar:/path2/file2.jar`.

Dieser Klassenpfad kann maximal 1024Zeichen enthalten. Mit der Manifestdatei in einer jar-Datei können Sie andere jar-Dateien angeben, sodass im Klassenpfad nur eine jar-Datei enthalten sein muss.

Standardwert

Es ist kein Standardwert definiert.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn `optimizationType` auf `ExternalLearning` festgelegt ist.

Interact | offerserving | External Learning Config | Parameter Data | [parameterName]

Diese Konfigurationseigenschaften definieren alle Parameter für das externe Lernmodul.

Wert

Syntax

Der Wert für jeden Parameter, der für die Klasse eines externen Lernmoduls erforderlich ist.

Standardwert

Es ist kein Standardwert definiert.

Beispiel

Wenn das externe Lernmodul einen Pfad zu einer Algorithmuslösung erfordert, erstellen Sie eine Parameterkategorie mit der Bezeichnung `solverPath`, und definieren Sie die Eigenschaft `value` als Pfad zu der Anwendung.

Interact | offerserving | Einschränkungen

Diese Konfigurationseigenschaften definieren die Einschränkungen, die für den Angebotsbereitstellungsprozess gelten.

maxOfferAllocationInMemoryPerInstance

Syntax

Die Größe eines Angebotsblocks. Unica Interact verwaltet einen Pool von Angeboten im Speicher. Auf diese Weise muss das System nicht jedes Mal, wenn ein Angebot zurückgegeben wird, die Datenbank abfragen. Wird ein Angebot zurückgegeben, dann passt das System den Pool an. Wenn der Pool vollständig ausgelastet ist, dann erhält Unica Interact einen weiteren Angebotsblock, um den Pool wieder zu füllen.

Standardwert

1000

Gültige Werte

Eine Ganzzahl größer 0.

maxDistributionPerIntervalPerInstanceFactor

Syntax

Der Prozentsatz der Einschränkungen einer angegebenen Angebotszuordnung für einen Laufzeitserver zur Unterstützung der Verteilung auf die Laufzeitserver.

Standardwert

100

Gültige Werte

Eine Ganzzahl zwischen 0 und 100.

constraintCleanupIntervalInDays

Syntax

Gibt an, wie oft die inaktivierten Zähler der Tabelle "UACI_OfferCount" bereinigt werden. Durch einen Wert kleiner als "1" wird diese Funktion inaktiviert.

Standardwert

7

Gültige Werte

Eine Ganzzahl größer 0.

Interact | Dienste

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für alle Services, die das Sammeln von Kontakt- und Antwortverlaufsdaten sowie Statistiken für die Berichterstellung und Schreibvorgänge in die Systemtabellen der Laufzeitumgebung verwalten.

externalLoaderStagingDirectory

Syntax

Diese Eigenschaft definiert die Position des Staging-Verzeichnisses für ein Datenbankladeprogramm.

Standardwert

Es ist kein Standardwert definiert.

Gültige Werte

Ein Pfad, der sich auf das Unica Interact-Installationsverzeichnis bezieht, oder ein absoluter Pfad zu einem Staging-Verzeichnis.

Wenn Sie ein Datenbankladeprogramm aktivieren, müssen Sie die Eigenschaft `cacheType` in den Kategorien `contactHist` und `responstHist` auf `External Loader File` setzen.

Affinium|interact|Dienste|contactHist|treatmentStoreReference

Dieser Konfigurationsparameter ist der Wurzelknoten für die Einstellungen, die sich auf den Datenspeicher der kürzlich bereitgestellten Verfahren beziehen.

daysBackForXSessContact

Die Anzahl der Tage, an denen ein bereitgestelltes Verfahren im Datenspeicher für die sitzungsübergreifende Suche aufbewahrt wird. Wenn der Wert nicht positiv oder null ist, ist die Funktion zur Verfolgung von Sitzungskontakten inaktiviert

Standardwert

0

Gültiger Wert

Beliebige positive Zahl

Interact | Dienste | contactHist

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Service, der Daten für die Staging-Tabellen für den Kontaktverlauf sammelt.

enableLog

Syntax

Wenn der Wert auf `true` festgelegt ist, ist der Service aktiviert, der Daten für die Aufzeichnung der Kontaktverlaufsdaten sammelt. Bei `false` werden keine Daten gesammelt.

Standardwert

True

Gültige Werte

True | False

cacheType

Syntax

Definiert, ob die für den Kontaktverlauf gesammelten Daten im Speicher (`Memory Cache`) oder in einer Datei (`External Loader file`) gespeichert werden. Sie können `External Loader File` nur verwenden, wenn Sie Unica Interact für die Verwendung eines Datenbankladeprogramms konfiguriert haben.

Wenn Sie `Memory Cache` auswählen, verwenden Sie die Kategorieeinstellungen `cache`. Wenn Sie `External Loader File` auswählen, verwenden Sie die Kategorieeinstellungen `fileCache`.

Standardwert

Memory Cache

Gültige Werte

Memory Cache | External Loader File

Interact | Dienste | contactHist | cache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der Daten für die Staging-Tabelle für den Kontaktverlauf sammelt. **Anmerkung**: Wenn contactHist oder responseHist für die Verwendung von memoryCache

konfiguriert ist, können Sie optional eine Datenquelle systemTablesDataSource erstellen und die Einstellungen unter Affinium|interact|allgemein|systemTablesDataSource|loaderProperties konfigurieren. Wenn dies durchgeführt wird, werden die Kontakt-/Antwort-Historie-Staging-Einträge in Dateien im Verzeichnis, wie durch Affinium|interact|Dienste|externalLoaderStagingDirectory festgelegt, persistiert, wenn die Persistenz in der Datenbank fehlschlägt. Andernfalls wird bei der Initialisierung ein INFO-Eintrag protokolliert, der angibt, dass das Failover nicht aktiviert ist.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten Kontaktverlaufsdaten in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | contactHist | contactStatusCodes

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den benutzerdefinierten Kontaktstatustyp, der zusammen mit Kontaktereignissen an Interact übergeben werden kann.

Neuer Kategoriename

Syntax

Diese Eigenschaft definiert den Namen der Codekategorie des Kontaktstatus.

Code

Syntax

Diese Eigenschaft definiert den benutzerdefinierten Code für Ihren Kontakttyp. Dieser definierte Code muss in der Systemtabelle UA_ContactStatus für HCL Campaign vorhanden sein.

action

Syntax

Die Aktion, die dem angepassten Kontakttypcode entspricht. Die hier definierte Aktion überschreibt die in der Tabelle UA_ContactStatus des Campaign Systems definierte Aktion.

Standardwert

Keine

Gültiger Wert

LogContact | Keine

Interact | Dienste | contactHist | fileCache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der die Kontaktverlaufsdaten sammelt, wenn Sie ein Datenbankladeprogramm verwenden. **Voraussetzung:** Für die Konfiguration Affinium|interact|services|externalLoaderStagingDirectory set loaderStagingData.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten Kontaktverlaufsdaten in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | defaultedStats

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Service, der Statistiken darüber sammelt, wie oft die Standardzeichenfolge für den Interaktionspunkt verwendet wurde.

enableLog

Syntax

Wenn der Wert auf `true` festgelegt ist, ist der Service aktiviert, der Statistiken, wie oft die Standardzeichenfolge für den Interaktionspunkt verwendet wurde, in der `UACI_DefaultedStat`-Tabelle sammelt. Bei `false` werden keine Statistiken über die Standardzeichenfolge gesammelt.

Wenn Sie die IBM-Berichterstellung nicht verwenden, können Sie diese Eigenschaft auf `false` setzen, da keine Datensammlung erforderlich ist.

Standardwert

True

Gültige Werte

True | False

Interact | Dienste | defaultedStats | cache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der Statistiken darüber sammelt, wie oft die Standardzeichenfolge für den Interaktionspunkt verwendet wurde.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten Statistiken über die Standardzeichenfolge in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | eligOpsStats

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Service, der die Statistiken über berechtigte Angebote schreibt.

enableLog

Syntax

Wenn der Wert auf `true` festgelegt ist, ist der Service aktiviert, der Statistiken über berechtigte Angebote sammelt. Bei `false` werden keine Statistiken über berechtigte Angebote gesammelt.

Wenn Sie die IBM-Berichterstellung nicht verwenden, können Sie diese Eigenschaft auf `false` setzen, da keine Datensammlung erforderlich ist.

Standardwert

True

Gültige Werte

True | False

Interact | Dienste | eligOpsStats | cache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der Statistiken über berechtigte Angebote sammelt.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten Statistiken über berechtigte Angebote in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | eventActivity

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Service, der Ereignisaktivitätsstatistiken sammelt.

enableLog

Syntax

Bei dem Wert `true` ist der Service aktiviert, der Ereignisaktivitätsstatistiken sammelt. Bei `false` werden keine Ereignisstatistiken gesammelt.

Wenn Sie die IBM-Berichterstellung nicht verwenden, können Sie diese Eigenschaft auf `false` setzen, da keine Datensammlung erforderlich ist.

Standardwert

True

Gültige Werte

True | False

Interact | Dienste | eventActivity | cache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der Ereignisaktivitätsstatistiken sammelt.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten Ereignisaktivitätsstatistiken in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | eventPattern

Die Konfigurationseigenschaften in der `eventPattern`-Kategorie definieren die Einstellungen für den Service, der Ereignismuster-Aktivitätsstatistik sammelt.

persistUnknownUserStates

Syntax

Gibt an, ob die Ereignismusterstatus für eine unbekannte Zielgruppen-ID (Besucher) in der Datenbank gespeichert bleiben. Standardmäßig werden die Statusangaben aller aktualisierten Ereignismuster, die der Zielgruppen-ID des Besuchers zugeordnet sind, beim Ende einer Sitzung in der Datenbank gespeichert. Voraussetzung hierfür ist, dass die Zielgruppen-ID bekannt ist (d.h., dass das Profil des Besuchers in der Profildatenquelle enthalten ist).

Die Eigenschaft `persistUnknownUserStates` legt fest, welche Aktion ausgeführt werden soll, wenn die Zielgruppen-ID nicht bekannt ist. Standardmäßig ist diese Eigenschaft auf den Wert `False` gesetzt. Für unbekannte Zielgruppen-IDs werden die Ereignismusterstatus am Ende der Sitzung gelöscht.

Wenn Sie diese Eigenschaft auf `True` setzen, werden die Ereignismusterstatus unbekannter Benutzer (deren Profil im Datenservice für konfigurierte Profile nicht gefunden werden kann) dauerhaft gespeichert.

Standardwert

False

Gültige Werte

True | False

mergeUnknowUserInSessionStates

Syntax

Legt fest, wie die Ereignismusterstatus für unbekannte Zielgruppen-IDs (Besucher) aufbewahrt werden sollen. Wenn sich die Zielgruppen-ID während einer Sitzung ändert, versucht Unica Interact, die gespeicherten Ereignismusterstatus für die neue Zielgruppen-ID aus der Datenbanktabelle zu laden. Wenn die Zielgruppen-ID vorher unbekannt war und die Eigenschaft `mergeUnknowUserInSessionStates` auf `True` gesetzt wird, werden die

Benutzerereignisaktivitäten, die zur vorherigen Zielgruppen-ID in derselben Sitzung gehören, in der neuen Zielgruppen-ID zusammengeführt.

Standardwert

False

Gültige Werte

True | False

enableUserEventLog

Syntax

Legt fest, ob Benutzerereignisaktivitäten in der Datenbank protokolliert werden.

Standardwert

False

Gültige Werte

True | False

Interact | Dienste | eventPattern | userEventCache

Die Konfigurationseigenschaften in der Kategorie `userEventCache` definieren die Einstellungen, die bestimmen, wann eine Ereignisaktivität aus dem Cache zur dauerhaften Speicherung in die Datenbank verschoben wird.

Schwellenwert

Syntax

Bestimmt die maximale Anzahl von Ereignismusterstatus, die im Ereignismusterstatus-Cache gespeichert werden können. Wenn das Limit erreicht ist, werden die am längsten nicht verwendeten Status aus dem Cache gelöscht.

Standardwert

100

Gültige Werte

Die gewünschte Anzahl von Ereignismusterstatus, die im Cache bleiben sollen.

insertPeriodInSecs

Syntax

Bestimmt die maximale Zeitdauer in Sekunden, für die Benutzerereignisaktivitäten in die Warteschlange des Speichers eingereiht werden. Wenn das durch diese Eigenschaft angegebene Zeitlimit erreicht ist, werden diese Aktivitäten dauerhaft in der Datenbank gespeichert.

Standardwert

3600 (60 Minuten)

Gültige Werte

Die gewünschte Anzahl von Sekunden.

Interact | Dienste | eventPattern | advancedPatterns

Die Konfigurationseigenschaften in dieser Kategorie steuern, ob die Integration mit Unica Interact Advanced Patterns aktiviert wird, und sie definieren die Zeitlimitintervalle für Verbindungen mit Unica Interact Advanced Patterns.

enableAdvancedPatterns

Syntax

Wenn `True` festgelegt ist, ist die Integration mit Unica Interact Advanced Patterns aktiviert. Wenn `False` festgelegt ist, ist die Integration nicht aktiviert. Wenn die Integration zuvor aktiviert war, verwendet Unica Interact die jüngsten von Unica Interact Advanced Patterns erhaltenen Musterstatus.

Standardwert

True

Gültige Werte

True | False

connectionTimeoutInMilliseconds

Syntax

Die maximale Zeitdauer bis zum Herstellen einer HTTP-Verbindung von der Unica Interact-Echtzeitumgebung zu Unica Interact Advanced Patterns. Wenn die Anfrage das Zeitlimit überschreitet, verwendet Unica Interact die zuletzt aus Mustern gespeicherten Daten.

Standardwert

30

readTimeoutInMilliseconds

Syntax

Die maximale Zeitdauer bis zum Empfang von Daten, nachdem eine HTTP-Verbindung zwischen der Unica Interact-Echtzeitumgebung und Unica Interact Advanced Patterns eingerichtet und eine Anfrage an Unica Interact Advanced Patterns gesendet wurde, um den Status eines Ereignismusters abzurufen. Wenn die Anfrage das Zeitlimit überschreitet, verwendet Unica Interact die zuletzt aus Mustern gespeicherten Daten.

Standardwert

100

connectionPoolSize

Syntax

Die Größe des HTTP-Verbindungspools für die Kommunikation zwischen der Unica Interact-Echtzeitumgebung und Unica Interact Advanced Patterns.

Standardwert

10

Interact | Dienste | eventPattern | advancedPatterns | autoReconnect

Die Konfigurationseigenschaften in dieser Kategorie geben Parameter für die Funktion der automatischen Verbindungswiederholung in der Integration mit Unica Interact Advanced Patterns an.

aktivieren

Syntax

Bestimmt, ob das System automatisch die Verbindung wiederherstellt, wenn Verbindungsprobleme zwischen der Unica Interact-Echtzeitumgebung und Unica Interact Advanced Patterns auftreten. Mit dem Standardwert **True** wird diese Funktion aktiviert.

Standardwert

True

Gültige Werte

True | False

durationInMinutes

Syntax

Diese Eigenschaft gibt das Zeitintervall in Minuten an, während dem das System wiederholte Verbindungsprobleme auswertet, die zwischen der Unica Interact-Echtzeitumgebung und Unica Interact Advanced Patterns auftreten.

Standardwert

10

numberOfFailuresBeforeDisconnect

Syntax

Diese Eigenschaft gibt die Anzahl der während des angegebenen Zeitraums zulässigen Verbindungsfehler an, bevor das System automatisch die Verbindung zu Unica Interact Advanced Patterns trennt.

Standardwert

3

consecutiveFailuresBeforeDisconnect

Syntax

Diese Eigenschaft bestimmt, ob die Funktion der automatischen Wiederverbindung nur aufeinanderfolgende Fehler der Verbindung zwischen der Unica Interact-Echtzeitumgebung und Unica Interact Advanced Patterns auswertet. Wenn Sie diesen Wert auf **False** setzen, werden alle Ausfälle innerhalb des angegebenen Zeitintervalls ausgewertet.

Standardwert

True

sleepBeforeReconnectDurationInMinutes

Syntax

Diese Eigenschaft gibt die Zeitdauer in Minuten an, die das System wartet, bevor es nach dem Trennen der Verbindung aufgrund von wiederholten Fehlern (wie in den anderen Eigenschaften in dieser Kategorie definiert) die Verbindung wiederherstellt.

Standardwert

5

sendNotificationAfterDisconnect

Syntax

Diese Eigenschaft bestimmt, ob das System eine E-Mail-Benachrichtigung sendet, wenn ein Verbindungsfehler auftritt. Die Benachrichtigung enthält den Namen der Unica Interact-Echtzeitinstanz, bei der die Fehler aufgetreten sind,

und die Zeitdauer bis zur Verbindungswiederholung, wie in der Eigenschaft **sleepBeforeReconnectDurationInMinutes** definiert. Wenn der Standardwert **True** festgelegt ist, werden Benachrichtigungen gesendet.

Standardwert

True

Interact | Dienste | customLogger

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Service, der benutzerdefinierte Daten sammelt, um sie in eine Tabelle zu schreiben (ein Ereignis, das den Ereignisparameter `UACICustomLoggerTableName` verwendet).

enableLog

Syntax

Wenn der Wert auf `true` festgelegt ist, ist die Funktion zum Konvertieren des benutzerdefinierten Protokolls in eine Tabelle aktiviert. Bei `false` hat der Ereignisparameter `UACICustomLoggerTableName` keine Auswirkung.

Standardwert

True

Gültige Werte

True | False

Interact | Dienste | customLogger | cache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der benutzerdefinierte Daten in einer Tabelle sammelt (ein Ereignis, das den Ereignisparameter `UACICustomLoggerTableName` verwendet).

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten benutzerdefinierten Daten in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | responseHist

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Service, der in die Staging-Tabellen für den Antwortverlauf schreibt.

enableLog

Syntax

Wenn der Wert auf `true` festgelegt ist, ist der Service, der in die Staging-Tabellen für den Antwortverlauf schreibt, aktiviert. Bei `false` werden keine Daten in die Staging-Tabellen für den Antwortverlauf geschrieben.

Die Staging-Tabelle für den Antwortverlauf wird durch die Eigenschaft `responseHistoryTable` für die Zielgruppenebene definiert. Die Standardeinstellung ist `UACI_RHStaging`.

Standardwert

True

Gültige Werte

True | False

cacheType

Syntax

Definiert, ob sich der Cache im Speicher oder in einer Datei befindet. Sie können `External Loader File` nur verwenden, wenn Sie Unica Interact für die Verwendung eines Datenbankladeprogramms konfiguriert haben.

Wenn Sie `Memory Cache` auswählen, verwenden Sie die Kategorieeinstellungen `cache`. Wenn Sie `External Loader File` auswählen, verwenden Sie die Kategorieeinstellungen `fileCache`.

Standardwert

Memory Cache

Gültige Werte

Memory Cache | External Loader File

actionOnOrphan

Syntax

Diese Einstellung legt fest, wie mit Antwortereignissen zu verfahren ist, für die noch keine entsprechenden Kontaktereignisse gesendet wurden. Die Einstellung bezieht sich auf Antwortereignisse während der Sitzung. Bei dem Wert `NoAction` wird das Antwortereignis so verarbeitet als ob das entsprechende Kontaktereignis gesendet wurde. Bei dem Wert `Warning` wird das Antwortereignis so verarbeitet als ob das entsprechende Kontaktereignis gesendet wurde, aber es wird eine Warnung in das Protokoll `interact.log` geschrieben. Bei dem Wert `Skip` wird die Antwort nicht verarbeitet, und es wird eine Fehlernachricht in das Protokoll `interact.log` geschrieben. Die hier ausgewählte Einstellung tritt unabhängig von der Aktivierung der Antwortverlaufsprotokollierung in Kraft.

Standardwert

NoAction

Gültige Werte

NoAction | Warnung | Überspringen

suppressionActionOnResponse

Syntax

Diese Einstellung verwaltet die Unterdrückung eines Angebots, das durch ein Antwortereignis in einer Sitzung beantwortet wird. Sie bietet die folgenden vier Optionen.

- NoSuppression. Dieses Angebot nicht unterdrücken
- SuppressionTillAudienceChange. Dieses Angebot wird unterdrückt, bis die aktive Zielgruppen-ID in dieser Sitzung geändert wird.
- SuppressionForAudience. Dieses Angebot wird unterdrückt, solange die aktive Zielgruppen-ID in dieser Sitzung mit der ID übereinstimmt, die bei der Rückgabe dieses Angebots verwendet wurde.
- SuppressionInSession. Dieses Angebot wird während der gesamten Sitzung unterdrückt, auch wenn die Zuschauer-ID geändert wird.

Nachfolgend sehen Sie ein Beispiel mit einer API-Sequenz.

1. startSession (Zielgruppe1)
2. getOffers -> Angebot A zurückgeben
3. postEvent (Kontakt von Angebot A)
4. postEvent (Angebot A annehmen oder ablehnen)
5. getOffers
6. setAudience (Zielgruppe 2)
7. getOffers
8. setAudience (Zielgruppe 1)
9. getOffers

Standardwert

SuppressionTillAudienceChange

Gültige Werte

NoSuppression | SuppressionTillAudienceChange| SuppressionForAudience| SuppressionInSession

Aus der folgenden Tabelle lässt sich erkennen, ob Angebot A in den Schritten 5, 7 und 9 unterdrückt wird.

Einstellung	Schritt 5	Schritt 7	Schritt 9
NoSuppression	N	N	N
SuppressionTillAudienceChange	J	N	N
SuppressionForAudience	J	N	J
SuppressionInSession	J	J	J

Interact | Dienste | responseHist | cache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der die Antwortverlaufsdaten sammelt. **Anmerkung**: Wenn contactHist oder responseHist für die Verwendung von memoryCache konfiguriert ist, können Sie optional eine Datenquelle systemTablesDataSource erstellen und die Einstellungen unter Affinium|interact|allgemein|systemTablesDataSource|loaderProperties konfigurieren. Wenn dies durchgeführt wird, werden die Kontakt-/Antwort-Historie-Staging-Einträge in Dateien im Verzeichnis, wie durch Affinium|interact|Dienste|externalLoaderStagingDirectory festgelegt, persistiert, wenn die Persistenz in der Datenbank fehlschlägt. Andernfalls wird bei der Initialisierung ein INFO-Eintrag protokolliert, der angibt, dass das Failover nicht aktiviert ist.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten Antwortverlaufsdaten in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | response Hist | responseTypeCodes

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Antwortverlaufservice.

Neuer Kategoriename

Syntax

Der Name Ihres Antworttypcodes.

Code

Syntax

Der angepasste Code Ihres Antworttyps.

Standardwert

Der angepasste Code, der zur Tabelle "UA_UsrResponseType" hinzugefügt wurde.

action

Syntax

Die Aktion, die dem angepassten Antworttypcode entspricht.

Die Aktion, die für das Ereignis definiert ist, das veröffentlicht wird, überschreibt die hier definierte Aktion. Wird ein Ereignis "logAccept" ohne eine Angabe für "responseTypeCode" veröffentlicht, dann wird dieses Ereignis aus diesem Grund als Annahmeereignis behandelt. Wenn ein Ereignis "logAccept"

mit einem Wert für "responseTypeCode" veröffentlicht wird, der in dieser Konfiguration vorhanden ist, dann wird die konfigurierte Aktion verwendet, um festzustellen, ob es sich um ein Annahmeereignis handelt. Wenn ein Ereignis "logAccept" mit einer Angabe für "responseTypeCode" veröffentlicht wird, die in der aktuellen Konfiguration nicht vorhanden ist, dann wird dieses Ereignis nicht als Annahmeereignis behandelt. Wenn ein Ereignis als Annahmeereignis behandelt wird, dann werden die Lernstatistiken entsprechend aktualisiert, wenn die Lernfunktion aktiviert ist. Angebotsausdrucksregeln werden ausgewertet, wenn eine der Regeln auf der Annahme dieses Angebots basiert.

Standardwert

Keine

Gültige Werte

LogAccept | LogReject | Keine

Interact | Dienste | responseHist | fileCache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der die Antwortverlaufsdaten sammelt, wenn Sie ein Datenbankladeprogramm verwenden.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor sie von Unica Interact in die Datenbank geschrieben werden.

`responseHist` – Die Tabelle, die durch die Eigenschaft `responseHistoryTable` für die Zielgruppenebene definiert ist. Die Standardeinstellung ist `UACI_RHStaging`.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die Datenbank.

Standardwert

3600

Interact | Dienste | crossSessionResponse

Die Konfigurationseigenschaften in dieser Kategorie definieren allgemeine Einstellungen für den crossSessionResponse-Service und das xsession-Verfahren. Sie müssen diese Einstellungen nur konfigurieren, wenn Sie die sitzungsübergreifende Antwortverfolgung in Unica Interact verwenden.

enableLog

Syntax

Wenn der Wert auf `true` festgelegt wird, wird der Service crossSessionResponse aktiviert und Unica Interact schreibt Daten in die Staging-Tabellen der sitzungsübergreifenden Antwortverfolgung. Bei dem Wert `false` ist der crossSessionResponse-Service inaktiviert.

Standardwert

False

xsessionProcessIntervalInSecs

Syntax

Die Anzahl der Sekunden zwischen Ausführungen des xsession-Verfahrens. Dieses Verfahren verschiebt Daten aus den Staging-Tabellen für die Antwortverfolgung in die Staging-Tabellen für den Antwortverlauf und das integrierte Lernmodul.

Standardwert

180

Gültige Werte

Eine Ganzzahl größer 0.

purgeOrphanResponseThresholdInMinutes

Syntax

Die Anzahl der Minuten, die der crossSessionResponse-Service wartet, bevor Antworten gekennzeichnet werden, die nicht mit den Kontakten in den Kontakt- und Antwortverlaufstabellen übereinstimmen.

Wenn für eine Antwort kein Treffer in den Kontakt- und Antwortverlaufstabellen gefunden wird, wird die Antwort nach `purgeOrphanResponseThresholdInMinutes` Minuten von Unica Interact in der Spalte `Mark` der xSessResponse-Staging-Tabelle mit dem Wert -1 gekennzeichnet. Diese Antworten können dann manuell zugewiesen oder gelöscht werden.

Standardwert

180

xsessionResponseBatchsize

Syntax

Die Anzahl der sitzungsübergreifenden Antwortdatensätze, die auf einmal verarbeitet werden müssen. Anstatt alle neuen Daten- oder Wiederholungssätze auf einmal zu verarbeiten, durchläuft das System die `xsessionResponseBatchSize`-Datensätze in einer Schleife. Dies stellt eine Leistungsänderung dar, da die Verarbeitung einer großen Anzahl von Datensätzen gleichzeitig zu einer Verlangsamung der Abläufe führen kann.

Standardwert

10000

Gültige Werte

Eine beliebige Ganzzahl größer 0.

generateOnlyOneResponseRecord

Syntax

Wenn eine sitzungsübergreifende Antwort verarbeitet wird, ist diese von Interact mit den verfügbaren Datensätzen des Kontaktverlaufs zu verknüpfen. Manchmal werden aufgrund der vorgegebenen Kriterien (Verfahrenscode oder Angebots-ID) mehrere übereinstimmende Datensätze des Kontaktverlaufs ermittelt. In diesem Fall verwendet Interact die Konfigurationseinstellung "generateOnlyOneResponseRecord", um das Ergebnis zu bestimmen.

Werte

- True: Es wird nur ein Antwortverlaufsdatensatz unter Verwendung des aktuellsten Kontaktverlaufsdatensatzes generiert.
- False: Es wird ein Antwortverlaufsdatensatz für jeden übereinstimmenden Kontaktverlaufsdatensatz generiert.

Standardwert

False

Interact | Dienste | crossSessionResponse | cache

Die Konfigurationseigenschaften in dieser Kategorie definieren die Cache-Einstellungen für den Service, der sessionübergreifende Antwortdaten sammelt.

Schwellenwert

Syntax

Die Anzahl der Datensätze, die angehäuft werden, bevor der flushCacheToDB-Service die gesammelten sessionübergreifenden Antwortdaten in die Datenbank schreibt.

Standardwert

100

insertPeriodInSecs

Syntax

Die Anzahl der Sekunden zwischen erzwungenen Schreibvorgängen in die `XSessResponse`-Tabelle.

Standardwert

3600

Interact | Services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

Die Eigenschaften in diesem Abschnitt definieren, wie die Antwortverfolgung Verfahrenscodes dem Kontakt- und Antwortverlauf zuweist.

SQL

Syntax

Diese Eigenschaft legt fest, ob Unica Interact die systemgenerierte SQL oder die benutzerdefinierte SQL aus der Eigenschaft `OverrideSQL` verwendet.

Standardwert

Verwenden Sie Systemgenerierte SQL

Gültige Werte

Verwenden Sie Systemgenerierte SQL | SQL überschreiben

OverrideSQL

Syntax

Wenn Sie nicht den Standard-SQL-Befehl verwenden, um den Verfahrenscode dem Kontakt- und Antwortverlauf zuzuordnen, geben Sie hier die SQL oder die gespeicherte Prozedur ein.

Dieser Wert wird ignoriert, wenn `SQL` auf `Use System Generated SQL` festgelegt ist.

Standardwert

useStoredProcedure

Syntax

Wenn der Wert auf true steht, muss `OverrideSQL` eine Referenz auf eine gespeicherte Prozedur enthalten, die den Verfahrenscode dem Kontakt- und Antwortverlauf zuweist.

Wenn der Wert auf true steht, muss `OverrideSQL`, falls verwendet, eine SQL-Abfrage sein.

Standardwert

INT32

Gültige Werte

true | false

Typ

Syntax

Der zugewiesene TrackingCodeType, der in der `UACI_TrackingType`-Tabelle in den Laufzeitumgebungstabellen definiert ist. Wenn Sie die `UACI_TrackingType`-Tabelle nicht überarbeiten, muss `Type` 1 sein.

Standardwert

1

Gültige Werte

Eine Ganzzahl, die in der `UACI_TrackingType`-Tabelle definiert ist.

Interact | Dienste | crossSessionResponse | OverridePerAudience | [Zielgruppe] | TrackingCodes | byOfferCode

Die Eigenschaften in diesem Abschnitt definieren, wie die Antwortverfolgung Angebotscodes dem Kontakt- und Antwortverlauf zuweist.

SQL

Syntax

Diese Eigenschaft legt fest, ob Unica Interact die systemgenerierte SQL oder die benutzerdefinierte SQL aus der Eigenschaft `OverrideSQL` verwendet.

Standardwert

Verwenden Sie Systemgenerierte SQL

Gültige Werte

Verwenden Sie Systemgenerierte SQL | SQL überschreiben

OverrideSQL

Syntax

Wenn Sie nicht den Standard-SQL-Befehl verwenden, um den Angebotscode dem Kontakt- und Antwortverlauf zuzuordnen, geben Sie hier die SQL oder die gespeicherte Prozedur ein.

Dieser Wert wird ignoriert, wenn `SQL` auf `Use System Generated SQL` festgelegt ist.

Standardwert

useStoredProcedure

Syntax

Wenn der Wert auf true steht, muss `OverrideSQL` einen Verweis auf eine gespeicherte Prozedur enthalten, die den Angebotscode dem Kontakt- und Antwortverlauf zuweist.

Wenn der Wert auf true steht, muss `OverrideSQL`, falls verwendet, eine SQL-Abfrage sein.

Standardwert

INT32

Gültige Werte

true | false

Typ

Syntax

Der zugewiesene TrackingCodeType, der in der `UACI_TrackingType`-Tabelle in den Laufzeitumgebungstabellen definiert ist. Wenn Sie die `UACI_TrackingType`-Tabelle nicht überarbeiten, muss `Type` 2 sein.

Standardwert

2

Gültige Werte

Eine Ganzzahl, die in der `UACI_TrackingType`-Tabelle definiert ist.

Interact | Dienste | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

Die Eigenschaften in diesem Abschnitt definieren, wie die Antwortverfolgung benutzerdefinierten alternativen Code dem Kontakt- und Antwortverlauf zuweist.

Name

Syntax

Diese Eigenschaft definiert den Namen für den alternativen Code. Dieser Name muss mit dem Namen in der `UACI_TrackingType`-Tabelle in den Tabellen der Laufzeitumgebung übereinstimmen.

Standardwert

OverrideSQL

Syntax

Der SQL-Befehl oder die gespeicherte Prozedur, die den alternativen Code dem Kontakt- und Antwortverlauf nach Angebotscode oder Verfahrenscode zuordnen soll.

Standardwert

useStoredProcedure

Syntax

Wenn der Wert auf true steht, muss `OverrideSQL` einen Verweis auf eine gespeicherte Prozedur enthalten, die den alternativen Code dem Kontakt- und Antwortverlauf zuweist.

Wenn der Wert auf true steht, muss `OverrideSQL`, falls verwendet, eine SQL-Abfrage sein.

Standardwert

INT32

Gültige Werte

true | false

Typ

Syntax

Der zugewiesene TrackingCodeType, der in der `UACI_TrackingType`-Tabelle in den Laufzeitumgebungstabellen definiert ist.

Standardwert

3

Gültige Werte

Eine Ganzzahl, die in der `UACI_TrackingType`-Tabelle definiert ist.

Interact | Services | threadManagement | contactAndResponseHist

Die Konfigurationseigenschaften in dieser Kategorie definieren die Thread-Administrationseinstellungen für die Services, die Daten für die Staging-Tabellen für den Kontakt- und Antwortverlauf sammeln.

corePoolSize

Syntax

Die Anzahl der Threads, die im Pool gespeichert werden, auch wenn sie sich im Leerlauf befinden, um die Daten für den Kontakt- und Antwortverlauf zu sammeln.

Standardwert

5

maxPoolSize

Syntax

Die maximale Anzahl der Threads, die im Pool gespeichert werden, um die Daten für den Kontakt- und Antwortverlauf zu sammeln.

Standardwert

5

keepAliveTimeSecs

Syntax

Wenn die Anzahl der Threads größer als der Kern ist, gibt dieser Wert die maximale Dauer an, die überzählige Threads im Leerlauf auf neue Aufgaben warten, bevor sie beendet werden, um die Daten für den Kontakt- und Antwortverlauf zu sammeln.

Standardwert

5

queueCapacity

Syntax

Die Größe der Warteschlange des Thread-Pools zum Sammeln der Daten für den Kontakt- und Antwortverlauf.

Standardwert

1000

termWaitSecs

Syntax

Beim Herunterfahren des Laufzeitservers gibt dieser Wert die Anzahl der Sekunden an, die darauf gewartet wird, dass die Service-Threads das Sammeln der Daten für den Kontakt- und Antwortverlauf abschließen.

Standardwert

5

Interact | Services | threadManagement | allOtherServices

Die Konfigurationseigenschaften in dieser Kategorie definieren die Thread-Administrationseinstellungen für die Services, die die Berechtigungsstatistiken für Angebote, Ereignisaktivitätsstatistiken, Statistiken zur Verwendung von Standardzeichenfolgen und die aus einem benutzerdefinierten Protokoll in eine Tabelle konvertierten Daten erfassen.

corePoolSize

Syntax

Die Anzahl der Threads, die, auch wenn sie sich im Leerlauf befinden, im Pool für die Services gespeichert werden, die die Berechtigungsstatistiken für Angebote, Ereignisaktivitätsstatistiken, Statistiken zur Verwendung von Standardzeichenfolgen und die aus einem benutzerdefinierten Protokoll in eine Tabelle konvertierten Daten erfassen.

Standardwert

5

maxPoolSize

Syntax

Die maximale Anzahl der Threads, die im Pool für die Services gespeichert werden, die die Berechtigungsstatistiken für Angebote, Ereignisaktivitätsstatistiken, Statistiken zur Verwendung von Standardzeichenfolgen und die aus einem benutzerdefinierten Protokoll in eine Tabelle konvertierten Daten erfassen.

Standardwert

5

keepAliveTimeSecs

Syntax

Wenn die Anzahl der Threads größer als der Kern ist, gibt dieser Wert die maximale Dauer an, die überzählige Threads im Leerlauf auf neue Aufgaben warten, bevor sie für die Services beendet werden, die die Berechtigungsstatistiken für Angebote, Ereignisaktivitätsstatistiken, Statistiken zur Verwendung von Standardzeichenfolgen und die aus einem benutzerdefinierten Protokoll in eine Tabelle konvertierten Daten erfassen.

Standardwert

5

queueCapacity

Syntax

Die Größe der Warteschlange des Thread-Pools für die Services, die die Berechtigungsstatistiken für Angebote, Ereignisaktivitätsstatistiken, Statistiken zur Verwendung von Standardzeichenfolgen und die aus einem benutzerdefinierten Protokoll in eine Tabelle konvertierten Daten erfassen.

Standardwert

1000

termWaitSecs

Syntax

Beim Herunterfahren des Laufzeitservers gibt dieser Wert die Anzahl der Sekunden an, die im Fall von Services, die die Berechtigungsstatistiken für Angebote, Ereignisaktivitätsstatistiken, Statistiken zur Verwendung von Standardzeichenfolgen und die aus einem benutzerdefinierten Protokoll in eine Tabelle konvertierten Daten erfassen, darauf gewartet wird, dass die Service-Threads für die Services abgeschlossen werden.

Standardwert

5

Interact | Dienste | threadManagement | flushCacheToDB

Die Konfigurationseigenschaften in dieser Kategorie definieren die Thread-Administrationseinstellungen für die Threads, die gesammelte Daten im Cache in die Datenbanktabellen der Laufzeitumgebung schreiben.

corePoolSize

Syntax

Die Anzahl der Threads, die im Pool für geplante Threads gespeichert werden, die Daten im Cache in den Datenspeicher schreiben.

Standardwert

5

maxPoolSize

Syntax

Die maximale Anzahl der Threads, die im Pool für geplante Threads gespeichert werden, die Daten im Cache in den Datenspeicher schreiben.

Standardwert

5

keepAliveTimeSecs

Syntax

Wenn die Anzahl der Threads größer als der Kern ist, gibt dieser Wert die maximale Dauer an, die überzählige Threads im Leerlauf auf neue Aufgaben warten, bevor sie für geplante Threads beendet werden, die Daten im Cache in den Datenspeicher schreiben.

Standardwert

5

queueCapacity

Syntax

Die Größe der Warteschlange des Thread-Pools für geplante Threads, die Daten im Cache in den Datenspeicher schreiben.

Standardwert

1000

termWaitSecs

Syntax

Beim Herunterfahren des Laufzeitservers gibt dieser Wert die Anzahl der Sekunden an, die bei geplanten Threads, die Daten im Cache in den Datenspeicher schreiben, darauf gewartet wird, dass die Service-Threads abgeschlossen werden.

Standardwert

5

Interact | Services | threadManagement | eventHandling

Die Konfigurationseigenschaften in dieser Kategorie definieren die Thread-Administrationseinstellungen für die Services, die Daten für die Ereignisbehandlung sammeln.

corePoolSize

Syntax

Die Anzahl der Threads, die im Pool gespeichert werden, auch wenn sie momentan inaktiv sind, und die zum Sammeln der Daten für die Ereignisbehandlung verwendet werden.

Standardwert

1

maxPoolSize

Syntax

Die maximale Anzahl der Threads, die im Pool für die Services gespeichert werden, die zum Sammeln der Daten für die Ereignisbehandlung verwendet werden.

Standardwert

5

keepAliveTimeSecs

Syntax

Wenn die Anzahl der Threads größer als der Kern ist, gibt dieser Wert die maximale Zeitdauer an, die überzählige inaktive Threads auf neue Aufgaben warten, bevor sie für das Sammeln von Daten für die Ereignisbehandlung beendet werden.

Standardwert

5

queueCapacity

Syntax

Die Größe der Warteschlange, die vom Thread-Pool zum Sammeln der Daten für die Ereignisbehandlung verwendet wird.

Standardwert

1000

termWaitSecs

Syntax

Beim Herunterfahren des Laufzeitservers gibt dieser Wert die Anzahl der Sekunden an, die darauf gewartet wird, dass die Service-Threads für die Services abgeschlossen werden, mit denen die Daten zur Ereignisbehandlung gesammelt werden.

Standardwert

5

Interact | Dienste | configurationMonitor

Die Konfigurationseigenschaften in dieser Kategorie ermöglichen Ihnen, die Integration mit Unica Interact Advanced Patterns zu aktivieren oder zu inaktivieren, ohne die Unica Interact-Echtzeitinstanz erneut starten zu müssen. Außerdem definieren sie das Intervall für die Abfrage des Eigenschaftswerts, der die Integration aktiviert.

aktivieren

Syntax

Ist die Eigenschaft auf `True` festgelegt, wird der Service aktiviert, der den Wert der Eigenschaft **Interact | Dienste | eventPattern | advancedPatterns enableAdvancedPatterns** aktualisiert. Ist die Eigenschaft auf `False` festgelegt, müssen Sie die Unica Interact-Echtzeitinstanz erneut starten, wenn Sie den Wert der Eigenschaft **Interact | Dienste | eventPattern | advancedPatterns enableAdvancedPatterns** ändern.

Standardwert

False

Gültige Werte

True | False

refreshIntervalInMinutes

Syntax

Definiert das Zeitintervall für die Abfrage des Werts der Eigenschaft **Interact | Dienste | eventPattern | advancedPatterns enableAdvancedPatterns**.

Standardwert

5

Interact | Dienste | CampaignSegments

isEnabled

Beschreibung

Wenn `True` festgelegt ist, ist diese Funktion aktiviert.

Die Methoden (startSession und setAudience), die die Segmentierung auslösen, führen einen Campaign-API-Aufruf aus, um die Campaign-Segmente für die Zielgruppen-ID abzurufen.

Standardwert

`False`

Gültige Werte

`True` | `False`

ServiceURL

Beschreibung

Die Campaign-Service-URL, zum Beispiel eine der folgenden.

Standardwert

http://localhost:7001/Campaign

corePoolSize

Beschreibung

Die Anzahl der Threads, die im Pool gespeichert werden, auch wenn sie momentan inaktiv sind, und die zum Abruf der Campaign-Segmente verwendet werden.

Standardwert

5

maxPoolSize

Beschreibung

Die maximale Anzahl von Threads, die im Pool für geplante Threads gespeichert und zum Abruf der Campaign-Segmente verwendet werden.

Standardwert

5

keepAliveTimeSecs

Beschreibung

Wenn die Anzahl der Threads größer als der Kern ist, gibt dieser Wert die maximale Zeitdauer an, die überzählige inaktive Threads auf neue Aufgaben warten, bevor sie für den Abruf der Campaign-Segmente beendet werden.

Standardwert

5

queueCapacity

Beschreibung

Die Größe der Warteschlange, die vom Thread-Pool verwendet wird, um die Campaign-Segmente abzurufen.

Standardwert

1000

readTimeoutInMilliseconds

Beschreibung

Standardmäßig erfolgt der Aufruf des Campaign-Service asynchron. Wenn der Parameter `UACIWaitForSegmentation` mit dem Wert `true` durch einen API-

Aufruf übergeben wird, erfolgt dieser API-Aufruf synchron und es wird dabei für die durch diesen Parameter festgelegte Dauer gewartet.

Standardwert

50

Interact | cacheManagement

Diese Gruppe von Konfigurationseigenschaften definiert die Einstellungen für die Auswahl und Konfiguration der einzelnen unterstützten Cache-Manager, die Sie zur Verbesserung der Leistung von Unica Interact verwenden können. Beispiele für solche Cache-Manager sind Ehcache oder Ignite, die in Ihrer Unica Interact-Installation integriert sind. Konfigurieren Sie mithilfe der Konfigurationseigenschaften **Unica Interact | cacheManagement | Cache-Manager** den zu verwendenden Cache-Manager. Geben Sie mit den Konfigurationseigenschaften **Unica Interact | cacheManagement | Caches** an, welcher Cache-Manager für die Verbesserung der Leistung von Unica Interact verwendet werden soll.

Interact | cacheManagement | Cache-Manager

Die Kategorie "Cache Managers" gibt die Parameter für die Cacheverwaltungslösungen an, die Sie zusammen mit Unica Interact verwenden möchten.

Interact | cacheManagement | Cache-Manager | Ehcache

Die Kategorie "EHCache" gibt die Parameter für die EHCache-Cacheverwaltungslösung an, damit Sie diese zur Verbesserung der Leistung von Unica Interact anpassen können.

Interact | Cache Managers | EHCache | Parameter Data

Die Konfigurationseigenschaften in dieser Kategorie steuern die Funktionsweise des EHCache-Cacheverwaltungssystems zum Verbessern der Leistung von Unica Interact.

cacheType

Beschreibung

Sie können die Unica Interact-Laufzeitserver in einer Servergruppe konfigurieren, um eine Multicastadresse für die gemeinsame Nutzung von Cachedaten zu verwenden. Dies wird als verteilter Cache bezeichnet. Der Parameter "cacheType" gibt an, ob Sie den integrierten Caching-Mechanismus EHCache im **lokalen** (eigenständigen) oder **verteilten** Modus (wie bei einer Laufzeitservergruppe) verwenden.

Anmerkung:

Wenn Sie als cacheType **Distributed** (Verteilt) auswählen, müssen alle Server, die den Cache gemeinsam nutzen, derselben einzelnen Servergruppe angehören. Sie müssen außerdem Multicasting zwischen allen Mitgliedern einer Servergruppe aktivieren.

Standardwert

Lokal

Gültige Werte

Local | Distributed

multicastIPAddress

Beschreibung

Wenn Sie den Parameter **cacheType** als "Distributed" (Verteilt) angeben, konfigurieren Sie den Cache so, dass er per Multicasting zwischen allen Mitgliedern einer Unica Interact-Laufzeitservergruppe betrieben wird. Der Wert "multicastIPAddress" ist die IP-Adresse, die alle Unica Interact-Server für die Servergruppe zur Überwachung verwenden.

Die IP-Adresse muss in allen Ihren Servergruppen eindeutig sein.

Standardwert

230.0.0.1

multicastPort

Beschreibung

Wenn Sie den Parameter **cacheType** auf "Distributed" festgelegt haben, gibt der Parameter **multicastPort** den Port an, den alle Unica Interact-Server für die Servergruppe zum Überwachen verwenden.

Standardwert

6363

overflowToDisk

Beschreibung

Der Cache-Manager EHCache verwaltet die Sitzungsdaten unter Verwendung des verfügbaren Speichers. Bei Umgebungen mit einer großen Sitzungsgröße (aufgrund eines großen Profils) ist die Anzahl der unterstützten Sitzungen im Speicher möglicherweise zu klein, um das Kundenszenario zu unterstützen. Für solche Situationen verfügt EHCache über eine optionale Funktion, die es ermöglicht, Cachedaten, die über die vom Speicher unterstützte Menge hinausgehen, stattdessen temporär auf die Festplatte zu schreiben.

Wenn Sie die Eigenschaft **overflowToDisk** auf "Yes" setzen, kann jede Java™ Virtual Machine (JVM) mehr gleichzeitige Sitzungen verwalten, als der Speicher allein zulassen würde.

Standardwert

Nein

Gültige Werte

No | Yes

diskStore

Beschreibung

Wenn für die Konfigurationseigenschaft **overflowToDisk** der Wert Yesfestgelegt ist, gibt diese Konfigurationseigenschaft das Plattenverzeichnis

an, welches die Cacheeinträge enthalten wird, die zu einem Überlauf im Speicher führen. Wenn diese Konfigurationseigenschaft nicht vorhanden oder ihr Wert nicht gültig ist, wird das Plattenverzeichnis automatisch im standardmäßigen temporären Verzeichnis des Betriebssystems erstellt.

Standardwert

Keine

Gültige Werte

Ein Verzeichnis, für das die Webanwendung, welche die Unica Interact-Laufzeit hostet, Schreibberechtigungen besitzt.

(Parameter)

Beschreibung

Eine Vorlage, mit der Sie einen angepassten Parameter erstellen können, der zusammen mit dem Cache-Manager verwendet wird. Sie können einen beliebigen Parameternamen und den für ihn vorgesehenen Wert konfigurieren.

Um einen angepassten Parameter zu erstellen, klicken Sie auf **(Parameter)** und geben Sie den Namen und den Wert ein, die Sie dem Parameter zuweisen möchten. Wenn Sie auf **Änderungen speichern** klicken, wird der von Ihnen erstellte Parameter zur Liste in der Kategorie Parameter Data hinzugefügt.

Standardwert

Keine

Interact | Caches

Mit dieser Gruppe von Konfigurationseigenschaften können Sie angeben, welchen der unterstützten Cache-Manager Sie verwenden möchten, um die Leistung von Unica Interact zu verbessern (beispielsweise Ehcache oder Ignite-Caching), und Sie können bestimmte Cacheeigenschaften für den gerade konfigurierten Laufzeitserver konfigurieren.

Dies umfasst die Caches für das Speichern von Sitzungsdaten, Ereignismusterzuständen und Segmentierungsergebnissen. Durch die Anpassung dieser Einstellungen können Sie

angeben, welche Cachelösung für die einzelnen Caching-Typen verwendet werden soll. Darüber hinaus können Sie einzelne Einstellungen angeben, um die Funktionsweise des Cache zu steuern.

Interact | cacheManagement | caches | InteractCache

Die Kategorie "InteractCache" konfiguriert das Caching für alle Sitzungsobjekte. Zu diesen gehören Profildaten, Segmentierungsergebnisse, zuletzt bereitgestellte Behandlungen, durch API-Methoden übergebene Parameter sowie weitere von der Unica Interact-Laufzeitinstanz verwendete Objekte.

Die Kategorie InteractCache ist erforderlich, damit Interact ordnungsgemäß funktioniert.

Die Kategorie InteractCache kann auch über eine externe EHCache-Konfiguration für Einstellungen konfiguriert werden, die in **Interact | cacheManagement | Caches** nicht unterstützt werden. Wird EHCache verwendet, müssen Sie sicherstellen, dass InteractCache ordnungsgemäß konfiguriert wird.

CacheManagerName

Beschreibung

Der Name des Cache-Managers, der den Unica Interact-Cache verwaltet.
Der hier eingegebene Wert muss einer der in der Konfigurationseigenschaft **Interact | cacheManagement | Cache Managers** definierten Cache-Manager sein, wie z. B. `EHCache` oder `Ignite Scale`.

Standardwert

EHCache

Gültige Werte

Jeder in der Konfigurationseigenschaft **Interact | cacheManagement | Cache Managers** definierte Cache-Manager.

maxEntriesInCache

Beschreibung

Die maximale Anzahl der in diesem Cache zu speichernden Sitzungsdatenobjekte. Wenn die maximale Anzahl der Sitzungsdatenobjekte erreicht ist und Daten für eine zusätzliche Sitzung gespeichert werden müssen, wird das am längsten nicht verwendete Objekt gelöscht.

Standardwert

100000

Gültige Werte

Ganzzahl größer 0.

timeoutInSecs

Beschreibung

Die Zeitdauer in Sekunden, die seit der Verwendung oder Aktualisierung eines Sitzungsdatenobjekts abgelaufen ist. Anhand von dieser Zeitdauer wird bestimmt, wann das Objekt aus dem Cache entfernt wird.

Anmerkung: Wenn Sie ein Upgrade von einer Version vor Version 9.1 durchgeführt haben, dann müssen Sie die Eigenschaft `timeoutInSecs` neu konfigurieren, weil die Eigenschaft verschoben wurde.

Standardwert

300

Gültige Werte

Ganzzahl größer 0.

Interact | Caches | Interact Cache | Parameter Data

Ein Cache-Manager "Ignite" wird unter dem Cache-Manager-Knoten hinzugefügt. Der Cache Unica Interact-Cache und der PatternStateCache können unabhängig voneinander entweder EHCache oder Ignite verwenden. Die folgenden Parameter sind für die Konfiguration verfügbar:

cacheType

Beschreibung

Wenn "Lokal" ausgewählt wird, wird jeder Knoten unabhängig voneinander ausgeführt. Wenn die Option "Verteilt" ausgewählt ist, bilden alle Knoten ein Netz, und die Daten werden über dieses Netz verteilt, was auch die Standardeinstellung ist.

Standardwert

Wenn die Option "Verteilt" ausgewählt ist, bilden alle Knoten ein Netz, und die Daten werden über dieses Netz verteilt, was auch die Standardeinstellung ist.

discoveryIPAddresses

Beschreibung

Die durch Komma getrennte Liste der Adressen der Knoten im Format <IP>:<port> für die gegenseitige Kommunikation der Knoten. Wenn es sich bei einer dieser Adressen um eine Multicast-Adresse handelt, wird die Multicast-Erkennung verwendet. Andernfalls wird die statische IP-Erkennung verwendet, von denen in diesem Fall mindestens eine zu jedem Zeitpunkt aktiv sein muss. Dies ist erforderlich, wenn als Cache-Typ "Verteilt" gewählt wird. Der Standardwert 230.0.0.1:6363 ist ein Multicastwert.

Standardwert

230.0.0.1:6363

localPort

Der Port, den jeder Knoten zur Kommunikation mit anderen Knoten verwendet. Wenn nicht angegeben, wird ein offener Port zwischen 47500 und 47509 verwendet. Bei Verwendung der statischen IP-Erkennung wird empfohlen, diese Einstellung zu konfigurieren. Dieser Wert kann durch die JVM-Eigenschaft "-Dinteract.ignitePort=<valid port>" überschrieben werden.

numberOfBackups

Das sind die Sicherungskopien der Daten, die im Netz gespeichert werden. Ein höherer Wert hat einen besseren Failover-Schutz und eine bessere Leseleistung zu den Kosten einer niedrigeren Schreibleistung. Wenn als Cache-Typ "Verteilt" gewählt wird, setzen Sie den Wert für die Anzahl der Backups auf 1.

overflowToDisk

Gibt an, ob die Daten in einer temporären Datei auf der Festplatte verbleiben.

Hinweis: Wenn eine Instanz stoppt und keine Datensicherung konfiguriert ist, können Anfragen für dieselbe Sitzung auf verschiedenen Instanzen fehlschlagen. Dies bedeutet, dass der API-Aufruf bei zwei verschiedenen RTs fehlschlägt, wenn es sich in einem Fall um den Cache-Typ "Ignite" handelt.

Interact | Caches | Interact Cache | Parameter Data

Die Konfigurationseigenschaften in dieser Kategorie steuern den Interact-Cache, der automatisch von Ihrer Unica Interact-Installation verwendet wird. Diese Einstellungen müssen für jeden Unica Interact-Laufzeitserver separat konfiguriert werden.

asyncIntervalMillis

Beschreibung

Die Zeit in Millisekunden, die der Cache-Manager EHCache warten soll, bevor er Änderungen auf andere Unica Interact-Laufzeitinstanzen repliziert. Wenn der Wert nicht positiv ist, werden diese Änderungen synchron repliziert.

Diese Konfigurationseigenschaft wird nicht standardmäßig erstellt. Wenn Sie diese Eigenschaft erstellen, wird sie nur verwendet, wenn EHCache der Cache-Manager ist und wenn die ehCache-Eigenschaft **cacheType** auf `distributed` festgelegt ist.

Standardwert

Keine.

(Parameter)

Beschreibung

Eine Vorlage, mit der Sie einen angepassten Parameter erstellen können, der zusammen mit dem Interact-Cache verwendet wird. Sie können einen beliebigen Parameternamen und den für ihn vorgesehenen Wert konfigurieren.

Um einen angepassten Parameter zu erstellen, klicken Sie auf **(Parameter)** und geben Sie den Namen und den Wert ein, die Sie dem Parameter zuweisen möchten. Wenn Sie auf **Änderungen speichern** klicken, wird der von Ihnen erstellte Parameter zur Liste in der Kategorie Parameter Data hinzugefügt.

Standardwert

Keine

Interact | cacheManagement | caches | PatternStateCache

In der Kategorie "PatternStateCache" werden die Status von Ereignismustern und Regeln für die Echtzeit-Angebotsunterdrückung gehostet. Standardmäßig ist dieser Cache als Read-through- und Durchschreibcache konfiguriert, sodass Unica Interact versucht, die ersten Ereignismuster- und Angebotsunterdrückungsdaten des Cache zu verwenden. Wenn der angeforderte Eintrag nicht im Cache vorhanden ist, lädt die Cache-Implementierung ihn aus der Datenquelle, entweder über die JNDI-Konfiguration oder direkt mittels einer JDBC-Verbindung.

Um eine JNDI-Verbindung zu verwenden, stellt Unica Interact die Verbindung zu einem vorhandenen Datenquellenprovider her, der über den angegebenen Server mit dem JNDI-Namen, der URL usw. definiert wurde. Für eine JDBC-Verbindung müssen Sie eine Gruppe von JDBC-Einstellungen bereitstellen, einschließlich des Klassennamens des JDBC-Treibers, der Datenbank-URL und Authentifizierungsinformationen.

Sollten Sie mehrere JNDI- und JDBC-Quellen definieren, wird die erste aktivierte JNDI-Quelle verwendet. Wenn es keine aktivierten JNDI-Quellen gibt, wird die erste aktivierte JDBC-Quelle verwendet.

Die Kategorie PatternStateCache ist erforderlich, damit Interact ordnungsgemäß funktioniert.

Die Kategorie PatternStateCache kann auch über eine externe EHCache-Konfiguration für Einstellungen konfiguriert werden, die in **Interact | cacheManagement | Caches** nicht unterstützt werden. Wird EHCache verwendet, müssen Sie sicherstellen, dass PatternStateCache ordnungsgemäß konfiguriert wird.

CacheManagerName

Syntax

Der Name des Cache-Managers, der den Unica Interact-Musterstatuscache verwaltet. Der hier eingegebene Wert muss einer der in der Konfigurationseigenschaft **Interact | cacheManagement | Cache Managers** definierten Cache-Manager sein, wie z. B. `EHCache` oder `Ignite Scale`.

Standardwert

EHCache

Gültige Werte

Jeder in der Konfigurationseigenschaft **Interact | cacheManagement | Cache Managers** definierte Cache-Manager.

maxEntriesInCache

Syntax

Die maximale Anzahl der in diesem Cache zu speichernden Ereignismusterstatus. Wenn die maximale Anzahl der Ereignismusterstatus erreicht ist und Daten für einen zusätzlichen Ereignismusterstatus gespeichert werden müssen, wird das am längsten nicht verwendete Objekt gelöscht.

Standardwert

100000

Gültige Werte

Ganzzahl größer 0.

timeoutInSecs

Syntax

Gibt die Zeit in Sekunden an, nach deren Ablauf für ein Ereignismusterstatus-Objekt im Cache für den Ereignismusterstatus eine Zeitlimitüberschreitung auftritt. Wenn ein solches Statusobjekt für die in `timeoutInSecs` angegebene Anzahl von Sekunden im Cache inaktiv war, kann es basierend auf der LRU-Regel aus dem Cache entfernt werden. Beachten Sie hierbei, dass der Wert dieser Eigenschaft größer als der Wert sein sollte, der in der Eigenschaft `sessionTimeoutInSecs` definiert wurde.

Anmerkung: Wenn Sie ein Upgrade von einer Version vor Version 9.1 durchgeführt haben, dann müssen Sie die Eigenschaft `timeoutInSecs` neu konfigurieren, weil die Eigenschaft verschoben wurde.

Standardwert

300

Gültige Werte

Ganzzahl größer 0.

Interact | Caches | PatternStateCache | Parameter Data

Die Konfigurationseigenschaften in dieser Kategorie steuern den Musterstatuscache, in dem die Status von Ereignismustern und Regeln für die Echtzeit-Angebotsunterdrückung gehostet werden.

(Parameter)

Syntax

Eine Vorlage, mit der Sie einen angepassten Parameter erstellen können, der zusammen mit dem Musterstatuscache verwendet wird. Sie können einen beliebigen Parameternamen und den für ihn vorgesehenen Wert konfigurieren.

Um einen angepassten Parameter zu erstellen, klicken Sie auf **(Parameter)** und geben Sie den Namen und den Wert ein, die Sie dem Parameter zuweisen möchten. Wenn Sie auf **Änderungen speichern** klicken, wird der von Ihnen erstellte Parameter zur Liste in der Kategorie Parameter Data hinzugefügt.

Standardwert

Keine

Interact | cacheManagement | caches | PatternStateCache | loaderWriter

Die Kategorie **loaderWriter** enthält die Konfiguration des Ladeprogramms, das zum Abrufen und dauerhaften Speichern von Ereignismustern mit externen Repositorys interagiert.

className

Syntax

Der vollständig qualifizierte Klassenname für dieses Ladeprogramm. Diese Klasse muss den Anforderungen des gewählten Cache-Managers gerecht werden.

Standardwert

```
com.unicacorp.interact.cache.ehcache.loaderwriter.
PatternStateEHCacheLoaderWriter
```

Gültige Werte

Ein vollständig qualifizierter Klassenname.

classPath

Syntax

Der Pfad zur Klassendatei des Ladeprogramms. Wenn Sie dieses Feld leer lassen oder der Eintrag ungültig ist, wird der für die Ausführung von Unica Interact verwendete Klassenpfad verwendet.

Standardwert

Keine

Gültige Werte

Ein gültiger Klassenpfad.

writeMode

Syntax

Gibt den Modus für das Ausgabeprogramm an, mit dem die neuen oder aktualisierten Ereignismusterstatus im Cache dauerhaft gespeichert werden. Folgende Optionen sind gültig:

- WRITE_THROUGH. Jeder neue Eintrag oder jeder vorhandene Eintrag, der aktualisiert wird, wird sofort in die Repositorys geschrieben.
- WRITE_BEHIND. Der Cache-Manager wartet einige Zeit, um mehrere Änderungen zu sammeln, bevor diese im Stapelbetrieb dauerhaft in den Repositorys gespeichert werden.

Standardwert

WRITE_THROUGH

Gültige Werte

WRITE_THROUGH oder WRITE_BEHIND.

batchSize

Syntax

Die maximale Anzahl von Ereignismusterstatus-Objekten, die das Ausgabeprogramm im Stapelbetrieb dauerhaft speichert. Diese Eigenschaft wird nur verwendet, wenn **writeMode** auf `WRITE_BEHIND` festgelegt wird.

Standardwert

100

Gültige Werte

Ganzzahlwert.

maxDelayInSecs

Syntax

Die maximale Zeitdauer in Sekunden, die der Cache-Manager wartet, bevor ein Ereignismusterstatus-Objekt dauerhaft gespeichert wird. Diese Eigenschaft wird nur verwendet, wenn **writeMode** auf `WRITE_BEHIND` festgelegt wird.

Standardwert

5

Gültige Werte

Ganzzahlwert.

Interact | Caches | PatternStateCache | loaderWriter | Parameter Data

Die Konfigurationseigenschaften in dieser Kategorie steuern das Musterstatuscache-Ladeprogramm.

(Parameter)

Syntax

Eine Vorlage, mit der Sie einen angepassten Parameter erstellen können, der zusammen mit dem Musterstatuscache-Ladeprogramm verwendet wird. Sie können einen beliebigen Parameternamen und den für ihn vorgesehenen Wert konfigurieren.

Um einen angepassten Parameter zu erstellen, klicken Sie auf **(Parameter)** und geben Sie den Namen und den Wert ein, die Sie dem Parameter zuweisen möchten. Wenn Sie auf **Änderungen speichern** klicken, wird der von Ihnen erstellte Parameter zur Liste in der Kategorie Parameter Data hinzugefügt.

Standardwert

Keine

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jndiSettings

Die Kategorie **jndiSettings** enthält die Konfiguration für die JNDI-Datenquelle, die das Ladeprogramm zum Kommunizieren mit der Sicherungsdatenbank verwendet. Um einen

neuen Satz von JNDI-Einstellungen zu erstellen, erweitern Sie die Kategorie **jdniSettings** und klicken Sie auf die Eigenschaft **(jndiSetting)**.

(jndiSettings)

Anmerkung: Wenn WebSphere Application Server verwendet wird, wird loaderWriter nicht mit **jndiSettings** verbunden.

Syntax

Wenn Sie auf diese Kategorie klicken, wird ein Formular angezeigt. Geben Sie zum Definieren einer JNDI-Datenquelle die folgenden Werte ein:

- **Neuer Kategoriename** ist der Name, den Sie zum Identifizieren dieser JNDI-Verbindung verwenden möchten.
- **Aktiviert** ermöglicht Ihnen anzugeben, ob diese JNDI-Verbindung zur Verwendung verfügbar sein soll oder nicht. Legen Sie diesen Wert für neue Verbindungen auf `True` fest.
- **jdniName** ist der JNDI-Name, der bereits beim Einrichten der Datenquelle in dieser definiert wurde.
- **providerUrl** ist die URL zum Auffinden dieser JNDI-Datenquelle. Wenn Sie dieses Feld leer lassen, wird die URL der Webanwendung verwendet, die die Unica Interact-Laufzeitinstanz hostet.
- **Ausgangskontextfactory** ist der vollständig qualifizierte Klassenname der Ausgangskontextfactory-Klasse für die Verbindung zum JNDI-Provider. Wenn die Webanwendung, die die Unica Interact-Laufzeitinstanz hostet, als **providerUrl** verwendet wird, lassen Sie dieses Feld leer.

Standardwert

Keine.

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jdbcSettings

Die Kategorie **jdbcSettings** enthält die Konfiguration für die JDBC-Verbindungen, die das Ladeprogramm zum Kommunizieren mit der Sicherungsdatenbank verwendet. Um einen neuen Satz von JDBC-Einstellungen zu erstellen, erweitern Sie die Kategorie **jdbcSettings** und klicken Sie auf die Eigenschaft **(jdbcSetting)**.

(jdbcSettings)

Syntax

Wenn Sie auf diese Kategorie klicken, wird ein Formular angezeigt. Geben Sie zum Definieren einer JDBC-Datenquelle die folgenden Werte ein:

- **Neuer Kategoriename** ist der Name, den Sie zum Identifizieren dieser JDBC-Verbindung verwenden möchten.
- **Aktiviert** ermöglicht Ihnen anzugeben, ob diese JDBC-Verbindung zur Verwendung verfügbar sein soll oder nicht. Legen Sie diesen Wert für neue Verbindungen auf `True` fest.
- **driverClassName** ist der vollständig qualifizierte Klassenname des JDBC-Treibers. Diese Klasse muss in dem Klassenpfad vorhanden sein, der zum Starten des Hosting-Cache-Servers konfiguriert wurde.
- **databaseUrl** ist die URL zum Auffinden dieser JDBC-Datenquelle.
- **asmUser** ist der Name des Unica-Benutzers, der in dieser JDBC-Verbindung mit den Berechtigungsnachweisen zum Herstellen der Verbindung zur Datenbank konfiguriert wurde.
- **asmDataSource** ist der Name der Unica-Datenquelle, die in dieser JDBC-Verbindung mit den Berechtigungsnachweisen zum Herstellen der Verbindung zur Datenbank konfiguriert wurde.
- **maxConnection** ist die maximale Anzahl gleichzeitig bestehender Verbindungen, die in dieser JDBC-Verbindung zur Datenbank hergestellt werden dürfen.

Standardwert

Keine.

Interact | triggeredMessage

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für alle ausgelösten Nachrichten und Angebotsbereitstellungen in Kanälen.

backendProcessIntervalMin

Syntax

Diese Eigenschaft definiert den Zeitraum in Minuten, in dem der Back-End-Thread verzögerte Angebotsbereitstellungen lädt und verarbeitet. Dieser Wert muss eine Ganzzahl sein. Wenn der Wert null oder negativ ist, wird der Back-End-Prozess inaktiviert.

Gültige Werte

Positive ganze Zahl

autoLogContactAfterDelivery

Syntax

Wenn diese Eigenschaft auf True festgelegt ist, wird automatisch ein Kontaktereignis übergeben, sobald dieses Angebot gesendet wurde oder für eine verzögerte Bereitstellung in die Warteschlange gestellt wurde. Wenn diese Eigenschaft auf False festgelegt ist, wird für die abgehenden Angebote nicht automatisch ein Kontaktereignis übergeben. Dies ist das Standardverhalten.

Gültige Werte

True | False

waitForFlowchart

Syntax

Diese Eigenschaft bestimmt, ob das Ablaufdiagramm warten sollte, bis die derzeit ausgeführte Segmentierung beendet wird, sowie das Verhalten, wenn das Zeitlimit für diese Wartezeit überschritten wird.

DoNotWait: Die Verarbeitung einer ausgelösten Nachricht beginnt unabhängig davon, ob die Segmentierung derzeit ausgeführt wird oder nicht. Wenn jedoch Segmente in der Bedingungsregel verwendet werden und/oder NextBestOffer als Methode zur Angebotsauswahl ausgewählt wurde, wird mit der TM-Ausführung noch gewartet.

OptionalWait: Mit der Verarbeitung einer ausgelösten Nachricht wird gewartet, bis die Segmentierung, die derzeit ausgeführt wird, abgeschlossen wird oder das zulässige Zeitlimit überschritten wird. Wenn die Wartezeit das zulässige Zeitlimit überschreitet, wird eine Warnung protokolliert und die Verarbeitung dieser ausgelösten Nachricht fortgesetzt. Dies ist der Standardwert.

MandatoryWait: Mit der Verarbeitung einer ausgelösten Nachricht wird gewartet, bis die Segmentierung, die derzeit ausgeführt wird, abgeschlossen wird oder das zulässige Zeitlimit überschritten wird. Wenn die Wartezeit das zulässige Zeitlimit überschreitet, wird ein Fehler protokolliert und die Verarbeitung dieser ausgelösten Nachricht abgebrochen.

Gültige Werte

DoNotWait | OptionalWait | MandatoryWait

Interact | triggeredMessage | offerSelection

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für die Angebotsauswahl in ausgelösten Nachrichten.

maxCandidateOffers

Syntax

Diese Eigenschaft definiert die maximale Anzahl der infrage kommenden Angebote, die von der Engine zurückgegeben werden, um das beste Angebot für die Bereitstellung zu bekommen. Es besteht die Möglichkeit, dass keines

dieser zurückgegebenen infrage kommenden Angebote basierend auf dem ausgewählten Kanal gesendet werden kann. Je mehr mögliche Angebote es gibt, desto geringer ist die Wahrscheinlichkeit, dass dies geschieht. Durch mehr mögliche Angebote kann die Verarbeitungszeit jedoch erhöht werden.

Gültige Werte

Positive ganze Zahl

defaultCellCode

Syntax

Wenn das bereitgestellte Angebot das Ergebnis der Auswertung einer strategischen Regel oder eines tabellengesteuerten Datensatzes ist, ist dem Angebot eine Zielzelle zugeordnet und die Informationen aus dieser Zelle werden bei jeder entsprechenden Protokollierung verwendet. Wenn jedoch eine Liste mit bestimmten Angeboten als Eingabe bei der Angebotsauswahl verwendet wird, ist keine Zielzelle verfügbar. In diesem Fall wird der Wert dieser Konfigurationseinstellung verwendet. Sie müssen sicherstellen, dass diese Zielzelle und die zugehörige Kampagne bei der Bereitstellung eingeschlossen werden. Dies geschieht am einfachsten, indem Sie die Zelle zu einer bereitgestellten Strategie hinzufügen.

Interact | triggeredMessage | Dispatchers

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für alle Dispatcher in ausgelösten Nachrichten.

dispatchingThreads

Syntax

Diese Eigenschaft definiert die Anzahl der Threads der Engine, die verwendet werden, um den Dispatcher asynchron aufzurufen. Wenn der Wert 0 oder eine negative Zahl ist, erfolgt das Aufrufen von Dispatchern synchron. Der Standardwert beträgt 0.

Gültige Werte

Eine Ganzzahl

Interact | triggeredMessage | Dispatchers | <dispatcherName>

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für einen bestimmten Dispatcher in ausgelösten Nachrichten.

Kategoriename

Syntax

Diese Eigenschaft definiert den Namen dieses Dispatchers. Der Name muss für jeden Dispatcher eindeutig sein.

Typ

Syntax

Diese Eigenschaft definiert den Dispatchertyp.

Gültige Werte

InMemoryQueue | JMSQueue | Benutzerdefiniert | Kafka

Anmerkung: Für WebSphere und WebLogic wird empfohlen, die aktuell bereitgestellte Version des JVM-Fixpacks zu verwenden. Wenn Sie Kafka in der Vorgängerversion verwendet haben, dann können Sie den Typ in der Upgrade-Version als Kafka einstellen.

JMSQueue unterstützt nur WebLogic. Sie können JMSQueue nicht verwenden, wenn Sie WebSphere Application Server verwenden.

className

Syntax

Diese Eigenschaft definiert den vollständig qualifizierten Klassennamen der Implementierung dieses Dispatchers. Wenn der Typ "InMemoryQueue" lautet, sollte der Wert leer sein. Wenn dieser Typ benutzerdefiniert ist, muss diese Einstellung den folgenden Wert haben:

"`com.unicacorp.interact.eventhandler.triggeredmessage.dispatchers.KafkaDispatcher`". Für den Typ Kafka muss der Wert leer gelassen werden.

classPath

Syntax

Diese Eigenschaft definiert die URL der JAR-Datei, die die Implementierung dieses Dispatchers umfasst. Für den Typ Kafka muss der Wert leer gelassen werden.

Interact | triggeredMessage | dispatchers | <dispatcherName> | Parameter Data

Die Konfigurationseigenschaften in dieser Kategorie definieren Parameter für einen bestimmten Dispatcher in ausgelösten Nachrichten.

Sie können zwischen drei Typen von Dispatchern auswählen. InMemoryQueue ist der interne Dispatcher für Unica Interact. Bei Kafka wird benutzerdefiniert verwendet. JMSQueue wird verwendet, um über JNDI eine Verbindung zu einem JMS-Provider herzustellen. Kafka wird als Streaming-Plattform vertrieben, auf der die Datenströme veröffentlicht und subskribiert werden.

Kategoriename

Beschreibung

Diese Eigenschaft definiert den Namen dieses Parameters. Der Name muss für jeden Parameter für diesen Dispatcher eindeutig sein.

Wert

Beschreibung

Diese Eigenschaft definiert die Parameter, im Format von Name/Wert-Paaren, die von diesem Dispatcher benötigt werden.

Note: Bei allen Parametern für Auslösenachrichten muss die Groß-/Kleinschreibung beachtet werden und sie sollten wie im Folgenden dargestellt eingegeben werden.

Für den Typ InMemoryQueue wird der folgende Parameter unterstützt.

- queueCapacity: Optional. Die maximale Anzahl der Angebote, die in der Warteschlange warten können, bis sie gesendet werden. Ist diese Eigenschaft angegeben, muss es eine positive Ganzzahl sein. Ist die Eigenschaft nicht angegeben oder ungültig, wird der Standardwert (1000) verwendet.

Für den Typ Custom werden die folgenden Parameter unterstützt.

- providerUrl: <hostname>:port (Groß-/Kleinschreibung beachten)
- queueManager: Der Name des Queue Managers, der auf dem Kafka Server erstellt wurde.
- messageQueueName: Der Name der Nachrichten-Queue, die auf dem Kafka Server erstellt wurde.
- enableConsumer: Diese Eigenschaft muss auf true gesetzt werden.
- asmUserforMQAuth: Der Benutzername für die Anmeldung am Server. Dieses ist erforderlich, wenn der Server eine Authentifizierung erzwingt. Andernfalls sollte es nicht angegeben werden.
- authDS: Das mit dem Benutzernamen verknüpfte Passwort für die Anmeldung am Server. Dieses ist erforderlich, wenn der Server eine Authentifizierung erzwingt. Andernfalls sollte es nicht angegeben werden.

Für den Typ JMSQueue wird der folgende Parameter unterstützt.

- providerUrl: Die URL zum JNDI-Provider (Groß-/Kleinschreibung beachten).
- connectionFactoryJNDI: Der JNDI-Name der JMS Connection Factory.

- messageQueueJNDI: Der JNDI-Name der JMS Queue, an die die ausgelösten Nachrichten gesendet und von der abgerufen werden.
- enableConsumer: Diese Eigenschaft gibt an, ob ein Kunde dieser ausgelösten Nachrichten in Unica Interact gestartet werden muss. Diese Eigenschaft muss auf True festgelegt sein. Ist die Eigenschaft nicht angegeben, wird der Standardwert (False) verwendet.
- initialContextFactory: Der vollständig qualifizierte Name der JNDI Ausgangskontext-Factory-Klasse. Wenn Sie WebLogic verwenden, sollte der Wert dieses Parameters `weblogic.jndi.WLInitialContextFactory` lauten.

Für den Typ Kafka wird der folgende Parameter unterstützt.

- providerUrl: Eine Liste von Paaren des Host/Ports, die für die Herstellung der ersten Verbindung mit dem Kafka-Cluster verwendet werden sollen. Diese Liste muss in der Form `host1:port1,host2:port2,...` vorliegen.
- Thema: Ein Thema ist eine Kategorie oder ein Feedname, zu dem Nachrichten gespeichert und veröffentlicht werden. Alle Nachrichten von Kafka sind nach Themen geordnet. Möchten Sie eine Nachricht senden, können Sie sie zu einem bestimmten Thema senden, und wenn Sie eine Nachricht lesen möchten, können Sie sie von einem bestimmten Thema aus lesen. Herstelleranwendungen geben Daten in Themen ein und Verbraucheranwendungen lesen aus Themen. Der Name des Themas muss ein alphanumerisches ASCII-Zeichen, '.', '_' und '-' enthalten. Aufgrund der Einschränkungen bei den Themennamen können Sie entweder Themen mit einem Punkt ('.') oder einem Unterstrich ('_') verwenden. Der Name eines Themas kann maximal 255 Zeichen lang sein. Wenn Sie beispielsweise einen Themennamen 'InteractTM_1' erstellen oder angeben und ein Thema wie 'InteractTM.1' erstellen, wird der folgende Fehler generiert. 'Thema InteractTM.1 entspricht dem bestehenden Themen: InteractTM _1.'
- group.id: Gibt den Namen der Kundengruppe an, zu der ein Kafka Kunde gehört.

- zookeeper.connect: Gibt die Zookeeper-Verbindungszeichenfolge in Form von `hostname:port` an, wobei hostname und port der Host und der Port eines ZooKeeper-Servers sind.
- authentication: Benutzer können Kafka verwenden, indem sie verschiedene Authentifizierungsmechanismen aktivieren.

Obligatorische Parameter für das Veröffentlichen und Abonnieren von Nachrichten

Standardmäßig unterstützt der Kafka-Server keinen Authentifizierungsmechanismus. Benutzer können den Kafka-Server starten, wenn der Authentifizierungsmechanismus deaktiviert ist. In diesem Fall können Sie den Parameter "Authentifizierung" auf den Wert "None" setzen.

Table 28. Obligatorische Parameter für die Veröffentlichung von Nachrichten

Parameter	Zulässige/Beispielparameterwerte
providerURL	<host>:<port> (Beispiel: local-host:9092)
topic	Beliebige Zeichenfolge (Beispiel: InteractTM)
Authentifizierung	keine | Einfach | SSL | SASL_SSL
zookeeper.connect	<host>:<port> (Beispiel: local-host:2181)

Table 29. Obligatorische Parameter für die Subskription von Nachrichten

Parameter	Zulässiger/Beispielparameterwert
providerURL	<host>:<port> (Beispiel: local-host:9092)

Table 29. Obligatorische Parameter für die Subskription von Nachrichten (continued)

Parameter	Zulässiger/Beispielparameterwert
group.id	Beliebige Zeichenfolge (Beispiel: InteractTMGateway)
topic	Beliebige Zeichenfolge (Beispiel: InteractTM)
Authentifizierung	keine | Einfach | SSL | SASL_SSL
zookeeper.connect	<host>:<port> (Beispiel: local-host:2181)

Authentifizierungsmechanismus

Sie können Kafka verwenden, indem Sie verschiedene Authentifizierungsmechanismen aktivieren.

Authentifizierung durch den SASL_PLAIN-Mechanismus

Wenn Sie den SASL_PLAIN-Authentifizierungsmechanismus verwenden möchten, müssen Sie den Parameter "authentication" auf den Wert "Plain" zusammen mit den unterstützten Parametern setzen.

Wenn der SASL_PLAIN-Mechanismus unterstützt wird, müssen die folgenden Parameter angegeben werden.

- asmUserforMQAuth: Der Benutzername für die Anmeldung am Server. Dieses ist erforderlich, wenn der Server eine Authentifizierung erzwingt.
- authDS: Das mit dem Benutzernamen verknüpfte Passwort für die Anmeldung am Server.
- Benutzername/Passwort: Der Benutzername oder das Kennwort des Kafka-Servers, der in der JASS Konfigurationsdatei konfiguriert wird.

Die folgende Tabelle enthält die für den SASL_PLAIN-Mechanismus erforderlichen Parameter.

Parameter	Zulässige/Beispielparameterwerte
Authentifizierung	Normal
asmUserforMQAuth	Beliebige Zeichenfolge (Beispiel: test_user)
authDS	Beliebige Zeichenfolge (Beispiel: authDS)
benutzername	Beliebige Zeichenfolge (Beispiel: test_user)
Kennwort	Beliebige Zeichenfolge (Beispiel: test-secret)

Wenn der "Authentifizierungs"-Parameter "Plain" lautet, müssen Sie entweder asmUserforMQAuth/authDS oder Benutzername/-Kennwortparameter für die Authentifizierung verwenden.

Erstellen Sie die Datenquellen (authDS) im Abschnitt "Benutzer" in der Plattformkonfiguration. Einzelheiten zu den Datenquellen sind im folgenden Beispiel aufgeführt.

Datenquelle	Username	Kennwort
authDS	test_user	test-secret

Authentifizierung durch SSL-Mechanismus

Wenn Sie den SSL-Authentifizierungsmechanismus verwenden möchten, müssen Sie den Parameter "authentication" zusammen mit den unterstützten Parametern auf den Wert "SSL" setzen.

Die folgenden Parameter sind zur Unterstützung des SSL-Mechanismus erforderlich.

- ssl.keystore.location: Der Speicherort der Key-Store Datei. Sie können sie für eine Zweiwege-Authentifizierung für den Client verwenden.
- ssl.truststore.location: Der Speicherort der Trust-Store Datei.
- SSLKeystoreDS: Der Name der Keystore-Datenquelle, die das Kennwort des SSL-Keystore speichert.
- SSLKeyDS: Der Schlüssel-Datenquellenname, der das Passwort des SSL-Schlüssels speichert.
- SSLTruststoreDS: Der Name der Truststore-Datenquelle, die das Passwort des SSL-Truststore speichert.

Die folgende Tabelle enthält die unterstützten Parameter für den SSL-Mechanismus.

Parameter	Zulässige/Beispielparameterwerte
Authentifizierung	SSL
ssl.keystore.location	SSL Keystore Speicherort (Beispiel: `C:/SSL/kafka.client.key-store.jks`)
ssl.truststore.location	SSL Keystore Speicherort (Beispiel: `C:/SSL/kafka.client.truststore.jks`)
asmUserforMQAuth	Beliebige Zeichenfolge (Beispiel: test_user)
SSLKeystoreDS	Beliebige Zeichenfolge (Beispiel: SSLKeystoreDS)
SSLKeyDS	Beliebige Zeichenfolge (Beispiel: SSLKeyDS)
SSLTruststoreDS	Beliebige Zeichenfolge (Beispiel: SSLTruststoreDS)

Erstellen Sie die Datenquellen (SSLKeystoreDS, SSLKeyDS und SSLTruststoreDS) im Abschnitt Benutzer in der Plattformkonfiguration. Einzelheiten zu den Datenquellen sind im folgenden Beispiel aufgeführt.

Datenquelle	Username	Kennwort
SSLKeystoreDS	Keystore	keystore-secret
SSLKeyDS	key	key-secret
SSLTruststoreDS	Truststore	truststore -secret

Note: Der Client-Keystore oder Truststore ist auf der Seite des Herstellers oder des Kunden in der Anwendung Unica Interact (wo die Interact Anwendung installiert wird) erforderlich. `C:/SSL/kafka.client.keystore.jks` und `C:/SSL/kafka.client.truststore.jks` sind die lokalen Speicherorten an denen die Interact Anwendung installiert wird.

Authentifizierung durch Kerbrose

Kerbrose wird als Authentifizierungsmethode in Kafka Empfänger und Kafka Ausgehende Gateway verwendet.

Um Kerbrose verwenden zu können, müssen zusätzlich zu den Parametern für 'Authentifizierung durch SSL Mechanismus' die folgenden Parameter mit ihren Werten auf den Orchestrator-Empfänger der Aktivitäten oder das Trigger-Message-Outbound Gateway gesetzt werden.

- `authentication = SASL_SSL`
- `sasl.mechanism = GSSAPI`

Darüber hinaus müssen dem Anwendungsserver, der die Interact Laufzeit hostet, die folgenden JVM Parameter hinzugefügt werden.

- `-Djava.security.auth.login.config=/path/to/jaas.conf`
- `-Djava.security.krb5.conf=/path/to/krb5.conf`

Authentifizierung durch den SASL_SSL-Mechanismus

Wenn Sie den SASL_SSL-Authentifizierungsmechanismus verwenden möchten, müssen Sie den Parameter "authentication" zusammen mit den unterstützten Parametern auf den Wert "SASL_SSL" setzen. Der SASL_SSL-Mechanismus besteht aus einer Kombination von SASL_PLAIN- und SSL-Mechanismen. Die folgende Tabelle enthält die unterstützten Parameter für den SASL_SSL-Mechanismus.

Parameter	Zulässige/Beispielparameterwerte
Authentifizierung	SASL_SSL
asmUserforMQAuth	Beliebige Zeichenfolge (Beispiel: test_user)
authDS	Beliebige Zeichenfolge (Beispiel: authDS)
benutzername	Beliebige Zeichenfolge (Beispiel: test_user)
Kennwort	Beliebige Zeichenfolge (Beispiel: test-secret)
ssl.keystore.location	SSL Keystore Speicherort (Beispiel: `C:/SSL/kafka.client.key-store.jks`)
ssl.truststore.location	SSL Keystore Speicherort (Beispiel: `C:/SSL/kafka.client.truststore.jks`)
SSLKeystoreDS	Beliebige Zeichenfolge (Beispiel: SSLKeystoreDS)

Parameter	Zulässige/Beispielparameterwerte
SSLKeyDS	Beliebige Zeichenfolge (Beispiel: SSLKeyDS)
SSLTruststoreDS	Beliebige Zeichenfolge (Beispiel: SSLTruststoreDS)

Wenn der "Authentifizierungs"-Parameter "SASL_SSL" lautet, müssen Sie entweder asmUserforMQAuth/authDS oder Benutzername/Kennwort verwenden.

Erstellen Sie die Datenquellen (authDS, SSLKeystoreDS, SSLKeyDS und SSLTruststoreDS) im Abschnitt "Benutzer" in der Plattformkonfiguration. Einzelheiten zu den Datenquellen sind im folgenden Beispiel aufgeführt.

Datenquelle	Username	Kennwort
authDS	Administrator	admin-secret
SSLKeystoreDS	Keystore	test1234
SSLKeyDS	key	test1234
SSLTruststoreDS	Truststore	test1234

Note: Wenn Sie Datenquellen wie authDS, SSLKeystoreDS, SSLKeyDS oder SSLTruststoreDS im Konfigurationsparameter der Plattform angeben, müssen Sie auch den Parameter asmUserforMQAuth angeben.

Der Client-Keystore/Truststore ist auf der Seite des Herstellers/des Kunden in der Anwendung Unica Interact (wo die Unica Interact Anwendung installiert wird) erforderlich. `C:/SSL/kafka.client.keystore.jks` und `C:/SSL/kafka.client.truststore.jks` sind die lokalen Speicherorten an denen die Interact Anwendung installiert wird.

Obligatorische Parameter für die Veröffentlichung von Nachrichten

Die folgenden optionalen Parameter können für die Veröffentlichung von Nachrichten verwendet werden.

- acks: Die acks Konfiguration steuert die Kriterien, unter denen Anfragen als abgeschlossen betrachtet werden. Die Einstellung "alle" führt zu einer Blockierung der vollständigen Übertragung des Datensatzes.
- Wiederholungen: Sollte die Anfrage fehlschlagen, kann der Hersteller erneut versuchen. Da die angegebenen Wiederholungsversuche auf 0 gesetzt sind, ist eine Wiederholung nicht möglich. Das Aktivieren von Wiederholungsversuchen kann zu Überschneidungen führen.
- batch.size: Die Standard-Batch-Größe wird in Bytes angegeben, wenn mehrere Datensätze als Batches an eine Partition gesendet werden.
- linger.ms: Der Hersteller wartet die angegebene Verzögerungszeit ab, damit weitere Datensätze gesendet werden können. Daher können ein Batch aller gesendeten Datensätze auf einmal erstellt werden.
- buffer.memory: Die Gesamtzahl der Speicherbytes, die der Hersteller zur Zwischenspeicherung von Datensätzen verwenden kann, die darauf warten, an den Server gesendet zu werden.

Die folgende Tabelle enthält die für die Veröffentlichung von Nachrichten erforderlichen optionalen Parameter.

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
acks	1	0, 1, all
retries	3	Nicht negative Ganz-zahl
batch.size	16384	Positive Ganzzahl
linger.ms	0	Nicht negative Ganz-zahl

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
buffer.memory	33554432	Positive Ganzzahl

Optionale Parameter zum Abonnement von Nachrichten

enable.auto.commit bezeichnet die automatische Übertragung von Offsets mit einer Frequenz, die durch die Konfiguration "auto.commit.interval.ms" gesteuert wird. Der Wert von auto.commit.interval.ms darf 1000 nicht überschreiten, da das Abfrageintervall auf 1000 eingestellt ist. Der Wert von auto.commit.interval.ms darf den Wert des Abfrageintervalls nicht überschreiten.

Die folgende Tabelle enthält die optionalen Parameter zum Abonnieren von Nachrichten.

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
enable.auto.commit	true	True, False
auto.commit.interval-.ms	200	Positive Ganzzahl

Optionale Thread-Verwaltungsparameter

Die folgenden optionalen Parameter können für die Verwaltung von Threads verwendet werden.

- corePoolSize: Die Anzahl der Threads, die zur Überwachung des Kafka-Services im Pool gehalten werden sollen.
- maxPoolSize: Die maximale Anzahl von Threads, die zur Überwachung des Kafka-Services im Pool gehalten werden sollen.
- keepAliveTimeSecs: Die maximale Zeit, die die überschüssigen inaktiven Threads auf neue Tasks warten, bevor sie die Überwachung des Kafka-

Service beenden, wenn die Anzahl der Threads größer als die des Kerns ist.

- queueCapacity: Der Umfang der Queue, die vom Thread-Pool zur Überwachung des Kafka-Service verwendet wird.

Die folgende Tabelle enthält die optionalen Parameter für die Verwaltung von Threads.

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
corePoolSize	1	Positive Ganzzahl
maxPoolSize	5	Positive Ganzzahl
keepAliveTimeSecs	5	Positive Ganzzahl
queueCapacity	100	Positive Ganzzahl

Optionale Zookeeper-Parameter

Die folgenden optionalen Parameter können für Zookeeper-Aktivitäten verwendet werden.

zookeeper.connection.timeout.ms: Die maximale Wartezeit für den Client, bis eine Verbindung mit Zookeeper hergestellt ist. Wenn nicht festgelegt, wird der Wert in "zookeeper.session.timeout.ms" verwendet.

Die folgende Tabelle enthält die optionalen Parameter für die Zookeeper-Aktivitäten.

Parameter	Stan-dardwert	Zuläs-siger/Beispiel-parame-terwert
zookeeper.connection.timeout.ms	6000	Positive Ganzzahl

Optionale Parameter für die Erstellung von Themen

Die folgenden optionalen Parameter können für die Erstellung von Themen verwendet werden.

- num.partitions: Die Anzahl der Partitionen für das Thema Offset-Commit.
- replication.factor: Der Replizierungsfaktor zur Änderung von Protokoll- und Repartitionsthemen, die von der Stream-Verarbeitungsanwendung erstellt wurden.

Die folgende Tabelle enthält die optionalen Parameter für die Erstellung von Themen.

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
num.partitions	1	Positive Ganzzahl
replication.factor	1	Positive Ganzzahl

Interact | triggeredMessage | gateways | <gatewayName>

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für ein bestimmtes Gateway in ausgelösten Nachrichten.

Unica Interact bietet keine Unterstützung für mehrere Instanzen desselben Gateways. Auf alle Dateien für die Gateway-Konfiguration sollte über jeden Unica Interact-Laufzeitknoten zugegriffen werden können. Im Falle einer verteilten Konfiguration müssen Sie sicherstellen, dass die Gateway-Dateien an einer gemeinsam genutzten Position abgelegt werden.

Anmerkung: Unter diesem Knoten stehen die sofort einsatzbereiten Gateways mit den Namen „EMail“, „MobilePush“, „UBX“ sowie alle erforderlichen Parameter und ihre jeweiligen Werte zur Verfügung. Sie müssen keinen der Werte in der Plattformkonfiguration aktualisieren. In den Eigenschaftendateien, auf die in diesen Konfigurationen verwiesen wird, sind nur Änderungen erforderlich. Wenn Sie ein

Upgrade von einer früheren Version von Interact durchgeführt haben, funktioniert die vorhandene Konfiguration mit diesen Gateways weiterhin unverändert.

Kategoriename

Syntax

Diese Eigenschaft definiert den Namen dieses Gateways. Er muss für jedes Gateway eindeutig sein.

className

Syntax

Diese Eigenschaft definiert den vollständig qualifizierten Klassennamen der Implementierung dieses Gateways.

classPath

Syntax

Diese Eigenschaft definiert den URI der JAR-Datei, die die Implementierung dieses Gateways enthält. Wird diese Eigenschaft leer gelassen, wird der Klassenpfad der hostenden Interact-Anwendung verwendet.

Wenn z. B. in einem Windows-System die Gateway-JAR-Datei im Verzeichnis `C:\HCL\Unica\EmailGateway\IBM_Interact_OMO_OutboundGateway_Silverpop_1.0\lib\OMO_OutboundGateway_Silverpop.jar` verfügbar ist, sollte der Klassenpfad `file:///C:/HCL/Unica/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar` sein. Wenn in einem Unix-System die Gateway-JAR-Datei im Verzeichnis `/opt/HCL/Unica/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar` verfügbar ist, sollte der Klassenpfad `file:///opt/HCL/Unica/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar` sein.

Interact | triggeredMessage | gateways | <gatewayName> | Parameter Data

Die Konfigurationseigenschaften in dieser Kategorie definieren Parameter für ein bestimmtes Gateway in ausgelösten Nachrichten.

Kategoriename

Syntax

Diese Eigenschaft definiert den Namen dieses Parameters. Der Name muss für jeden Parameter für dieses Gateway eindeutig sein.

Wert

Syntax

Diese Eigenschaft definiert die Parameter, im Format von Name/Wert-Paaren, die von diesem Gateway benötigt werden. Die folgenden Parameter werden für alle Gateways unterstützt.

Anmerkung:

- Bei allen Parametern für Auslösenachrichten muss die Groß-/Kleinschreibung beachtet werden und sie sollten wie im Folgenden dargestellt eingegeben werden.
  - validationTimeoutMillis: Die Dauer in Millisekunden, in der das Zeitlimit der Validierung eines Angebots über dieses Gateway überschritten wird. Der Standardwert ist 500.
  - deliveryTimeoutMillis: Die Dauer in Millisekunden, in der das Zeitlimit der Validierung eines Angebots über dieses Gateway überschritten wird. Der Standardwert ist 1000.
- Um Gateway-bezogene Protokolle in der Datei Interact.log zu ermitteln, platzieren Sie das alte 'interact_log4j2.xml' aus Versionen vor 11.1 in 'InteractRT.war/WEB-INF/classes' und legen Sie es auch an einem beliebigen Speicherort außerhalb

der War-Datei ab. Sie müssen den folgenden JVM-Parameter auf dem Anwendungsserver angeben: -Dlog4j.configuration = file: /opt/any_location/interact_log4j.properties

Interact | triggeredMessage | channels

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für alle Kanäle in ausgelösten Nachrichten.

Typ

Syntax

Diese Eigenschaft definiert den Stammknoten für Einstellungen zu einem bestimmten Gateway. Bei "Default" wird die integrierte Kanalauswahl verwendet, die auf der Liste der Kanäle basiert, die in der Benutzeroberfläche der ausgelösten Nachrichten definiert wurden. Ist "Default" ausgewählt, sollten die Werte className und classPath leer gelassen werden. Bei "Custom" wird die benutzerdefinierte Implementierung von IChannelSelector verwendet.

Gültige Werte

Standard | Benutzerdefiniert

className

Syntax

Diese Eigenschaft definiert den vollständig qualifizierten Klassennamen der Kundenimplementierung der Kanalauswahl. Diese Einstellung ist bei dem Typ "Custom" erforderlich.

classPath

Syntax

Diese Eigenschaft definiert die URL der JAR-Datei, die die Implementierung der Kundenimplementierung der Kanalauswahl umfasst. Wird diese Eigenschaft

leer gelassen, wird der Klassenpfad der hostenden Interact-Anwendung verwendet.

Interact | triggeredMessage | Kanäle | Parameterdaten

Die Konfigurationseigenschaften in dieser Kategorie definieren Parameter für einen bestimmten Kanal in ausgelösten Nachrichten.

Kategoriename

Syntax

Diese Eigenschaft definiert den Namen dieses Parameters. Der Name muss für jeden Parameter für diesen Kanal eindeutig sein.

Wert

Syntax

Diese Eigenschaft definiert die Parameter, im Format von Name/Wert-Paaren, die für diese Kanalauswahl benötigt werden.

Wenn Sie **Vom Kunden bevorzugte Kanäle** für Ihren Kanal verwenden, müssen Sie Folgendes erstellen

Interact | triggeredMessage | Kanäle | <channelName>

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für einen bestimmten Kanal in ausgelösten Nachrichten.

Kategoriename

Syntax

Diese Eigenschaft definiert den Namen des Kanals, über den Angebote gesendet werden. Sie sollte mit den in der Designzeit unter **Campaign | Partitionen | <Partition[N]> | Interact | outboundChannels** definierten Namen übereinstimmen.

Interact | triggeredMessage | Kanäle | <channelName> | <handlerName>

Die Konfigurationseigenschaften in dieser Kategorie definieren Einstellungen für einen bestimmten Handler in ausgelösten Nachrichten, der zum Senden von Angeboten verwendet wird.

Kategoriename

Syntax

Diese Eigenschaft definiert den Namen des Handlers, der im Kanal zum Senden von Angeboten verwendet wird.

Dispatcher

Syntax

Diese Eigenschaft definiert den Namen des Dispatchers, über den dieser Handler Angebote an das Gateway sendet. Es muss eines der unter **interact | triggeredMessage | gateways** definierten Gateways sein.

Gateway

Syntax

Diese Eigenschaft definiert den Namen des Gateways, an das dieser Handler schließlich Angebote sendet. Es muss eines der unter **interact | triggeredMessage | gateways definierten Gateways** sein.

mode

Syntax

Diese Eigenschaft definiert den Verwendungsmodus dieses Handlers. Wenn "Failover" ausgewählt wird, wird dieser Handler nur verwendet, wenn keiner der Handler mit höheren Prioritäten, die in diesem Kanal definiert sind, ein Angebot senden konnte. Wenn "Addon" ausgewählt wird, wird dieser Handler unabhängig davon verwendet, ob andere Handler erfolgreich Angebote gesendet haben.

Priorität

Syntax

Diese Eigenschaft definiert die Priorität dieses Handlers. Die Engine versucht zunächst, den Handler mit der höchsten Priorität zum Senden der Angebote zu verwenden.

Gültige Werte

Beliebige Ganzzahl

Standard

100

Interact | activityOrchestrator

Die Kategorie "activityOrchestrator" gibt die Empfänger und Gateways für die eingehende Gateway-Aktivität von Unica Interact an.

Verwenden Sie die Konfigurationseigenschaften **Interact | activityOrchestrator | receivers**, um die Unica Interact-Empfänger zu konfigurieren. Verwenden Sie die Konfigurationseigenschaften **Interact | activityOrchestrator | gateways**, um die Gateways zu konfigurieren, die in Unica Interact verwendet werden sollen.

Interact | activityOrchestrator | receivers

Die Kategorie "activityOrchestrator | receivers" legt die Ereignisempfänger für die eingehende Gateway-Aktivität von Unica Interact fest.

Kategoriename

Beschreibung

Der Name Ihres Empfängers.

Typ

Beschreibung

Der Typ des Empfängers. Sie können zwischen Kafka und Custom wählen. Für Custom müssen Sie eine Implementierung von `iReceiver` verwenden.

Anmerkung: Wenn Sie Kafka in der Vorgängerversion verwendet haben, dann können Sie den Wert des Typs als Kafka in der Upgrade-Version einstellen.

Aktiviert

Beschreibung

Wählen Sie `True` aus, um den Empfänger zu aktivieren, oder wählen Sie `False` aus, um ihn zu inaktivieren.

className

Beschreibung

Diese Eigenschaft definiert den vollständig qualifizierten Klassennamen der Implementierung dieses Empfängers. Er wird nur verwendet, wenn der Typ `Custom` lautet. Für den Typ Kafka muss der Wert leer gelassen werden.

classPath

Beschreibung

Diese Eigenschaft definiert den URI der JAR-Datei, die die Implementierung dieses Empfängers enthält. Wird diese Eigenschaft leer gelassen, wird der Klassenpfad der hostenden Unica Interact-Anwendung verwendet. Er wird nur verwendet, wenn der Typ `Custom` lautet. Für den Typ Kafka muss der Wert leer gelassen werden.

Interact | activityOrchestrator | receivers | Parameter Data

Sie können Empfängerparameter (z.B. "queueManager" und "messageQueueName") hinzufügen, um Ihre Empfängerwarteschlange zu definieren.

Für den Typ Kafka wird der folgende Parameter unterstützt.

- providerUrl: Eine Liste von Host/Port-Paaren, die für die Herstellung der ersten Verbindung mit dem Kafka-Cluster verwendet werden sollen. Diese Liste muss das Format `host1:port1,host2:port2,...` haben.
- Thema: Ein Thema ist eine Kategorie oder ein Feed-Name, in dem Nachrichten gespeichert und veröffentlicht werden. Alle Nachrichten von Kafka sind nach Themen geordnet. Möchten Sie eine Nachricht senden, können Sie sie zu einem bestimmten Thema senden, und wenn Sie eine Nachricht lesen möchten, können Sie sie von einem bestimmten Thema aus lesen. Herstelleranwendungen geben Daten in Themen ein und Verbraucheranwendungen lesen aus Themen. Der Name des Themas muss ein alphanumerisches ASCII-Zeichen, '.', '_' und '-' enthalten. Aufgrund der Einschränkungen bei den Themennamen können Sie entweder Themen mit einem Punkt ('.') oder einem Unterstrich ('_') verwenden. Der Name eines Themas kann maximal 255 Zeichen lang sein. Wenn Sie beispielsweise einen Themennamen 'InteractTM_1' erstellen oder angeben und versuchen, ein Thema wie 'InteractTM.1' zu erstellen, wird der folgende Fehler generiert. "Topic InteractTM.1 kollidiert mit bestehenden Themen: InteractTM _1."
- group.id: Gibt den Namen der Kundengruppe an, zu der ein Kafka-Kunde gehört.
- zookeeper.connect: Gibt die Zookeeper-Verbindungszeichenfolge in Form von `hostname:port` an, wobei Hostname und Port als Host und Port eines Zookeeper-Servers definiert sind.
- Authentifizierung: Benutzer können Kafka verwenden, indem sie verschiedene Authentifizierungsmechanismen aktivieren.

Obligatorische Parameter für das Abonnement von Nachrichten

Standardmäßig unterstützt der Kafka-Server keinen Authentifizierungsmechanismus. Sie können den Kafka-Server starten, wenn der Authentifizierungsmechanismus deaktiviert ist. In diesem Fall können Sie den Parameter "Authentifizierung" auf den Wert "None" setzen. Die folgende

Tabelle enthält die obligatorischen Parameter, die zum Abonnieren von Nachrichten erforderlich sind.

Parameter	Zulässiger/Beispielparameterwert
providerUrl	<host>:<port> (Beispiel: local-host:9092)
group.id	Jede beliebige Zeichenfolge (Beispiel: InteractTMGateway)
topic	Jede beliebige Zeichenfolge (Beispiel: InteractTM)
Authentifizierung	Jede Zeichenfolge
zookeeper.connect	<host>:<port> (Beispiel: local-host:2181)

Authentifizierungsmechanismus

Sie können Kafka verwenden, indem Sie verschiedene Authentifizierungsmechanismen aktivieren.

Authentifizierung durch den SASL_PLAIN-Mechanismus

Wenn Sie den SASL_PLAIN-Authentifizierungsmechanismus verwenden möchten, müssen Sie den Parameter "authentication" auf den Wert "Plain" zusammen mit den unterstützten Parametern setzen.

Wenn der SASL_PLAIN-Mechanismus unterstützt wird, müssen die folgenden Parameter angegeben werden.

- asmUserforMQAuth: Der Benutzername für die Anmeldung auf dem Server. Dieses ist erforderlich, wenn der Server eine Authentifizierung erzwingt.
- authDS: Das Kennwort, das dem Benutzernamen für die Anmeldung auf dem Server zugeordnet ist.
- Benutzername/Kennwort: Der Benutzername oder das Kennwort des Kafka-Servers, der in der JASS-Konfigurationsdatei konfiguriert ist.

Die folgende Tabelle enthält die für den SASL_PLAIN-Mechanismus erforderlichen Parameter.

Parameter	Zulässige/Beispielparameterwerte
Authentifizierung	Normal
asmUserforMQAuth	Jede beliebige Zeichenfolge (Beispiel: test_user)
authDS	Jede beliebige Zeichenfolge (Beispiel: authDS)
Benutzername	Jede beliebige Zeichenfolge (Beispiel: test_user)
Kennwort	Jede beliebige Zeichenfolge (Beispiel: test-secret)

Wenn der "Authentifizierungs"-Parameter "Plain" lautet, müssen Sie entweder asmUserforMQAuth/authDS oder Benutzername/-Kennwortparameter für die Authentifizierung verwenden.

Erstellen Sie die Datenquellen (authDS) im Abschnitt "Benutzer" in der Plattformkonfiguration. Einzelheiten zu den Datenquellen sind im folgenden Beispiel aufgeführt.

Datenquelle	Benutzername	Kennwort
authDS	test_user	test-secret

Authentifizierung durch SSL-Mechanismus

Um den SSL-Authentifizierungsmechanismus zu verwenden, müssen Sie den Parameter 'authentication' zusammen mit den unterstützten Parametern auf den Wert 'SSL' setzen.

Die folgenden Parameter sind zur Unterstützung des SSL-Mechanismus erforderlich.

- ssl.keystore.location: Die Adresse der Schlüsselspeicherdatei. Sie können sie für eine Zweiwege-Authentifizierung für den Client verwenden.
- ssl.truststore.location: Die Adresse der Truststore-Datei.
- SSLKeystoreDS: Der Name der Schlüsselspeicherdatenquelle, in der das Kennwort von SSL-Schlüsselspeicher gespeichert ist.
- SSLKeyDS: Der Name der Schlüsseldatenquelle, die das Paßwort des SSL-Schlüssels speichert.
- SSLTruststoreDS: Der Truststore-Datenquellenname, der das Kennwort von SSL-Truststore speichert.

Die folgende Tabelle enthält die unterstützten Parameter für den SSL-Mechanismus.

Parameter	Zulässige/Beispielparameterwerte
Authentifizierung	SSL
ssl.keystore.location	Adresse des SSL-Schlüsselspeichers (Beispiel: `C:/SSL/kafka.client.keystore.jks`)

Parameter	Zulässige/Beispielparameterwerte
ssl.truststore.location	Adresse des SSL-Schlüsselspeichers (Beispiel: `C:/SSL/kafka.client.truststore.jks`)
asmUserforMQAuth	Jede beliebige Zeichenfolge (Beispiel: test_user)
SSLKeystoreDS	Jede beliebige Zeichenfolge (Beispiel: SSLKeystoreDS)
SSLKeyDS	Jede beliebige Zeichenfolge (Beispiel: SSLKeyDS)
SSLTruststoreDS	Jede beliebige Zeichenfolge (Beispiel: SSLTruststoreDS)

Erstellen Sie die Datenquellen (SSLKeystoreDS, SSLKeyDS und SSLTruststoreDS) im Abschnitt Benutzer in der Plattformkonfiguration. Einzelheiten zu den Datenquellen sind im folgenden Beispiel aufgeführt.

Datenquelle	Benutzername	Kennwort
SSLKeystoreDS	Keystore	keystore-secret
SSLKeyDS	Schlüssel	key-secret
SSLTruststoreDS	Truststore	truststore -secret

Anmerkung: Der Client Keystore oder Truststore wird auf Produzenten- oder Konsumentenseite in der Interact-Anwendung benötigt (wo die Interact-Anwendung installiert ist). `C:/SSL/kafka.client.keystore.jks` und `C:/SSL/kafka.client.truststore.jks` sind die lokalen Speicherorte, an denen die Interact-Anwendung installiert ist.

Authentifizierung durch den SASL_SSL-Mechanismus

Wenn Sie den SASL_SSL-Authentifizierungsmechanismus verwenden möchten, müssen Sie den Parameter "authentication" zusammen mit den unterstützten Parametern auf den Wert "SASL_SSL" setzen. Der SASL_SSL-Mechanismus besteht aus einer Kombination von SASL_PLAIN- und SSL-Mechanismen. Die folgende Tabelle enthält die unterstützten Parameter für den SASL_SSL-Mechanismus.

Parameter	Zulässige/Beispielparameterwerte
Authentifizierung	SASL_SSL
asmUserforMQAuth	Jede beliebige Zeichenfolge (Beispiel: test_user)
authDS	Jede beliebige Zeichenfolge (Beispiel: authDS)
Benutzername	Jede beliebige Zeichenfolge (Beispiel: test_user)
Kennwort	Jede beliebige Zeichenfolge (Beispiel: test-secret)
ssl.keystore.location	Adresse des SSL-Schlüsselspeichers (Beispiel: `C:/SSL/kafka.client.keystore.jks`)
ssl.truststore.location	Adresse des SSL-Schlüsselspeichers (Beispiel: `C:/SSL/kafka.client.truststore.jks`)
SSLKeystoreDS	Jede beliebige Zeichenfolge (Beispiel: SSLKeystoreDS)
SSLKeyDS	Jede beliebige Zeichenfolge (Beispiel: SSLKeyDS)

Parameter	Zulässige/Beispielparameterwerte
SSLTruststoreDS	Jede beliebige Zeichenfolge (Beispiel: SSLTruststoreDS)

Wenn der "Authentifizierungs"-Parameter "SASL_SSL" lautet, müssen Sie entweder asmUserforMQAuth/authDS oder Benutzername/Kennwort verwenden.

Erstellen Sie die Datenquellen (authDS, SSLKeystoreDS, SSLKeyDS und SSLTruststoreDS) im Abschnitt "Benutzer" in der Plattformkonfiguration. Einzelheiten zu den Datenquellen sind im folgenden Beispiel aufgeführt.

Datenquelle	Benutzername	Kennwort
authDS	Administrator	admin-secret
SSLKeystoreDS	Keystore	test1234
SSLKeyDS	Schlüssel	test1234
SSLTruststoreDS	Truststore	test1234

Anmerkung: Wenn Sie Datenquellen wie authDS, SSLKeystoreDS, SSLKeyDS oder SSLTruststoreDS im Konfigurationsparameter der Plattform angeben, müssen Sie auch den Parameter asmUserforMQAuth angeben.

Der Client Keystore/Truststore wird auf Produzenten- oder Konsumentenseite in der Interact-Anwendung benötigt (wo die Interact-Anwendung installiert ist). `C:/SSL/kafka.client.keystore.jks` und `C:/SSL/kafka.client.truststore.jks` sind die lokalen Speicherorte, an denen die Interact-Anwendung installiert ist.

Optionale Parameter zum Abonnement von Nachrichten

enable.auto.commit bezeichnet die automatische Übertragung von Offsets mit einer Frequenz, die durch die Konfiguration "auto.commit.interval.ms" gesteuert wird. Der Wert von auto.commit.interval.ms darf 1000 nicht überschreiten, da das Abfrageintervall auf 1000 eingestellt ist. Der Wert von auto.commit.interval.ms darf den Wert des Abfrageintervalls nicht überschreiten.

Die folgende Tabelle enthält die optionalen Parameter zum Abonnieren von Nachrichten.

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
enable.auto.commit	true	True, False
auto.commit.interval-.ms	200	Positive Ganzzahl

Optionale Thread-Verwaltungsparameter

Die folgenden optionalen Parameter können für die Verwaltung von Threads verwendet werden.

- corePoolSize: Die Anzahl der Threads, die zur Überwachung des Kafka-Services im Pool verbleiben müssen.
- maxPoolSize: Die maximale Anzahl von Threads, die zur Überwachung des Kafka-Service im Pool verbleiben müssen.
- keepAliveTimeSecs: Die maximale Zeit, die die überschüssigen inaktiven Threads benötigen, um auf neue Tasks zu warten, bevor sie die Überwachung des Kafka-Service abbrechen, wenn die Anzahl der Threads größer ist als der Kern.
- queueCapacity Der Umfang der Warteschlange, die vom Thread-Pool zur Überwachung des Kafka-Service verwendet wird.

Die folgende Tabelle enthält die optionalen Parameter für die Verwaltung von Threads.

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
corePoolSize	1	Positive Ganzzahl
maxPoolSize	5	Positive Ganzzahl
keepAliveTimeSecs	5	Positive Ganzzahl
pqueueCapacity	100	Positive Ganzzahl

Optionale Zookeeper-Parameter

Der folgende optionale Parameter kann für Zookeeper-Aktivitäten verwendet werden.

zookeeper.connection.timeout.ms: Die maximale Wartezeit für den Client, bis eine Verbindung mit Zookeeper hergestellt ist. Wenn nicht festgelegt, wird der Wert in "zookeeper.session.timeout.ms" verwendet.

Die folgende Tabelle enthält die optionalen Parameter für die Zookeeper-Aktivitäten.

Parameter	Standardwert	Zulässiger/Beispielparameterwert
zookeeper.connection.timeout.ms	6000	Positive Ganzzahl

Optionale Parameter für die Erstellung von Themen

Die folgenden optionalen Parameter können für die Erstellung von Themen verwendet werden.

- num.partitions: Die Anzahl der Partitionen für das Thema Offset-Commit.
- replication.factor: Der Replizierungsfaktor zum Ändern von Protokoll- und Repartitionsthemen, die von der Stream-Verarbeitungsanwendung erstellt wurden.

Die folgende Tabelle enthält die optionalen Parameter für die Erstellung von Themen.

Parameter	Standardwert	Zulässige/Beispiel-parameterwerte
num.partitions	1	Positive Ganzzahl
replication.factor	1	Positive Ganzzahl

Interact | activityOrchestrator | gateways

Die Kategorie "activityOrchestrator | gateways" gibt die Gateways für die eingehende Gateway-Aktivität von Unica Interact an.

Anmerkung: Unter diesem Knoten stehen die sofort einsatzbereiten Gateways mit den „UBX“ sowie alle erforderlichen Parameter und ihre jeweiligen Werte zur Verfügung. Sie müssen keinen der Werte in der Plattformkonfiguration aktualisieren. In den Eigenschaftendateien, auf die in diesen Konfigurationen verwiesen wird, sind nur Änderungen erforderlich. Wenn Sie ein Upgrade von einer früheren Version von Interact durchgeführt haben, funktioniert die vorhandene Konfiguration mit diesen Gateways weiterhin unverändert.

Kategoriename

Beschreibung

Der Name Ihres Gateways.

className

Beschreibung

Diese Eigenschaft definiert den vollständig qualifizierten Klassennamen der Implementierung dieses Gateways.

classPath

Beschreibung

Diese Eigenschaft definiert den URI der JAR-Datei, die die Implementierung dieses Gateways umfasst. Wird diese Eigenschaft leer gelassen, wird der Klassenpfad der hostenden Unica Interact-Anwendung verwendet. Er wird nur verwendet, wenn der Typ `Custom` lautet.

Interact | activityOrchestrator | gateways | Parameter Data

Sie können Gateway-Parameter für Ihre Gateway-Konfigurationsdateien hinzufügen, beispielsweise `OMO-conf_inbound_UBX_interactEventNameMapping` und `OMO-conf_inbound_UBX_interactEventPayloadMapping`.

Konfigurationsbaumpfade

```
Interact|activityOrchestrator|gateways|<gatewayName>|Parameter Data|<partitionName>|processTimeoutMillis|value=XXX

Interact|activityOrchestrator|gateways|<gatewayName>|Parameter Data|<partitionName>|OMO-processTimeoutMillis|value=XXX
```

Interact | ETL | patternStateETL

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den ETL-Prozess.

Neuer Kategoriename

Syntax

Geben Sie einen Namen an, der diese Konfiguration eindeutig identifiziert. Beachten Sie, dass Sie bei der Ausführung des eigenständigen ETL-Prozesses genau diesen Namen angeben müssen. Aus praktischen Gründen wird empfohlen, keinen Namen mit Leerzeichen oder Interpunktionen zu wählen, da dieser Name in der Befehlszeile angegeben werden muss. Beispiel: `ETLProfile1`.

runOnceADay

Syntax

Legt fest, ob der eigenständige ETL-Prozess in dieser Konfiguration einmal täglich ausgeführt werden soll. Gültige Antworten sind **Yes** oder **No**. Wenn Sie hier **Nein** antworten, bestimmt der **processSleepIntervalInMinutes** den Ausführungsplan für den Prozess.

preferredStartTime

Syntax

Die bevorzugte Uhrzeit, zu der der eigenständige ETL-Prozess starten soll. Geben Sie die Uhrzeit im Format HH:MM:SS AM/PM an. Beispiel: `01:00:00 AM`.

preferredEndTime

Syntax

Die bevorzugte Uhrzeit, zu der der eigenständige ETL-Prozess stoppen soll. Geben Sie die Uhrzeit im Format HH:MM:SS AM/PM an. Beispiel: `08:00:00 AM`.

processSleepIntervalInMinutes

Syntax

Wenn Sie in der Konfiguration des eigenständigen ETL-Prozesses nicht festgelegt haben, dass dieser einmal täglich ausgeführt werden soll (Angabe der Eigenschaft **runOnceADay**), gibt diese Eigenschaft das Intervall

zwischen den Ausführungen des ETL-Prozesses an. Wenn Sie an dieser Stelle beispielsweise `15` angeben, wartet der eigenständige ETL-Prozess 15 Minuten nach der Beendigung seiner Ausführung, bis der Prozess wieder gestartet wird.

maxJDBCInsertBatchSize

Syntax

Die maximale Anzahl der Datensätze eines JDBC-Batches vor dem Ausführen der Abfrage. Standardmäßig ist diese Eigenschaft auf 5000 gesetzt. Beachten Sie, dass dies nicht die maximale Anzahl der Datensätze ist, die vom ETL-Prozess in einer Iteration verarbeitet werden. Während jeder Iteration verarbeitet der ETL-Prozess alle verfügbaren Datensätze aus der Tabelle UACI_EVENTPATTERNSTATE. Diese Datensätze werden jedoch in **maxJDBCInsertSize**-Datenblöcke unterteilt.

maxJDBCFetchBatchSize

Syntax

Die maximale Anzahl der Datensätze eines von der Staging-Datenbank abzurufenden JDBC-Batches.

Sie müssen diesen Wert möglicherweise erhöhen, um die Leistung des ETL-Prozesses zu optimieren.

communicationPort

Syntax

Der Netzport, an dem der eigenständige ETL-Prozess für eine Stoppanforderung empfangsbereit ist. Unter normalen Umständen muss dieser Standardwert nicht geändert werden.

queueLength

Syntax

Ein Wert, der für die Leistungsoptimierung verwendet wird. Die Erfassungen der Musterzustandsdaten werden abgerufen und in Objekte transformiert, die einer Warteschlange hinzugefügt werden, damit sie verarbeitet und in die Datenbank geschrieben werden können. Diese Eigenschaft steuert die Größe der Warteschlange.

completionNotificationScript

Syntax

Gibt den absoluten Pfad zu einem Script an, das ausgeführt wird, wenn der ETL-Prozess abgeschlossen ist. Wenn Sie ein Skript angeben, werden drei Argumente an das Fertigstellungsbenachrichtigungsskript übergeben: Startzeit, Endzeit und Gesamtzahl der verarbeiteten Ereignismusterdatensätze. Die Start- und Endzeit sind numerische Werte, die die Anzahl der seit 1970 vergangenen Millisekunden darstellen.

Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für die Laufzeitdatenquelle des ETL-Prozesses.

Typ

Beschreibung

Eine Liste der von Ihnen definierten unterstützten Datenbanktypen für die Datenquelle.

dsname

Beschreibung

Der JNDI-Name der Datenquelle. Dieser Name muss auch in der Datenquellenkonfiguration des Benutzers verwendet werden, um sicherzustellen, dass der Benutzer auf die Ziel- und Laufzeitdatenquellen zugreifen kann.

driver

Beschreibung

Der Name des zu verwendenden JDBC-Treibers. Beispiele:

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

Beschreibung

Die Datenquellen-URL, z.B. eine der folgenden:

Oracle: `jdbc:oracle:thin:@ <your_db_host>:<your_db_port>:<your_db_service_name>`

Microsoft SQL Server: `jdbc:sqlserver:// <your_db_host>:<your_db_port> ;databaseName= <your_db_name>`

IBM DB2: `jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>`

MariaDB: `jdbc:mariadb:// <your_db_host>:<your_db_port>/ <your_db_name>`

connectionpoolSize

Beschreibung

Ein Wert, der die Größe des Verbindungspools angibt. Er wird zur Leistungsoptimierung bereitgestellt. Die Musterzustandsdaten werden gelesen und gleichzeitig je nach den verfügbaren Datenbankverbindungen transformiert. Durch eine Erhöhung der Verbindungspoolgröße können mehr gleichzeitige Datenbankverbindungen verwendet werden. Dies hängt jedoch auch von Einschränkungen des Speichers und der Lese-/Schreibfunktionalität für Datenbanken ab. Wenn Sie diesen Wert beispielsweise auf 4 setzen, werden vier Jobs gleichzeitig ausgeführt. Falls Sie ein umfangreiches

Datenvolumen haben, müssen Sie diesen Wert unter Umständen auf eine Zahl wie 10 oder 20 erhöhen, sofern der verfügbare Speicher und die Datenbankleistung ausreichen.

Schema

Beschreibung

Der Name des Datenbankschemas, mit dem sich diese Konfiguration verbindet.

connectionRetryPeriod

Beschreibung

Die Eigenschaft `ConnectionRetryPeriod` gibt die Zeit in Sekunden an, in der Unica Interact versucht, die Datenbankverbindungsanforderung bei einem Ausfall automatisch erneut zu stellen. Unica Interact versucht automatisch, die Verbindung zur Datenbank für diese Zeitspanne erneut herzustellen, bevor ein Datenbankfehler oder -ausfall gemeldet wird. Wird der Wert auf 0 gesetzt, wiederholt Unica Interact den Verbindungsversuch unbegrenzt. Wenn -1 festgelegt ist, wird kein wiederholter Versuch unternommen.

connectionRetryDelay

Beschreibung

Die Eigenschaft `ConnectionRetryDelay` gibt in Sekunden an, wie lange Unica Interact wartet, bevor es versucht, die Verbindung mit der Datenbank wiederherzustellen, nachdem ein Fehler aufgetreten ist. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für die Zieldatenquelle des ETL-Prozesses.

Typ

Beschreibung

Eine Liste der von Ihnen definierten unterstützten Datenbanktypen für die Datenquelle.

dsname

Beschreibung

Der JNDI-Name der Datenquelle. Dieser Name muss auch in der Datenquellenkonfiguration des Benutzers verwendet werden, um sicherzustellen, dass der Benutzer auf die Ziel- und Laufzeitdatenquellen zugreifen kann.

driver

Beschreibung

Der Name des zu verwendenden JDBC-Treibers. Beispiele:

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

Beschreibung

Die Datenquellen-URL, z.B. eine der folgenden:

Oracle: `jdbc:oracle:thin:@ <your_db_host>:<your_db_port>:<your_db_service_name>`

Microsoft SQL Server: `jdbc:sqlserver:// <your_db_host>:<your_db_port> ;databaseName= <your_db_name>`

IBM DB2: `jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>`

MariaDB: `jdbc:mariadb:// <your_db_host>:<your_db_port>/ <your_db_name>`

connectionpoolSize

Beschreibung

Ein Wert, der die Größe des Verbindungspools angibt. Er wird zur Leistungsoptimierung bereitgestellt. Die Musterzustandsdaten werden gelesen und gleichzeitig je nach den verfügbaren Datenbankverbindungen transformiert. Durch eine Erhöhung der Verbindungspoolgröße können mehr gleichzeitige Datenbankverbindungen verwendet werden. Dies hängt jedoch auch von Einschränkungen des Speichers und der Lese-/Schreibfunktionalität für Datenbanken ab. Wenn Sie diesen Wert beispielsweise auf 4 setzen, werden vier Jobs gleichzeitig ausgeführt. Falls Sie ein umfangreiches Datenvolumen haben, müssen Sie diesen Wert unter Umständen auf eine Zahl wie 10 oder 20 erhöhen, sofern der verfügbare Speicher und die Datenbankleistung ausreichen.

Schema

Beschreibung

Der Name des Datenbankschemas, mit dem sich diese Konfiguration verbindet.

connectionRetryPeriod

Beschreibung

Die Eigenschaft `ConnectionRetryPeriod` gibt die Zeit in Sekunden an, in der Unica Interact versucht, die Datenbankverbindungsanforderung bei einem Ausfall automatisch erneut zu stellen. Unica Interact versucht automatisch, die Verbindung zur Datenbank für diese Zeitspanne erneut herzustellen, bevor ein Datenbankfehler oder -ausfall gemeldet wird. Wird der Wert auf 0 gesetzt, wiederholt Unica Interact den Verbindungsversuch unbegrenzt. Wenn -1 festgelegt ist, wird kein wiederholter Versuch unternommen.

connectionRetryDelay

Beschreibung

Die Eigenschaft `ConnectionRetryDelay` gibt in Sekunden an, wie lange Unica Interact wartet, bevor es versucht, die Verbindung mit der Datenbank wiederherzustellen, nachdem ein Fehler aufgetreten ist. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Interact | ETL | patternStateETL | <patternStateETLName> | Bericht

Die Konfigurationseigenschaften in dieser Kategorie definieren die Einstellungen für den Aggregationsprozessbericht des ETL-Prozesses.

aktivieren

Syntax

Sie können die Berichtsintegration in den ETL-Prozess aktivieren oder inaktivieren. Diese Eigenschaft ist standardmäßig inaktiviert.

Wenn sie auf `disable` (Inaktivieren) gesetzt ist, inaktiviert diese Eigenschaft Aktualisierungen in der Tabelle UARI_DELTA_PATTERNS. Sie inaktiviert jedoch nicht die gesamte Berichterstellung.

Anmerkung: Wenn Sie die Berichtsintegration in den ETL-Prozess inaktivieren möchten, müssen Sie auch den Auslöser TR_AGGREGATE_DELTA_PATTERNS ändern, um die UACI_ETLPATTERNSTATERUN-Staging-Tabelle zu inaktivieren.

retryAttemptsIfAggregationRunning

Syntax

Gibt an, wie oft der ETL-Prozess versucht, zu prüfen, ob die Berichtsaggregation abgeschlossen ist, wenn das Flag "lock" festgelegt ist. Diese Eigenschaft ist standardmäßig auf 3 gesetzt.

sleepBeforeRetryDurationInMinutes

Syntax

Die Ruhezeit in Minuten zwischen aufeinanderfolgenden Versuchen. Diese Eigenschaft ist standardmäßig auf 5 Minuten gesetzt.

aggregationRunningCheckSql

Syntax

Mit dieser Eigenschaft können Sie angepasstes SQL definieren, mit dessen Ausführung angezeigt werden kann, ob das Flag "lock" für die Berichtsaggregation festgelegt ist. Diese Eigenschaft ist standardmäßig leer. Wenn diese Eigenschaft nicht festgelegt ist, führt der ETL-Prozess das folgende SQL aus, um das Flag "lock" abzurufen.

```
select count(1) AS ACTIVERUNS from uari_pattern_lock where isloc
k='Y'
=> If ACTIVERUNS is > 0, lock is set
```

aggregationRunningCheck

Syntax

Aktivieren oder inaktivieren Sie die Prüfung, ob die Berichtsaggregation aktiv ist, bevor die Ausführung des ETL-Prozesses erfolgt. Diese Eigenschaft ist standardmäßig aktiviert.

Kapitel 15. Unica Interact Simulator

In diesem Abschnitt werden alle Konfigurationseigenschaften für den Unica Interact-Simulator beschrieben.

Es wird nicht empfohlen, andere simulationsbezogene Aktionen (z. B. Hinzufügen, Kopieren, Löschen und Ändern) durchzuführen, wenn ein Simulationslauf in Bearbeitung ist.

Interact | Simulator

Die Konfigurationskategorie definiert die Parameter, die zur Ausführung des Abdeckungsanalyseszenarios des Simulatormoduls definiert werden müssen.

numberOfThreads

Syntax

Die Anzahl der zur Ausführung der Simulation verwendeten Threads

Standardwert

1

maxOffersToInclude

Syntax

Die maximale Anzahl der Angebote, die in jedem getOffers-Aufruf für jede Zielgruppen-ID im Szenario der Abdeckungsanalyse zurückgegeben werden.

Standardwert

10

insertBatchSize

Syntax

Definieren Sie die Größe jedes Stapels, um die resultierenden Datensätze persistent zu speichern.

Standardwert

1000

Interact | simulator|scenarioDataSource

Diese Konfigurationen sind erforderlich, um das Szenario der Simulatorabdeckungsanalyse auszuführen

jndiName

Beschreibung

Verwenden Sie diese jndiName-Eigenschaft, um die JNDI-Datenquelle (Java Naming and Directory Interface) zu identifizieren, die auf dem Anwendungsserver (Websphere oder WebLogic) für die Interact Design Time-Tabellen definiert ist.

Standardwert

Es ist kein Standardwert definiert.

Schema

Beschreibung

Der Name des Schemas, das die Tabellen für das Datenquellenmodul von Interact Design Time enthält. Ineract fügt den Wert für diese Eigenschaft vor allen Tabellennamen ein. Beispielsweise wird UACI_IntChannel zu schema.UACI_IntChannel.

Sie müssen ein Schema definieren. Wenn Sie kein Schema angeben, geht Interact davon aus, dass der Eigner der Tabellen mit dem Schema übereinstimmt. Es ist erforderlich, den Namen des Schemas anzugeben, um das Deckungsszenario erfolgreich auszuführen.

Standardwert

Es ist kein Standardwert definiert.

Typ

Beschreibung

Der Datenbanktyp für die Datenquelle, die von den Interact Design-Zeittabellen verwendet wird, auf die der Interact Simulator zugreift.

Standardwert

sql-server

Gültiger Wert

`sqlserver | Db2 | Oracle| MariaDB`

connectionRetryPeriod

Beschreibung

Die Eigenschaft ConnectionRetryPeriod gibt in Sekunden an, wie lange Interact eine fehlgeschlagene Datenbankverbindungsanforderung für die Lerntabellen automatisch wiederholt. Interact versucht in diesem Zeitraum automatisch, die Verbindung zur Datenbank wiederherzustellen, bevor ein Datenbankfehler gemeldet wird. Wird der Wert auf 0 gesetzt, versucht Interact unbegrenzt, die Verbindung wiederherzustellen. Wenn -1 festgelegt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

connectionRetryDelay

Beschreibung

Die ConnectionRetryDelay-Eigenschaft gibt die Zeit in Sekunden an, die Interact wartet, bevor es versucht, nach einem Fehler bei den Lerntabellen wieder eine Verbindung zur Datenbank herzustellen. Wenn der Wert auf -1 gesetzt ist, wird nicht versucht, die Verbindung wiederherzustellen.

Standardwert

-1

Fehlerbehandlung beim Simulator

Dieser Abschnitt listet die Statuscodes auf, die die Anwendung in die Tabelle UACI_SimulationHistory schreibt, die in der Interact Design-Zeitdatenbank vorhanden ist.

Im Falle eines Fehlers zeigt die Anwendung die Meldung "Szenario fehlgeschlagen" auf der Seite "Simulator ausführen" an. Der detaillierte Statuscode ist in der Datenbanktabelle UACI_SimulationHistory zu finden.

Im Folgenden finden Sie eine Liste der möglichen Statuscodes, die ein Verlauf einer Szenarioausführung haben könnte: // Der Statuscode 0-99 dient zu Informationszwecken

Status	**Code**	**Dringlichkeitsstufe**	**HTTP-Status**	**Mögliche UI-Nachricht**
ERFOLG	0	*INFO*	*OK*	Simulation erfolgreich ausgeführt
RUNNING	1	*INFO*	*OK*	Wird ausgeführt
ABBRECHEN	2	*INFO*	*OK*	Abbrechen
CANCELED	3	*INFO*	*OK*	Abgebrochen
EXPORTING_TO_CSV	4	*INFO*	*OK*	Exportieren in CSV
EXPORTED_TO_CSV	5	*INFO*	*OK*	In CSV exportiert

// Statuscode 101-999 stehen für Fehler

Status	**Code**	**Dringlichkeitsstufe**	**HTTP-Status**	**Mögliche UI-Nachricht**

NOT_ENAB-LED	101	*WARN*	SERVICE_-UNA-VAILABLE	Die Simulation ist auf diesem Lauf-zeitserver nicht aktiviert.
ERROR_RE-TRIEVE_SCEN-ARIO	102	ERROR	INTERNAL_-SERVER_ER-ROR	Fehler beim Abrufen der Szenarioin-formationen für die Simulation
INVALID_S-CENARIO	103	ERROR	BAD_RE-QUEST	Ungültige Szenarioinformationen der Simulation
ERROR_CREA-TE_RESULT_-TABLE	104	ERROR	INTERNAL_-SERVER_ER-ROR	Fehler bei der Erstellung der Tabelle zur Speicherung der Simulationser-gebnisse
ERROR_RE-TRIEVE_AU-DIENCE	105	ERROR	INTERNAL_-SERVER_ER-ROR	Fehler beim Abrufen von Zielgrup-pen-IDs für die Simulation
ERROR_CON-NECT_DATA-BASE	106	ERROR	INTERNAL_-SERVER_ER-ROR	Fehler bei der Verbindung zur {0}-Da-tenbank für die Simulation
ERROR_-PERSIST_-RESULT	107	ERROR	INTERNAL_-SERVER_ER-ROR	Fehler bei der Speicherung von Ergeb-nissen in der Datenbank für die Simu-lation
SCENARIO_-NOT_FOUND	108	ERROR	NOT_-FOUND	Kein ausführbares Szenario gefunden
GENERIC_ER-ROR	109	ERROR	INTERNAL_-SERVER_ER-ROR	Server-Fehler bei der Simulationsaus-führung
ERROR_UP-DATE_RESULT	110	ERROR	INTERNAL_-SERVER_ER-ROR	Fehler bei der Aktualisierung des Er-gebnisses für die Simulation

ERROR_IN-VALID_IC	111	ERROR	BAD_RE-QUEST	Interaktiver Kanal wird nicht bereitge-stellt

// // Statuscode 1001 und höher gelten nur für UI, werden nicht in der Datenbank gespeichert

Status	**Code**	**Dringlich-keitsstu-fe**	**HTTP-Sta-tus**	**Mögliche UI-Nachricht**
SIMULATION_AL-READY_RUNNING	1001	WARN	PRECON-DITION_FAI-LED	Für dieses Szenario wird bereits eine Simulation ausgeführt.
SIMULATION_NO-T_FOUND	1002	WARN	NO_CON-TENT	Keine laufende Simulation für dieses Szenario gefunden
SIMULATION_-RUNNING	1003	INFO	OK	Aktiv
SIMULATION_NO-T_RUNNING	1004	INFO	OK	Simulation nicht ausgeführt

Kapitel 16. Unica Interact Designumgebung - Konfigurationseigenschaften

In diesem Abschnitt werden alle Konfigurationseigenschaften für die Unica Interact-Designumgebung beschrieben.

Campaign | Partitionen | Partition[n] | Berichte

Die Eigenschaft **Campaign | Partitionen | Partition[n] | Berichte** definiert die unterschiedlichen Typen von Ordnern für Berichte.

offerAnalysisTabCachedFolder

Beschreibung

Die Eigenschaft `offerAnalysisTabCachedFolder` gibt die Position des Ordners an, der die Informationen für Bursting-Angebotsberichte (erweiterte Angebotsberichte) enthält, die auf der Registerkarte "Analyse" aufgeführt sind, wenn Sie den Ordner durch Klicken auf den Link "Analyse" im Navigationsbereich öffnen. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

Beschreibung

Die Eigenschaft `segmentAnalysisTabOnDemandFolder` gibt die Position des Ordners an, der die Segmentberichte enthält, die auf der Registerkarte "Analyse" eines Segments aufgeführt sind. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

Beschreibung

Die Eigenschaft `offerAnalysisTabOnDemandFolder` gibt die Position des Ordners an, der die Angebotsberichte enthält, die auf der Registerkarte "Analyse" eines Angebots aufgeführt sind. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

segmentAnalysisTabCachedFolder

Beschreibung

Die Eigenschaft `segmentAnalysisTabCachedFolder` gibt die Position des Ordners an, der die Informationen für Bursting-Segmentberichte (erweiterte Segmentberichte) enthält, die auf der Registerkarte "Analyse" aufgeführt sind, wenn Sie den Ordner durch Klicken auf den Link "Analyse" im Navigationsbereich öffnen. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']
```

analysisSectionFolder

Beschreibung

Die Eigenschaft `analysisSectionFolder` gibt die Position des Stammordners an, in dem Berichtinformationen gespeichert werden. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign']
```

campaignAnalysisTabOnDemandFolder

Beschreibung

Die Eigenschaft `campaignAnalysisTabOnDemandFolder` gibt die Position des Ordners an, der die Kampagnenberichte enthält, die auf der Registerkarte Analyse einer Kampagne aufgeführt sind. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']
```

campaignAnalysisTabCachedFolder

Beschreibung

Die Eigenschaft `campaignAnalysisTabCachedFolder` gibt die Position des Ordners an, der die Informationen für Bursting-Kampagnenberichte (erweiterte Kampagnenberichte) enthält, die auf der Registerkarte Analyse aufgeführt sind, wenn Sie den Ordner durch Klicken auf den Link Analyse im Navigationsbereich öffnen. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']/folder[@name='cached']
```

campaignAnalysisTabDeliverOnDemandFolder

Beschreibung

Die Eigenschaft `campaignAnalysisTabDeliverOnDemandFolder` gibt die Position des Ordners an, der die Unica Deliver-Berichte enthält, die auf der

Registerkarte „Analyse“ einer Kampagne aufgeführt sind. Der Pfad wird mithilfe der XPath-Schreibweise angegeben.

Standardwert

```
/content/folder[@name='Affinium Campaign']/folder[@name='Deliver Reports']
```

campaignAnalysisTabInteractOnDemandFolder

Beschreibung

Zeichenfolge für den Berichtsserverordner für Unica Interact-Berichte.

Standardwert

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert wird.

interactiveChannelAnalysisTabOnDemandFolder

Beschreibung

Zeichenfolge für Berichtsserverordner für Berichte über die Registerkarte zur Analyse des interaktiven Kanals.

Standardwert

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert wird.

Campaign | Partitionen | Partition[n] | UnicaInsightsReports

Die Eigenschaft Campaign | Partitionen | Partition[n] | UnicaInsightsReports definiert die unterschiedlichen Typen von Ordnern für Berichte.

interactAnalysisSectionFolder

Beschreibung

Zeichenfolge für den Berichtsserverordner für Unica Interact-Berichte.

Standardwert

```
/folder[@name='Unica Interact']
```

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Sie Unica Interact installieren und Unica Insights aktivieren.

interactiveChannelAnalysisTabOnDemandFolder

Beschreibung

Zeichenfolge für den Berichtsserverordner für Unica Interact-Berichte.

Standardwert

```
folder[@name='Unica Interact - Object Specific Reports']
```

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Sie Unica Interact installieren und Unica Insights aktivieren.

Campaign | Partionen | partition[n] | Interact | contactAndResponseHistTracking

Diese Konfigurationseigenschaften definieren die Einstellungen für das Unica Interact-Modul für Kontakt- und Antwortverlauf.

isEnabled

Beschreibung

Wenn der Wert auf `yes` steht, wird das Unica Interact-Modul für Kontakt- und Antwortverlauf aktiviert, das die Unica Interact-Kontakt- und Antwortverlaufsdaten aus den Staging-Tabellen in der Unica Interact-Laufzeit in die Unica Campaign-Kontakt- und Antwortverlaufstabellen kopiert. Die Eigenschaft `interactInstalled` muss ebenfalls auf `yes` gesetzt werden.

Standardwert

Nein

Gültige Werte

yes | no

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

runOnceADay

Beschreibung

Gibt an, dass der Kontakt- und Antwortverlauf-ETL-Prozess einmal pro Tag ausgeführt wird. Wenn Sie diese Eigenschaft auf `Yes` festlegen, wird der ETL-Prozess während des geplanten Intervalls ausgeführt, der durch `preferredStartTime` und `preferredEndTime` festgelegt ist.

Wenn der ETL-Prozess mehr als 24 Stunden für die Ausführung benötigt und dadurch die Startzeit am nächsten Tag versäumt, überspringt er diesen Tag und wird zur geplanten Zeit am nächsten Tag ausgeführt. Beispiel: Wenn der ETL-Prozess so konfiguriert ist, dass er zwischen 1:00 und 3:00 ausgeführt wird und der Prozess um 1:00 am Montag startet und um 2:00 am Dienstag abgeschlossen wird, wird die nächste Ausführung, die ursprünglich für 1:00 am Dienstag geplant war, übersprungen, und der nächste ETL-Prozess startet um 1:00 am Mittwoch.

Die ETL-Planung berücksichtigt nicht die Sommerzeit. Wenn die Ausführung des ETL-Prozesses beispielsweise zwischen 1:00 und 3:00 geplant ist, könnte er um 0:00 oder 2:00 ausgeführt werden, wenn die Sommerzeit einsetzt.

Standardwert

Nein

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

processSleepIntervalInMinutes

Beschreibung

Die Anzahl von Minuten, die das Unica Interact-Modul für Kontakt- und Antwortverlauf wartet, bevor es Daten aus den Staging-Tabellen der Laufzeitumgebung von Unica Interact in die Unica Campaign-Kontakt- und Antwortverlaufstabellen kopiert.

Standardwert

60

Gültige Werte

Eine beliebige Ganzzahl größer 0.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

preferredStartTime

Beschreibung

Die bevorzugte Zeit für den Start des täglichen ETL-Prozesses. Wenn diese Eigenschaft zusammen mit der Eigenschaft "preferredEndTime" verwendet wird, legt sie das bevorzugte Zeitintervall für die Ausführung des ETL-Prozesses fest. Der ETL-Prozess startet während des angegebenen Zeitintervalls und verarbeitet maximal die mit `maxJDBCFetchBatchSize`

angegebene Anzahl von Datensätzen. Das Format ist HH:mm:ss AM oder PM unter Verwendung des 12-Stunden-Formats.

Standardwert

12:00:00 AM

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

preferredEndTime

Beschreibung

Die bevorzugte Zeit für den Abschluss des täglichen ETL-Prozesses. Wenn diese Eigenschaft zusammen mit der Eigenschaft "preferredStartTime" verwendet wird, legt sie das bevorzugte Zeitintervall für die Ausführung des ETL-Prozesses fest. Der ETL-Prozess startet während des angegebenen Zeitintervalls und verarbeitet maximal die mit `maxJDBCFetchBatchSize` angegebene Anzahl von Datensätzen. Das Format ist HH:mm:ss AM oder PM unter Verwendung des 12-Stunden-Formats.

Standardwert

2:00:00 AM

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

purgeOrphanResponseThresholdInMinutes

Beschreibung

Die Anzahl von Minuten, die das Unica Interact-Modul für Kontakt- und Antwortverlauf wartet, bevor Antworten ohne entsprechenden Kontakt bereinigt werden. So wird vermieden, dass Antworten ohne Kontakte protokolliert werden.

Standardwert

180

Gültige Werte

Eine beliebige Ganzzahl größer 0.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

maxJDBCInsertBatchSize

Beschreibung

Die maximale Anzahl der Datensätze eines JDBC-Batches vor dem Ausführen der Abfrage. Dies ist nicht die maximale Anzahl von Datensätzen, die das Unica Interact-Modul für Kontakt- und Antwortverlauf in einer einzelnen Iteration verarbeitet. Während jeder Iteration verarbeitet das Unica Interact-Modul für Kontakt- und Antwortverlauf alle verfügbaren Datensätze aus den Staging-Tabellen. Diese Datensätze werden jedoch in `maxJDBCInsertSize`-Datenblöcke unterteilt.

Standardwert

1000

Gültige Werte

Eine beliebige Ganzzahl größer 0.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

maxJDBCFetchBatchSize

Beschreibung

Die maximale Anzahl der Datensätze eines von der Staging-Datenbank abzurufenden JDBC-Batches. Sie müssen diesen Wert möglicherweise erhöhen, um die Leistung des Moduls für Kontakt- und Antwortverlauf zu optimieren.

Beispiel: Um 2,5 Millionen Kontaktverlaufsdatensätze pro Tag zu verarbeiten, sollten Sie `maxJDBCFetchBatchSize` auf einen höheren Wert als 2,5M festlegen, damit alle Datensätze für einen Tag verarbeitet werden.

Sie können dann `maxJDBCFetchChunkSize` und `maxJDBCInsertBatchSize` auf kleinere Werte festlegen (in diesem Beispiel vielleicht auf 50.000 bzw. 10.000). Einige Datensätze vom nächsten Tag werden möglicherweise ebenfalls verarbeitet, aber bis zum nächsten Tag beibehalten.

Standardwert

1000

Gültige Werte

Eine beliebige Ganzzahl größer 0

maxJDBCFetchChunkSize

Beschreibung

Die maximale Anzahl einer JDBC-Datenblockgröße von Daten, die während des ETL-Prozesses (ETL = Extrahieren, Transformieren, Laden) gelesen werden. In manchen Fällen kann eine Datenblockgröße, die größer als die Einfügegröße ist, die Geschwindigkeit des ETL-Prozesses verbessern.

Standardwert

1000

Gültige Werte

Eine beliebige Ganzzahl größer 0

deleteProcessedRecords

Beschreibung

Legt fest, ob Kontaktverlaufs- und Antwortverlaufsdatensätze beibehalten werden, nachdem sie verarbeitet wurden.

Standardwert

Ja

completionNotificationScript

Beschreibung

Gibt den absoluten Pfad zu einem Script an, das ausgeführt wird, wenn der ETL-Prozess abgeschlossen ist. Wenn Sie ein Script angeben, werden fünf Argumente an das Fertigstellungsbenachrichtigungsscript übergeben: Startzeit, Endzeit, Gesamtzahl der verarbeiteten Kontaktprotokoll- und Antwortverlaufdatensätze. Die Start- und Endzeit sind numerische Werte, die die Anzahl der seit 1970 vergangenen Millisekunden darstellen. Das Statusargument zeigt an, ob der ETL-Job ein Erfolg oder Misserfolg war. 0 zeigt an, dass der ETL-Job erfolgreich war. 1 zeigt einen Misserfolg an und dass der ETL-Job einige Fehler enthält.

Standardwert

Keine

fetchSize

Beschreibung

Ermöglicht es Ihnen, den JDBC-Abrufumfang beim Abrufen aus Staging-Tabellen festzulegen.

Passen Sie besonders bei Oracle-Datenbanken diese Einstellung an die Anzahl von Datensätzen an, die JDBC bei jedem Netz-Umlauf abrufen soll. Versuchen Sie bei umfangreichen Batches von mindestens 100KB den Wert 10000. Achten Sie darauf, hier keinen zu großen Wert zu verwenden, weil sich das auf die Speicherbelegung auswirkt und die Leistungszunahme vernachlässigbar, wenn nicht sogar negativ ist.

Standardwert

Keine

daysBackInHistoryToLookupContact

Beschreibung

Beschränkt die Anzahl der Datensätze, die während der Ausführung von Antwortverlaufsabfragen durchsucht werden, auf die Datensätze, die innerhalb der angegebenen Anzahl von vergangenen Tagen erstellt wurden. Für Datenbanken mit einer größeren Anzahl von Antwortverlaufsdatensätzen kann auf diese Weise die Verarbeitungszeit für Abfragen reduziert werden, indem die Suchperiode auf die angegebene Anzahl von Tagen eingeschränkt wird.

Wenn der Wert für "daysBackInHistoryToLookupContact" größer als Null ist, wird der RH-Join-Abfrage eine Datumseinschränkung hinzugefügt. Dies ist nützlich, wenn die Tabelle UA_DtlContactHist datumspartitioniert ist. Die Datumseinschränkung schränkt die Datensätze ein, nach denen innerhalb der Datumseinschränkung gesucht wird".

Der Standardwert 0 gibt an, dass alle Datensätze durchsucht werden.

Standardwert

0 (Null)

Campaign | Partitionen | Partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

Diese Konfigurationseigenschaften definieren die Datenquelle für das Unica Interact-Modul für Kontakt- und Antwortverlauf.

jndi-name

Beschreibung

Verwenden Sie die Eigenschaft `systemTablesDataSource`, um die JNDI-Datenquelle (Java™ Naming and Directory Interface) zu identifizieren, die auf dem Anwendungsserver (Websphere oder WebLogic) für die Unica Interact-Laufzeittabellen definiert ist.

Die Unica Interact-Laufzeitdatenbank ist die mit den DLL-Scripts `aci_runtime` und `aci_populate_runtime` gefüllte Datenbank und enthält beispielsweise (u.a.) die folgenden Tabellen: `UACI_CHOfferAttrib` und `UACI_DefaultedStat`.

Standardwert

Es ist kein Standardwert definiert.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

databaseType

Beschreibung

Datenbanktyp für die Unica Interact-Laufzeitdatenquelle.

Standardwert

SQL-Server

Gültige Werte

SQLServer | Oracle | DB2® | MariaDB

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

schemaName

Beschreibung

Der Name des Schemas, das die Staging-Tabellen des Moduls für Kontakt- und Antwortverlauf enthält. Dieser Name sollte mit den Tabellen der Laufzeitumgebung übereinstimmen.

Sie müssen kein Schema definieren.

Standardwert

Es ist kein Standardwert definiert.

Campaign | partionen | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

Diese Konfigurationseigenschaften definieren den Kontakttyp von Campaign, der zu Berichts- oder Lernzwecken einem "Kontakt" zugeordnet wird.

contacted

Beschreibung

Der Wert, der der Spalte `ContactStatusID` der Tabelle `UA_DtlContactHist` in den Unica Campaign-Systemtabellen für einen Angebotskontakt zugewiesen wird. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_ContactStatus` sein. Hinweise zum Hinzufügen von Kontakttypen finden Sie im Unica Campaign-Administratorhandbuch.

Standardwert

2

Gültige Werte

Eine Ganzzahl größer 0.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

Diese Konfigurationseigenschaften definieren die Antworten für das Akzeptieren oder Ablehnen für die Berichterstellung und das Lernmodul.

annehmen

Beschreibung

Der Wert, der der Spalte `ResponseTypeID` der Tabelle `UA_ResponseHistory` in den Systemtabellen von Unica Campaign für ein angenommenes Angebot zugewiesen wird. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` sein. Der Spalte `CountsAsResponse` sollte der Wert `1`, eine Antwort, zugewiesen werden.

Hinweise zum Hinzufügen von Antworttypen finden Sie im Unica Campaign-Administratorhandbuch.

Standardwert

3

Gültige Werte

Eine Ganzzahl größer 0.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Zurückweisen

Beschreibung

Der Wert, der der Spalte `ResponseTypeID` der Tabelle `UA_ResponseHistory` in den Systemtabellen von Unica Campaign für ein abgelehntes Angebot zugewiesen wird. Der Wert muss ein gültiger Eintrag in der Tabelle `UA_UsrResponseType` sein. Der Spalte `CountsAsResponse` sollte der Wert `2`, eine Ablehnung, zugewiesen werden. Hinweise zum Hinzufügen von Antworttypen finden Sie im Unica Campaign-Administratorhandbuch.

Standardwert

8

Gültige Werte

Eine beliebige Ganzzahl größer 0.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | Bericht

Diese Konfigurationseigenschaften definieren die Berichtsnamen bei der Integration in Cognos®.

interactiveCellPerformanceByOfferReportName

Beschreibung

Name für den Bericht "Erfolg von interaktiven Zellen nach Angebot". Dieser Name muss mit dem Namen dieses Berichts auf dem Cognos®-Server übereinstimmen.

Standardwert

Erfolg von interaktiven Zellen nach Angebot

treatmentRuleInventoryReportName

Beschreibung

Name für den Bericht "Inventar der Verfahrensregeln". Dieser Name muss mit dem Namen dieses Berichts auf dem Cognos®-Server übereinstimmen.

Standardwert

Bestandsaufnahme Verfahrensregeln des Kanals

deploymentHistoryReportName

Beschreibung

Name für den Bericht "Bereitstellungsverlaufsbericht". Dieser Name muss mit dem Namen dieses Berichts auf dem Cognos®-Server übereinstimmen

Standardwert

Verlauf der Kanalbereitstellung

Campaign | partionen | partition[n] | Interact | lernen

Diese Konfigurationseigenschaften ermöglichen eine Optimierung des integrierten Lernmoduls.

confidenceLevel

Beschreibung

Ein Prozentsatz, der angibt, wie stark das Lerndienstprogramm den gesammelten Daten vertrauen soll, bevor es von der Untersuchung zur Nutzung wechselt. Mit dem Wert 0 wird die Untersuchung effektiv beendet.

Diese Eigenschaft ist nur anwendbar, wenn die Eigenschaft `Interact > offerserving > optimizationType` für die Unica Interact-Laufzeit auf `BuiltInLearning` festgelegt ist.

Standardwert

95

Gültige Werte

Eine Ganzzahl aus dem Bereich von 0 bis 95, teilbar durch 5 oder 99.

validateonDeployment

Beschreibung

Wenn `No` festgelegt wird, prüft Unica Interact das Lernmodul bei der Bereitstellung nicht. Wenn `yes` festgelegt wird, prüft Unica Interact das Lernmodul bei der Bereitstellung.

Standardwert

`Nein`

Gültige Werte

Yes | No

maxAttributeNames

Beschreibung

Die maximale Anzahl von Lernattributen, die das Unica Interact-Lerndienstprogramm überwachen soll.

Diese Eigenschaft ist nur anwendbar, wenn die Eigenschaft `Interact > offerserving > optimizationType` für die Unica Interact-Laufzeit auf `BuiltInLearning` festgelegt ist.

Standardwert

`10`

Gültige Werte

Beliebige Ganzzahl.

maxAttributeValues

Beschreibung

Die maximale Anzahl von Werten, die das Unica Interact-Lernmodul für die einzelnen Lernattribute verfolgen soll.

Diese Eigenschaft ist nur anwendbar, wenn die Eigenschaft `Interact > offerserving > optimizationType` für die Unica Interact-Laufzeit auf `BuiltInLearning` festgelegt ist.

Standardwert

5

otherAttributeValue

Beschreibung

Der Standardname für den Attributtyp, der zur Darstellung aller Attributtype dient, die den Wert von `maxAttributeValues` überschreiten.

Diese Eigenschaft ist nur anwendbar, wenn die Eigenschaft `Interact > offerserving > optimizationType` für die Unica Interact-Laufzeit auf `BuiltInLearning` festgelegt ist.

Standardwert

`Andere`

Gültige Werte

Eine Zeichenfolge oder Zahl.

Beispiel

Steht maxAttributeValues auf 3 und otherAttributeValue auf "Other", verfolgt das Lernmodul die ersten drei Werte. Alle anderen Werte werden der Kategorie "Other" zugewiesen. Wenn Sie beispielsweise das Benutzerattribut Haarfarbe verfolgen möchten und die ersten fünf Benutzer die Haarfarbe schwarz, braun, blond, rot und grau haben, so verfolgt das Lerndienstprogramm die

Haarfarben schwarz, braun und blond. Die Farben rot und grau werden unter `otherAttributeValue` (Other) eingruppiert.

percentRandomSelection

Beschreibung

Der Prozentsatz der Zeit, während der das Lernmodul ein Zufallsangebot anzeigt. Wenn beispielsweise für "percentRandomSelection" der Wert "5" festgelegt wird, dann bedeutet dies, dass das Lernmodul während 5 % der Zeit (5 aus jeweils 100 Empfehlungen) unabhängig von der Bewertung ein Zufallsangebot anzeigt. Durch die Aktivierung von `percentRandomSelection` wird die Konfigurationseigenschaft `offerTieBreakMethod` außer Kraft gesetzt. Wenn `percentRandomSelection` aktiviert ist, wird diese Eigenschaft festgelegt, unabhängig davon, ob Lernen aktiviert oder inaktiviert ist oder ob integriertes oder externes Lernen verwendet wird.

Standardwert

5

Gültige Werte

Eine beliebige Ganzzahl zwischen 0 (inaktiviert die Funktion `percentRandomSelection`) und 100.

recencyWeightingFactor

Beschreibung

Die Dezimaldarstellung eines Prozentsatzes der Datenmenge, die durch den Wert von `recencyWeightingPeriod` definiert wird. Beispielsweise bedeutet der Standardwert 0,15, dass 15% der vom Lerndienstprogramm verwendeten Daten aus dem Wert von `recencyWeightingPeriod` stammen.

Diese Eigenschaft ist nur anwendbar, wenn die Eigenschaft `Interact > offerserving > optimizationType` für die Unica Interact-Laufzeit auf `BuiltInLearning` festgelegt ist.

Standardwert

0,15

Gültige Werte

Ein Dezimalwert kleiner als 1.

recencyWeightingPeriod

Beschreibung

Die Größe von Daten in Stunden, denen der Prozentsatz des Gewichts `recencyWeightingFactor` vom Lernmodul gewährt wurde. Beispielsweise bedeutet der Standardwert 120, dass der Wert von `recencyWeightingFactor` der vom Lernmodul verwendeten Daten aus den letzten 120 Stunden stammt.

Diese Eigenschaft ist nur anwendbar, wenn `optimizationType` auf `builtInLearning` festgelegt ist.

Standardwert

120

minPresentCountThreshold

Beschreibung

Die minimale Anzahl der Anzeigewiederholungen eines Angebots, bevor seine Daten in Berechnungen verwendet werden und das Lernmodul in den Untersuchungsmodus wechselt.

Standardwert

0

Gültige Werte

Eine Ganzzahl größer oder gleich 0.

enablePruning

Beschreibung

Wenn Sie `Yes` festlegen, bestimmt das Unica Interact-Lerndienstprogramm algorithmisch, wenn ein Lernattribut (Standard oder dynamisch) nicht

prognostiziert werden kann. Wenn ein Lernattribut nicht prognostiziert werden kann, wird dieses Attribut bei der Ermittlung des Gewichts für ein Angebot vom Lernmodul nicht berücksichtigt. Dieser Vorgang setzt sich fort, bis das Lernmodul Daten aggregiert.

Wenn dieser Wert auf `No` festgelegt ist, verwendet das Lernmodul immer alle Lernattribute. Dadurch, dass nicht prognostizierbare Attribute nicht gelöscht werden, arbeitet das Lernmodul möglicherweise nicht so präzise wie eigentlich möglich.

Standardwert

Ja

Gültige Werte

Yes | No

Campaign | partionen | partition[n] | Interact | lernen | learningAttributes | [learningAttribute]

Diese Konfigurationseigenschaften definieren die Lernattribute.

attributeName

Beschreibung

Jeder Wert von `attributeName` ist der Name eines Benutzerattributs, das vom Lernmodul überwacht werden soll. Dieser Wert muss mit dem Namen eines Name/Wert-Paars in den Sitzungsdaten übereinstimmen.

Diese Eigenschaft ist nur anwendbar, wenn die Eigenschaft `Interact > offerserving > optimizationType` für die Unica Interact-Laufzeit auf `BuiltInLearning` festgelegt ist.

Standardwert

Es ist kein Standardwert definiert.

Campaign | partionen | partition[n] | Interact | einsatz

Diese Konfigurationseigenschaften definieren die Bereitstellungseinstellungen.

chunkSize

Beschreibung

Die maximale Größe der Fragmentierung in KB für jedes Unica Interact-Bereitstellungspaket.

Standardwert

500

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | serverGroups | [serverGroup]

Diese Konfigurationseigenschaften definieren die Servergruppeneinstellungen.

serverGroupName

Beschreibung

Der Name der Unica Interact-Laufzeitservergruppe. Dies ist der Name, der auf der Registerkarte "Übersicht des interaktiven Kanals" angezeigt wird.

Standardwert

Es ist kein Standardwert definiert.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

Diese Konfigurationseigenschaften definieren die Unica Interact-Laufzeitserver.

instanceURL

Beschreibung

Die URL des Unica Interact-Laufzeitservers. Eine Servergruppe kann mehrere Unica Interact-Laufzeitserver enthalten, jeder Server muss allerdings unter einer neuen Kategorie erstellt werden.

Standardwert

Es ist kein Standardwert definiert.

Beispiel

`http://server:port/interact`

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | Ablaufsdiagramm

Diese Konfigurationseigenschaften definieren die Unica Interact-Laufzeitumgebung, die für Testläufe interaktiver Ablaufdiagramme verwendet wird.

serverGroup

Beschreibung

Der Name der Servergruppe von Unica Interact, die von Unica Campaign zur Ausführung eines Testlaufs verwendet wird. Dieser Name muss mit dem Kategorienamen übereinstimmen, den Sie unter `serverGroups` erstellen.

Standardwert

Es ist kein Standardwert definiert.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

dataSource

Beschreibung

Verwenden Sie die Eigenschaft `dataSource`, um die physische Datenquelle für Unica Campaign zu identifizieren, die beim Ausführen von Testläufen interaktiver Ablaufdiagramme verwendet werden soll. Diese Eigenschaft muss mit der von der Eigenschaft `Campaign > partitions > partitionN > dataSources` definierten Datenquelle für die Testlaufdatenquelle übereinstimmen, die für die Unica Interact-Entwicklungszeit definiert ist.

Standardwert

Es ist kein Standardwert definiert.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

eventPatternPrefix

Beschreibung

Die Eigenschaft `eventPatternPrefix` ist ein Zeichenfolgewert, der Ereignismusternamen vorangestellt wird, um deren Nutzung in Ausdrücken in Auswahl- oder Entscheidungsprozessen innerhalb interaktiver Ablaufdiagramme zu ermöglichen.

Wenn Sie diesen Wert ändern, müssen Sie globale Änderungen im interaktiven Kanal implementieren, damit diese aktualisierte Konfiguration wirksam wird.

Standardwert

`EventPattern`

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

Diese Konfigurationseigenschaften definieren den Standardzellcode für die Standardangebotstabelle. Sie müssen diese Eigenschaften nur konfigurieren, wenn Sie globale Angebotszuweisungen definieren.

DefaultCellCode

Beschreibung

Der Standardzellcode, den Unica Interact verwendet, wenn Sie keinen Zellencode in der Standardangebotstabelle definieren.

Standardwert

Es ist kein Standardwert definiert.

Gültige Werte

Eine Zeichenfolge, die mit dem in Unica Campaign definierten Zellencodeformat übereinstimmt.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

Diese Konfigurationseigenschaften definieren den Standardzellcode für die offersBySQL-Tabelle. Sie müssen diese Eigenschaften nur konfigurieren, wenn Sie SQL-Abfragen verwenden, um einen gewünschten Satz potenzieller Angebote zu erhalten.

DefaultCellCode

Beschreibung

Der Standardzellcode, den Unica Interact für alle Angebote in OffersBySQL-Tabellen verwendet, die in der Zellencodespalte einen Nullwert aufweisen

(oder bei denen die Zellencodespalte fehlt). Dieser Wert muss ein gültiger Zellencode sein.

Standardwert

Es ist kein Standardwert definiert.

Gültige Werte

Eine Zeichenfolge, die mit dem in Unica Campaign definierten Zellencodeformat übereinstimmt.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

Diese Konfigurationseigenschaften definieren den Standardzellcode für die Tabelle für die Bewertungsüberschreibung. Sie müssen diese Eigenschaften nur konfigurieren, wenn Sie einzelne Angebotszuweisungen definieren.

DefaultCellCode

Beschreibung

Der Standardzellcode, den Unica Interact verwendet, wenn Sie in der Tabelle für die Bewertungsüberschreibung keinen Zellencode definieren.

Standardwert

Es ist kein Standardwert definiert.

Gültige Werte

Eine Zeichenfolge, die mit dem in Unica Campaign definierten Zellencodeformat übereinstimmt.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | Partitionen | Partition[n] | Server | intern

Eigenschaften in dieser Kategorie geben Integrationseinstellungen und die internalID-Grenzwerte für die ausgewählte Unica Campaign-Partition an. Wenn Ihre Unica Campaign-Installation aus mehreren Partitionen besteht, legen Sie diese Eigenschaften für jede Partition fest, für die sie gelten sollen.

internalIdLowerLimit

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Die Eigenschaften `internalIdUpperLimit` und `internalIdLowerLimit` beschränken die internen IDs von Unica Campaign so, dass diese im angegebenen Bereich liegen. Beachten Sie, dass die Werte inklusiv sind: das heißt, Unica Campaign kann sowohl die untere als auch die obere Grenzwerte verwenden.

Standardwert

`0` (Null)

internalIdUpperLimit

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Die Eigenschaften `internalIdUpperLimit` und `internalIdLowerLimit` beschränken die internen IDs von Unica Campaign so, dass diese im angegebenen Bereich liegen. Die Werte sind inklusiv: das heißt, Unica Campaign kann sowohl die untere als auch die obere Grenzwerte verwenden. Wenn Unica Collaborate installiert ist, setzen Sie den Wert auf `2147483647`.

Standardwert

`4294967295`

deliverInstalled

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Gibt an, dass Unica Deliver installiert ist. Wenn Sie `Yes` auswählen, sind die Unica Deliver-Funktionen in der Unica Campaign-Benutzeroberfläche verfügbar.

Das Installationsprogramm setzt diesen Wert für die Standardpartition Ihrer Unica Deliver-Installation auf `Ja`. Für weitere Partitionen, auf denen Unica Deliver installiert ist, müssen Sie diese Eigenschaft manuell konfigurieren.

Standardwert

`No`

Gültige Werte

`Yes | No`

interactInstalled

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Nach der Installation der Unica Interact-Designumgebung sollte diese Konfigurationseigenschaft auf `Yes` gesetzt werden, um die Unica Interact-Designumgebung in Unica Campaign zu aktivieren.

Wenn Unica Interact nicht installiert ist, setzen Sie die Eigenschaft auf `No`. Wird die Eigenschaft auf `No` gesetzt, werden die Menüs und Optionen von Unica Interact nicht aus der Benutzeroberfläche entfernt. Um Menüs und Optionen zu entfernen, müssen Sie die Registrierung von Unica Interact mithilfe des `configTool`-Dienstprogramms manuell aufheben.

Standardwert

Nein

Gültige Werte

yes | no

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

MO_UC_integration

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Ermöglicht für die Partition die Integration mit Unica Plan, wenn die Integration in den **Platform**-Konfigurationseinstellungen aktiviert ist. Weitere Informationen finden Sie im Unica PlanUnica Campaign-Integrationshandbuch.

Standardwert

Nein

Gültige Werte

Yes | No

MO_UC_BottomUpTargetCells

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Für diese Partition werden Bottom-up-Zellen für Arbeitsblätter für Zielzellen erlaubt, wenn **MO_UC_integration** aktiviert wurde. Bei der Einstellung `Yes` sind sowohl Top-Down- als auch Bottom-Up-Zielzellen sichtbar, Bottom-Up-Zielzellen sind jedoch schreibgeschützt. Weitere Informationen finden Sie im Unica PlanUnica Campaign-Integrationshandbuch.

Standardwert

Nein

Gültige Werte

Yes | No

Legacy_campaigns

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Aktiviert für diese Partition den Zugriff auf Kampagnen, die vor der Integration von Unica Plan und Unica Campaign erstellt wurden. Gilt nur, wenn **MO_UC_integration** auf `Yes` gesetzt ist. Veraltete Kampagnen umfassen außerdem Kampagnen, die in Unica Campaign 7.x erstellt und mit Plan 7.x-Projekten verlinkt wurden. Weitere Informationen finden Sie im Unica PlanUnica Campaign-Integrationshandbuch.

Standardwert

Nein

Gültige Werte

Yes | No

Unica Plan - Angebotsintegration

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Aktiviert die Möglichkeit zur Verwendung von Unica Plan für die Ausführung von Lifecycle-Management-Aufgaben für Angebote in dieser Partition, wenn **MO_UC_integration** für diese Partition aktiviert wurde. Die Angebotsintegration muss in den **Platform**-Konfigurationseinstellungen

aktiviert sein. Weitere Informationen finden Sie im Unica PlanUnica Campaign-Integrationshandbuch.

Standardwert

Nein

Gültige Werte

Yes | No

UC_CM_integration

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Ermöglicht die Digital Analytics-Onlinesegmentintegration für eine Unica Campaign-Partition. Wenn Sie diesen Wert auf `Yes` festlegen, steht im Auswahlprozessfeld die Option zur Verfügung, **Digital Analytics-Segmente** als Eingabe zu verwenden. Um die Digital Analytics-Integration für die einzelnen Partitionen zu konfigurieren, wählen Sie **Einstellungen > Konfiguration > Unica Campaign** | **Partitionen** | **Partition[n]** | **Coremetrics** aus.

Standardwert

Nein

Gültige Werte

Yes | No

linkInstalled

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Gibt an, dass Link installiert ist. Wenn Sie "Ja" auswählen, ist die Funktion "Link-Verbindungen verwalten" in der Schnittstelle von Unica Campaign

verfügbar. Das Installationsprogramm legt diese Eigenschaft für die Standardpartition in Ihrer Link-Installation auf "Nein" fest. Für weitere Partitionen, auf denen Link installiert ist, müssen Sie diese Eigenschaft manuell konfigurieren.

Standardwert

Nein

Gültige Werte

Yes | No

numRowsReadToParseDelimitedFile

Konfigurationskategorie

`Campaign|partitions|partition[n]|server|internal`

Beschreibung

Diese Eigenschaft wird verwendet, wenn eine Datei mit begrenzter Satzlänge als Benutzertabelle zugeordnet wird. Zudem wird sie vom Prozessfeld "Bewertung" verwendet, wenn eine Bewertungsausgabedatei über IBM SPSS Modeler Advantage Enterprise Marketing Management Edition importiert wird. Um eine Datei mit begrenzter Satzlänge importieren oder zuordnen zu können, muss Unica Campaign die Datei zur Identifizierung der Spalten, Datentypen (Feldtypen) und Spaltenbreiten (Feldlängen) parsen.

Der Standardwert 100 bedeutet, dass Unica Campaign die ersten 50 und die letzten 50 Zeileneinträge in der abgegrenzten Datei untersucht. Unica Campaign weist dann die Feldlänge auf der Grundlage des größten Wertes, den es innerhalb dieser Einträge findet, zu. In den meisten Fällen reicht der Standardwert zur Ermittlung von Feldlängen aus. In sehr großen Dateien mit begrenzter Satzlänge überschreitet ein später hinzugefügtes Feld jedoch möglicherweise die von Unica Campaign berechnete geschätzte Länge. Dies kann einen Fehler während der Laufzeit des Ablaufdiagramms verursachen. Wenn Sie eine sehr große Datei zuordnen, können Sie diesen Wert daher erhöhen, damit Unica Campaign weitere Zeileneinträge überprüfen kann.

Bei dem Wert 100 kann Unica Campaign beispielsweise die ersten 100 Zeileneinträge und die letzten 100 Zeileneinträge der Datei überprüfen.

Bei dem Wert 0 wird die gesamte Datei überprüft. Dies ist in der Regel nur dann notwendig, wenn Sie Dateien importieren oder zuordnen, deren Felder eine variable Datenbreite aufweisen, die nicht durch das Lesen der ersten und letzten Zeilen ermittelt werden kann. Bei extrem großen Dateien kann sich die erforderliche Bearbeitungszeit durch das Lesen der gesamten Datei bei der Ausführung einer Tabellenzuordnung und des Prozessfelds "Bewertung" erhöhen.

Standardwert

100

Gültige Werte

0 (alle Zeilen) oder eine beliebige positive Ganzzahl

Campaign | monitoring

Die Eigenschaften in dieser Kategorie geben an, ob Operational Monitoring (Funktion zur Überwachung von Arbeitsabläufen) aktiviert ist, und legen die URL des Operational Monitoring-Servers sowie das Cachingverhalten fest. Operational Monitoring wird angezeigt und ermöglicht eine Steuerung aktiver Ablaufdiagramme.

cacheCleanupInterval

Beschreibung

Die Eigenschaft `cacheCleanupInterval` gibt das Intervall zwischen automatischen Bereinigungen des Statuscache für Ablaufdiagramme in Sekunden an.

Diese Eigenschaft ist in Unica Campaign-Versionen vor Version 7.0 nicht verfügbar.

Standardwert

`600 (10 Minuten)`

cacheRunCompleteTime

Beschreibung

Die Eigenschaft `cacheRunCompleteTime` gibt die Dauer in Minuten an, über die abgeschlossene Ausführungen zwischengespeichert werden und auf der Überwachungsseite angezeigt werden.

Diese Eigenschaft ist in Unica Campaign-Versionen vor Version 7.0 nicht verfügbar.

Standardwert

`4320`

monitorEnabled

Beschreibung

Die Eigenschaft `monitorEnabled` gibt an, ob die Überwachung aktiviert ist.

Diese Eigenschaft ist in Unica Campaign-Versionen vor Version 7.0 nicht verfügbar.

Standardwert

FALSE

Gültige Werte

TRUE | FALSE

serverURL

Beschreibung

Die Eigenschaft `Campaign > monitoring > serverURL` gibt die URL des Operational Monitoring-Servers an. Dies ist eine obligatorische Einstellung. Ändern Sie den Wert, wenn die URL des Operational Monitoring-Servers nicht dem Standardwert entspricht.

Wenn Unica Campaign für die Verwendung der SSL-Kommunikation (Secure Sockets Layer) konfiguriert ist, geben Sie als Wert dieser Eigenschaft die

HTTPS-Verbindungsdaten an. Beispiel: `serverURL=https://host:SSL_port/Campaign/OperationMonitor`. Dabei gilt Folgendes:

- *host* ist der Name oder die IP-Adresse des Computers, auf dem die Webanwendung installiert ist.
- *SSL_Port* ist der SSL-Port der Webanwendung.

Beachten Sie das `https` in der URL.

Standardwert

`http://localhost:7001/Campaign/OperationMonitor`

monitorEnabledForInteract

Beschreibung

Wenn der Wert auf `TRUE` festgelegt wird, wird der JMX-Connector-Server von Unica Campaign für Unica Interact aktiviert. Unica Campaign hat keine JMX-Sicherheit.

Steht dieser Wert auf `FALSE`, können Sie keine Verbindung zum Unica Campaign-JMX-Connector-Server herstellen.

Diese JMX-Überwachung gilt nur für das Unica Interact-Modul für Kontakt- und Antwortverlauf.

Standardwert

FALSE

Gültige Werte

TRUE | FALSE

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

protocol

Beschreibung

Überwachungsprotokoll für den Unica Campaign-JMX-Connector-Server, wenn `monitorEnabledForInteract` auf "yes" festgelegt ist.

Diese JMX-Überwachung gilt nur für das Unica Interact-Modul für Kontakt- und Antwortverlauf.

Standardwert

JMXMP

Gültige Werte

JMXMP | RMI

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Port

Beschreibung

Überwachungsport für den Unica Campaign-JMX-Connector-Server, wenn `monitorEnabledForInteract` auf "yes" festgelegt ist.

Diese JMX-Überwachung gilt nur für das Unica Interact-Modul für Kontakt- und Antwortverlauf.

Standardwert

2004

Gültige Werte

Eine Ganzzahl zwischen 1025 und 65535.

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Unica Interact installiert ist.

Campaign | partionen | partition[n] | Interact | outboundChannels

Diese Konfigurationseigenschaften ermöglichen Ihnen eine Optimierung des Kanals für abgehende Nachrichten für ausgelöste Nachrichten.

Anmerkung: Unter diesem Knoten stehen nun die sofort einsatzbereiten Gateways mit den Namen "EMail", "MobilePush", "UBX" zur Verfügung.

Kategoriename

Beschreibung

Diese Eigenschaft definiert den Namen dieses Kanals für abgehende Nachrichten. Der Name muss für jeden Kanal für abgehende Nachrichten eindeutig sein.

Name

Beschreibung

Der Name Ihres Kanals für abgehende Nachrichten.

Anmerkung: Sie müssen Ihren Anwendungsserver erneut starten, damit die Änderungen wirksam werden.

Campaign | partionen | partition[n] | Interact | outboundChannels | Parameter Datei

Diese Konfigurationseigenschaften ermöglichen Ihnen eine Optimierung des Kanals für abgehende Nachrichten für ausgelöste Nachrichten.

Kategoriename

Beschreibung

Diese Eigenschaft definiert den Namen dieses Parameters. Der Name muss für jeden Parameter für diesen Kanal für abgehende Nachrichten eindeutig sein.

Wert

Beschreibung

Diese Eigenschaft definiert die Parameter, im Format von Name/Wert-Paaren, die von diesem ausgehenden Gateway benötigt werden.

Campaign | partitions | partition[n] | Interact | Simulator

Diese Konfigurationseigenschaften definieren die Servergruppe, die zur Ausführung von API-Simulationen verwendet werden soll.

serverGroup

Beschreibung

Geben Sie die Laufzeitservergruppe an, die zur Ausführung von API-Simulationen verwendet wird.

Standardwert

defaultServerGroup

offerArbitrition

Beschreibung

Mit OfferArbitration können Sie verschiedene Konfigurationen für Interact Design-Zeit im Zusammenhang mit Angeboten erstellen. Sie können damit Campaign als optionale Eigenschaft der Strategie festlegen. Es bietet auch Konfigurationsoptionen, die die Verwendung von Centralized Offers Management-APIs ermöglichen, wenn letzteres installiert ist.

isCampiagnRequiredForStrategy

Diese Einstellung kann zwei Werte haben.

- Yes: Eine Kampagne muss ausgewählt werden, wenn Sie eine Strategie erstellen/bearbeiten/kopieren.
- No: Eine Kampagne ist nicht erforderlich.

Standardwert

NO

relyOnOfferManagement

Setzen Sie diese Konfigurationsoption auf „Ja“, wenn Centralized Offers Management installiert ist. Dies ermöglicht die Verwendung von Centralized Offers Management-APIs in Interact Design-Zeit, um auf Informationen zu Angeboten zuzugreifen. Diese Einstellung kann zwei Werte haben.

- Yes: Aktivieren Sie die Verwendung von Centralized Offers Management-APIs.
- No: Verwenden Sie keine Centralized Offers Management-APIs.

Standardwert

NO

offerManagementURL

Die URL des Centralized Offers Management-Servers.

Standardwert

Es ist kein Standardwert definiert.

Beispiel

http://server:port/Offer

Verfügbarkeit

Diese Eigenschaft ist nur anwendbar, wenn Centralized Offers Management installiert ist.

connectionTimeout

Das API-Verbindungszeitlimit in Sekunden.

Standardwert

5 Sekunden

Kapitel 17. Echtzeit-Personalisierung von Angeboten auf der Clientseite

In bestimmten Situationen kann es sinnvoll sein, die Echtzeit-Personalisierung von Angeboten ohne Low-Level-Implementierung von SOAP-Aufrufen oder Java™ Code auf dem Unica Interact Server anzubieten. Zum Beispiel, wenn ein Besucher zunächst eine Webseite lädt, bei der Javascript-Inhalte Ihre einzige verfügbare erweiterte Programmierung sind, oder wenn ein Besucher eine E-Mail-Nachricht öffnet, bei der nur HTML-Inhalte möglich sind. Unica Interact bietet mehrere Konnektoren, die eine Echtzeitangebotsverwaltung in Fällen ermöglichen, in denen Sie die Kontrolle nur über den Webinhalt haben, der auf der Clientseite geladen wird, oder in denen Sie Ihre Schnittstelle zu Unica Interact vereinfachen möchten

Ihre Unica Interact Installation enthält zwei Konnektoren für die Angebotspersonalisierung, die auf der Clientseite aufgebaut wird:

- Informationen zum Unica Interact Message Connector *(auf Seite 623)*. Wenn Sie den Message Connector verwenden, können in Webinhalten in E-Mail-Nachrichten oder anderen elektronischen Medien zum Beispiel Link- und Image-Tags enthalten sein, über die Sie den Unica Interact Server aufrufen können, um Angebotspräsentationen und Landing-Pages zum Durchklicken auf der Seite zu laden.
- Informationen zum Unica Interact Web Connector *(auf Seite 646)*. Wenn Sie den Web Connector (auch JS Connector genannt) verwenden, können Webseiten auf der Clientseite JavaScript™ verwenden, um die Prioritäten, die Präsentation und den Kontakt- oder Antwortverlauf von Angeboten über Angebotspräsentationen und Landing-Pages zum Durchklicken auf der Seite zu verwalten.

Informationen zum Unica Interact Message Connector

Mit dem Unica Interact Message Connector können E-Mail-Nachrichten und andere elektronische Medien Unica Interact aufrufen, um während der Öffnungszeit die Präsentation personalisierter Angebote zu ermöglichen. Außerdem kann sich der Kunde jederzeit durch die Nachricht bis zur angegebenen Website klicken. Dies wird durch die

Verwendung von zwei Schlüssel-Tags erreicht: Der Image-Tag (`IMG`), der während der Öffnungszeit die personalisierten Angebote lädt, und der Link-Tag (`A`), der Informationen zum Durchklicken bereitstellt und den Kunden auf eine bestimmte Landing-Page weiterleitet.

Beispiel

Das folgende Beispiel zeigt einen HTML-Code, den Sie in eine Werbefläche einschließen können (z. B. innerhalb einer E-Mail-Nachricht). Der Code enthält sowohl eine URL für den `IMG`-Tag (damit können Informationen übergeben werden, wenn das Dokument auf dem Unica Interact Server geöffnet wird, um im Gegenzug die entsprechende Grafik für das Angebot abzurufen) als auch eine URL für den `A`-Tag (damit wird festgelegt, welche Informationen beim Durchklicken an den Unica Interact Server übergeben werden):

```
<a href="http://www.example.com/MessageConnector/
offerClickthru.jsp?msgId=1234&linkId=1&userid=1&referral=test"><img
src="http:// www.example.com/MessageConnector/offerImage.jsp?
msgId=1234&linkId=1&userid=1&incomeLevel=5&incomeType=numeric"/></a>
```

Im folgenden Beispiel ist ein `IMG`-Tag in einen `A`-Tag eingebunden. Dies führt zu folgendem Verhalten:

1. Wenn die E-Mail-Nachricht geöffnet wird, erhält der Message Connector eine Anforderung, die die im `IMG`-Tag kodierten Informationen enthält: `msgID` und `linkID` für diese Nachricht, sowie Kundenparameter, die die Userid, das Einkommensniveau und die Einkommensart umfassen.
2. Diese Informationen werden über einen API-Aufruf an den Unica Interact Laufzeitserver übergeben.
3. Der Laufzeitserver gibt ein Angebot an den Message Connector zurück, der die URL der Grafik für das Angebot abruft und diese URL (einschließlich aller zusätzlichen Parameter) bereitstellt, bevor er die Grafikanfrage an die URL für dieses Angebot weiterleitet.
4. Das Angebot wird dem Kunden als Grafik angezeigt.

An diesem Punkt kann der Kunde auf die Grafik klicken, um auf das Angebot zu reagieren. Diese Klickabfolge mit dem `A`-Tag und dem zugehörigen `HREF`-Attribut (das die Ziel-URL

angibt) sendet eine weitere Anfrage an den Message Connector, um die Landing-Page abzurufen, die mit der URL für dieses Angebot verknüpft ist. Der Browser des Kunden wird dann auf die im Angebot konfigurierte Landing-Page umgeleitet.

Hinweis: Ein `A`-Tag ist für die Klickabfolge nicht notwendigerweise erforderlich. Das Angebot kann auch nur aus einem Bild bestehen, z. B. ein Coupon zum Ausdrucken durch den Kunden.

Installieren des Message Connectors

Die Dateien, die zum Installieren, Implementieren und Ausführen des Message Connectors erforderlich sind, wurden automatisch in der Unica Interact Installation des Laufzeitservers eingebunden. In diesem Abschnitt werden die Schritte zusammengefasst, die erforderlich sind, um den Message Connector zur Verfügung zu stellen.

Das Installieren und Implementieren des Message Connectors umfasst die folgenden Aufgaben:

- Optionales Konfigurieren der Standardeinstellungen für den Message Connector, wie in Konfigurieren des Message Connectors *(auf Seite 625)* beschrieben.
- Erstellen der Datenbanktabellen, die zum Speichern der Transaktionsdaten im Message Connector erforderlich sind, wie in Erstellen der Message Connector-Tabellen *(auf Seite 636)* beschrieben.
- Installieren der Message Connector-Webanwendung, wie in Bereitstellen und Ausführen des Message Connectors *(auf Seite 639)* beschrieben.
- Erstellen der `IMG`- und `A`-Tags in den Werbeflächen (z. B. E-Mails oder Webseiten), die zum Aufrufen der Message Connector-Angebote beim Öffnen und Durchklicken erforderlich sind, wie in Erstellen der Message Connector-Links *(auf Seite 640)* beschrieben.

Konfigurieren des Message Connectors

Bevor Sie den Message Connector bereitstellen können, müssen Sie die Konfigurationsdatei in Ihrer Installation an die jeweilige Umgebung anpassen. Dazu können Sie die XML-Datei `MessageConnectorConfig.xml` ändern, die sich im Message Connector-Verzeichnis

auf dem Unica Interact Laufzeitserver befindet, zum Beispiel `<Unica Interact_home>/msgconnector/config/MessageConnectorConfig.xml`.

Die Datei `MessageConnectorConfig.xml` enthält sowohl obligatorische als auch optionale Konfigurationseinstellungen. Alle Einstellungen, die Sie verwenden, müssen an Ihre spezifische Installation angepasst werden. Folgen Sie den hier dargestellten Arbeitsschritten, um die Konfiguration zu ändern.

1. Wenn der Message Connector bereits aktiviert wurde und auf dem Webanwendungsserver ausgeführt wird, müssen Sie den Message Connector inaktivieren, bevor Sie den Vorgang fortsetzen.
2. Öffnen Sie auf dem Unica Interact Laufzeitserver die Datei `MessageConnectorConfig.xml` in einem beliebigen Text- oder XML-Editor.
3. Nehmen Sie die erforderlichen Änderungen an den Konfigurationseinstellungen vor und vergewissern Sie sich, dass die folgenden *Pflichteinstellungen* für Ihre Installation korrekt sind.
    - `<interactUrl>`. Die URL des Unica Interact Laufzeitservers, auf dem der Message Connector läuft und mit dem die Tags der Message Connector-Seite verbunden werden sollen.
    - `<interactUsername>`. Der Unica Interact-Benutzername authentifiziert Message Connector mit der Unica Interact-Laufzeit. Er ist erforderlich, wenn die Unica Interact-API-Authentifizierung (`Affinium|interact|general|API`) auf `True` gesetzt ist.
    - `<interactPassword>`. Das Unica Interact-Kennwort authentifiziert Message Connector mit der Unica Interact-Laufzeit. Er ist erforderlich, wenn die Unica Interact-API-Authentifizierung (`Affinium|interact|general|API`) auf `True` gesetzt ist. Nur die Kennwörter, die mit dem Dienstprogramm encryptPasswords von Platform verschlüsselt wurden, sind gültig. Passwörter im Nur-Text-Format sind nicht erlaubt.
    - `<imageErrorLink>`, die URL, auf die der Message Connector umleitet, wenn eine angeforderte Grafik für das Angebot nicht ordnungsgemäß geladen werden kann.

- `<landingPageErrorLink>`, die URL, auf die der Nachrichten-Connector umleitet, wenn bei der Verarbeitung einer Anfrage für eine Angebots-Landingpage ein Fehler auftritt.
- `<audienceLevels>`, ein Abschnitt der Konfigurationsdatei, der eine oder mehrere Einstellungen für die Zielgruppenebene enthält und die Standardzielgruppenebene festlegt, sofern nicht im Link für den Message Connector angegeben. Mindestens eine Zielgruppenebene muss konfiguriert werden.

Alle Konfigurationseinstellungen werden ausführlich unter Konfigurationseinstellungen für den Message Connector (auf Seite 627) beschrieben.

4. Wenn Sie keine weiteren Konfigurationsänderungen vornehmen möchten, speichern und schließen Sie die Datei `MessageConnectorConfig.xml`.
5. Fahren Sie mit der Einrichtung und Bereitstellung des Message Connectors fort.

Konfigurationseinstellungen für den Message Connector

Um den Message Connector zu konfigurieren, können Sie die XML-Datei mit dem Namen `MessageConnectorConfig.xml` im Message Connector-Verzeichnis auf dem Unica Interact Laufzeitserver ändern, normalerweise `<Unica Interact_home>/msgconnector/config/MessageConnectorConfig.xml`. Hier werden die einzelnen Konfigurationen in dieser XML-Datei beschrieben. Hinweis: Wenn Sie diese Datei ändern, nachdem der Message Connector implementiert und aktiv ist, müssen Sie die Bereitstellung für den Message Connector erneut ausführen oder den Anwendungsserver erneut starten, um diese Einstellungen erneut zu laden, nachdem Sie die Datei geändert haben.

Allgemeine Einstellungen

Die folgende Tabelle enthält eine Liste mit den optionalen und erforderlichen Einstellungen, die im Abschnitt `generalSettings`der Datei `MessageConnectorConfig.xml` enthalten sind.

Tabelle 30. Allgemeine Einstellungen für den Message Connector

Eine Liste mit den optionalen und erforderlichen Einstellungen im Abschnitt *generalSettings* der Message Connector-Konfigurationsdatei

`<interactURL>`	Die URL des Unica Interact Laufzeitservers zum Bearbeiten der Aufrufe von Message Connector-Seitentags, zum Beispiel für den Laufzeitserver, auf dem der Message Connector ausgeführt wird. Dieses Element ist erforderlich.	`http://localhost:7001/interact`
`<interactUsername>`	Der Unica Interact-Benutzername authentifiziert Message Connector mit der Unica Interact-Laufzeit. Er ist erforderlich, wenn die Unica Interact-API-Authentifizierung (`Affinium|interact|general|API`) auf `True` gesetzt ist.	
`<interactPassword>`	Das Unica Interact-Kennwort authentifiziert Message Connector mit der Unica Interact-Laufzeit. Er ist erforderlich, wenn die Unica Interact-API-Authentifizierung (`Affinium|interact|general|API`) auf `True` gesetzt ist. Nur die Kennwörter, die	

Tabelle 30. Allgemeine Einstellungen für den Message Connector

Eine Liste mit den optionalen und erforderlichen Einstellungen im Abschnitt *generalSettings* der Message Connector-Konfigurationsdatei

(Fortsetzung)

	mit dem Dienstprogramm encryptPasswords von Platform verschlüsselt wurden, sind gültig. Passwörter im Nur-Text-Format sind nicht erlaubt.	
`<defaultDateTimeFormat>`	Das Standarddatumsformat.	`MM/dd/yyyy`
`<log4jConfigFileLocation>`	Die Position der Eigenschaftendatei Log4j. Die Angabe ist relativ zur *$MESSAGE_CONNECTOR_HOME* Umgebungsvariable, sofern gesetzt. Andernfalls ist dieser Wert relativ zum Rootpfad der Message Connector-Webanwendung.	`config/MessageConnectorLog4j.properties`

Standardparameterwerte

Die folgende Tabelle enthält eine Liste mit den optionalen und erforderlichen Einstellungen, die im Abschnitt `defaultParameterValues`der Datei `MessageConnectorConfig.xml` enthalten sind.

Tabelle 31. Standardparametereinstellungen für den Message Connector

Elemente, die in dem Abschnitt *defaultParameterValues* der *MessageConnectorConfig.xml*-Datei enthalten sind.

Element	Beschreibung	Standardwert
<interactiveChannel>	Der Name des interaktiven Standardkanals.	
<interactionPoint>	Der Name des Standardinteraktionspunkts.	
<debugFlag>	Legt fest, ob das Debugging aktiviert ist. Die zulässigen Werte sind true und false.	falsch
<contactEventName>	Der Standardname des übergebenen Kontaktereignisses.	
<acceptEventName>	Der Standardname des übergebenen Annahmeereignisses.	
<imageUrlAttribute>	Der Standardname des Angebotsattributs mit der URL für die Angebotsgrafik, sofern nicht im Message Connector-Link angegeben.	
<landingPageUrlAttribute>	Die Standard-URL für die Klickabfolge zur Landing-Page, sofern nicht im Message Connector-Link angegeben.	

Verhaltenseinstellungen

Die folgende Tabelle enthält eine Liste mit den optionalen und erforderlichen Einstellungen, die im Abschnitt `behaviorSettings`der Datei `MessageConnectorConfig.xml` enthalten sind.

Tabelle 32. Verhaltenseinstellungen für den Message Connector

Elemente, die in dem Abschnitt `behaviorSettings` der `MessageConnectorConfig.xml`-Datei enthalten sind.

Element	Beschreibung	Standardwert
`<imageErrorLink>`	Die URL, an die der Connector umleitet, wenn beim Verarbeiten einer Angebotsgrafikanfrage ein Fehler auftritt. Diese Einstellung ist erforderlich.	`/images/default.jpg`
`<landingPageError-Link>`	Die URL, an die der Connector umleitet, wenn beim Verarbeiten einer Landing-Page-Anfrage zur Klickabfolge ein Fehler auftritt. Diese Einstellung ist erforderlich.	`/jsp/default.jsp`
`<alwaysUseExisting-Offer>`	Legt fest, ob das in den Cache gestellte Angebot zurückgegeben werden soll, obwohl es bereits abgelaufen ist. Die zulässigen Werte sind `true` und `false` false.	`false`
`<offerExpireAction>`	Die Aktion, die durchgeführt werden soll, wenn das ursprüngliche Angebot gefunden	`RedirectToErrorPage`

Tabelle 32. Verhaltenseinstellungen für den Message Connector

Elemente, die in dem Abschnitt *behaviorSettings* der *MessageConnectorConfig.xml*-Datei enthalten sind.

(Fortsetzung)

Element	Beschreibung	Standardwert
	wird, aber bereits abgelaufen ist. Zulässige Werte sind: • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	

Speichereinstellungen

Die folgende Tabelle enthält eine Liste mit den optionalen und erforderlichen Einstellungen, die im Abschnitt storageSettingsder Datei MessageConnectorConfig.xml enthalten sind.

Tabelle 33. Speichereinstellungen für Message Connector

Elemente, die in dem Abschnitt *storageSettings* der *MessageConnectorConfig.xml*-Datei enthalten sind.

Element	Beschreibung	Standardwert
<persistenceMode>	Wenn der Cache neue Einträge in die Datenbank stellt. Die zulässigen Werte sind WRITE-BEHIND (dabei werden die Daten zunächst in den Cache geschrieben und zu einem späteren Zeitpunkt in der Datenbank aktualisiert) und WRITE-THROUGH (dabei werden die Daten gleichzeitig in den Cache und in die Datenbank geschrieben).	WRITE-THROUGH

Tabelle 33. Speichereinstellungen für Message Connector

Elemente, die in dem Abschnitt *storageSettings* der *MessageConnectorConfig.xml*-Datei enthalten sind.

(Fortsetzung)

Element	Beschreibung	Standardwert
<max-CacheSize>	Die maximale Anzahl an Einträgen im Speichercache.	5000
<max-Persistence-BatchSize>	Die maximale Stapelgröße, während Einträge in die Datenbank gestellt werden.	200
<mac-CachePersist-Interval>	Die maximale Zeit in Sekunden, wie lange ein Eintrag im Cache bleibt, bevor er in die Datenbank gestellt wird.	3
<maxelementonDisk>	Die maximale Anzahl an Einträgen im Plattencache.	5000
<cache-EntryTimeToEx-	Die maximale Zeitdauer, wie lange die Einträge im Plattencache bleiben, bevor sie ablaufen.	60000

Tabelle 33. Speichereinstellungen für Message Connector

Elemente, die in dem Abschnitt *storageSettings* der *MessageConnectorConfig.xml*-Datei enthalten sind.

(Fortsetzung)

Element	Beschreibung	Standardwert
pire-InSe-conds>		
<jdbc-Set-tings>	Ein Abschnitt der XML-Datei mit spezifischen Informationen, wenn eine JDBC-Verbindung verwendet wird. Dieser Abschnitt und der Abschnitt <dataSource-Settings> sind gegenseitig ausschließend.	Mit der Standardkonfiguration wird eine Verbindung mit der SQL Server-Datenbank hergestellt, die auf dem lokalen Server konfiguriert ist. Wenn Sie jedoch diesen Abschnitt aktivieren, müssen Sie die tatsächlichen JDBC-Einstellungen und die Berechtigungsnachweise für die Anmeldung angeben.
<data-Sour-ceSet-tings>	Ein Abschnitt der XML-Datei mit spezifischen Informationen, wenn eine Datenquellenverbindung verwendet wird. Dieser Abschnitt und der Abschnitt <jdbcSet-tings> sind gegenseitig ausschließend.	Mit der Standardkonfiguration wird eine Verbindung mit der InteractDS-Datenquelle hergestellt, die auf dem lokalen Webanwendungsserver definiert ist.

Zielgruppenebenen

Die folgende Tabelle enthält eine Liste mit den optionalen und erforderlichen Einstellungen, die im Abschnitt audienceLevelsder Datei MessageConnectorConfig.xml enthalten sind.

Hinweis: Das audienceLevels-Element wird optional verwendet, um die zu verwendende Standardzielgruppenebene anzugeben, sofern nicht im Message Connector-Link angegeben, wie im folgenden Beispiel dargestellt:

```
<audienceLevels default="Customer">
```

In diesem Beispiel stimmt der Wert für das Standardattribut mit dem Namen eines `audienceLevel` überein, das in diesem Abschnitt definiert ist. In dieser Konfigurationsdatei muss mindestens eine Zielgruppenebene definiert sein.

Tabelle 34. Zielgruppenebenen-Einstellungen für Message Connector

Eine Liste mit den optionalen und erforderlichen Einstellungen, die im Abschnitt *`audienceLevels`* der Datei *`MessageConnectorConfig.xml`* enthalten sind.

Element	Element	Beschreibung	Standardwert
`<audienceLevel>`		Das Element, das die Konfiguration der Zielgruppenebene enthält. Geben Sie ein Namensattribut an, wie in `<audienceLevel name="Customer">`	
	`<messageLogTable>`	Der Name der Protokolltabelle. Dieser Wert ist erforderlich.	`UACI_MESSAGE_CONNECTOR_LOG`
`<fields>`	`<field>`	Die Definition eines oder mehrerer Felder mit der jeweiligen Zielgruppen-ID für `audienceLevel`.	
	`<name>`	Der Name des Feldes mit der Zielgruppen-ID,	

Tabelle 34. Zielgruppenebenen-Einstellungen für Message Connector

Eine Liste mit den optionalen und erforderlichen Einstellungen, die im Abschnitt *audienceLevels* der Datei *MessageConnectorConfig.xml* enthalten sind.

(Fortsetzung)

Element	Element	Beschreibung	Standardwert
		wie in der Interact-Laufzeit angegeben.	
	<httpParameterName>	Der entsprechende Parametername für dieses Feld mit der Zielgruppen-ID.	
	<dbColumnName>	Der entsprechende Spaltenname in der Datenbank für dieses Feld mit der Zielgruppen-ID.	
	<type>	Der Typ des Feldes mit der Zielgruppen-ID, wie in der Interact-Laufzeit angegeben. Die Werte können string oder numeric sein.	

Erstellen der Message Connector-Tabellen

Bevor Sie den Unica Interact Message Connector bereitstellen können, müssen Sie zunächst die Tabellen in der Datenbank erstellen, in der die Unica Interact Laufzeitdaten gespeichert werden. Sie müssen für jede definierte Zielgruppenebene jeweils eine Tabelle erstellen. Unica Interact verwendet die erstellten Tabellen, um für jede Zielgruppenebene die Informationen zu den Transaktionen in Message Connector aufzuzeichnen.

Führen Sie mit Ihrem Datenbankclient im Message Connector das SQL-Script für die entsprechende Datenbank oder das entsprechende Schema aus, um die erforderlichen Tabellen zu erstellen. Die SQL-Scripts für die unterstützte Datenbank werden automatisch installiert, wenn Sie den Unica Interact Laufzeitserver installieren. Weitere Informationen zum Herstellen einer Verbindung mit der Datenbank, die die Unica Interact Laufzeittabellen enthält, finden Sie in den Arbeitsblättern, die Sie im *Unica Interact-Installationshandbuch* ausgefüllt haben.

1. Starten Sie den Datenbankclient und stellen Sie eine Verbindung mit der Datenbank her, in der gegenwärtig die Unica Interact Laufzeittabellen gespeichert werden.
2. Führen Sie das entsprechende Script im Verzeichnis `<Unica Interact_home>/msgconnector/scripts/ddl` aus. In der folgenden Tabelle sind die SQL-Beispielscripts aufgeführt, die Sie zum manuellen Erstellen der Message Connector-Tabellen verwenden können:

Tabelle 35. Scripts zum Erstellen von Message Connector-Tabellen

Datenquellentyp	Scriptname
IBM® DB2®	`db_scheme_db2.sql`
Microsoft™ SQL-Server	`db_scheme_sqlserver.sql`
Oracle	`db_scheme_oracle.sql`
MariaDB	`db_scheme_mariadb.sql`

Beachten Sie, dass diese Scripts als Muster bereitgestellt werden. Wenn Sie andere Namenskonventionen oder Strukturen für die Werte der Zielgruppen-ID verwenden, müssen Sie das Script ändern, bevor Sie es ausführen. Im Allgemeinen hat sich das Verfahren bewährt, jeweils eine Tabelle pro Zielgruppenebene zuzuordnen.

Die erstellten Tabellen müssen die folgenden Informationen enthalten:

Tabelle 36. Informationen, die von SQL-Beispielscripts erstellt werden

Eine Liste mit den von den SQL-Beispielscripts erstellten Spalten und Beschreibungen.

Spaltenname	Beschreibung
`LogId`	Der Primärschlüssel dieses Eintrags.
`MessageId`	Die eindeutige ID jeder Instanz für die Nachrichtenübertragung.
`LinkId`	Die eindeutige ID für jeden Link in den elektronischen Medien (z. B. E-Mail-Nachricht).
`OfferImageUrl`	Die URL für die zugehörige Grafik des zurückgegebenen Angebots.
`OfferLandingPage-Url`	Die URL der zugehörigen Landing-Page des zurückgegebenen Angebots.
`TreatmentCode`	Der Verfahrenscode des zurückgegebenen Angebots.
`OfferExpiration-Date`	Ablaufdatum und Uhrzeit des zurückgegebenen Angebots.
`OfferContactDate`	Datum und Uhrzeit, wann das Angebot an den Kunden zurückgegeben wurde.
`AudienceId`	Die Zielgruppen-ID der elektronischen Medien.

Beachten Sie folgende Hinweise zur dieser Tabelle:

- Abhängig von der Zielgruppenebene gibt es eine AudienceId-Spalte für jede Komponente des Zielgruppenschlüssels.
- Die Kombination aus MessageId, LinkId und AudienceId(s) bildet einen eindeutigen Schlüssel dieser Tabelle.

Wenn das Script erfolgreich ausgeführt wurde, haben Sie die erforderlichen Tabellen für den Message Connector erstellt.

Sie können jetzt die Webanwendung für den Message Connector bereitstellen.

Bereitstellen und Ausführen des Message Connectors

Der Unica Interact Message Connector wird als eigenständige Webanwendung auf einem unterstützten Webanwendungsserver implementiert.

Stellen Sie vor der Implementierung des Message Connectors sicher, dass die folgenden Aufgaben abgeschlossen sind:

- Sie müssen den Unica Interact Laufzeitserver installiert haben. Die implementierbare Message Connector-Anwendung wird automatisch zusammen mit dem Laufzeitserver installiert und kann im Unica Interact Ausgangsverzeichnis bereitgestellt werden.
- Sie müssen auch die SQL-Scripts ausgeführt haben, die mit der Installation bereitgestellt wurden, um die erforderlichen Tabellen in der Unica Interact Laufzeitdatenbank zu erstellen, die der Message Connector verwendet, wie in Erstellen der Message Connector-Tabellen *(auf Seite 636)* beschrieben

Sie müssen zunächst die Message Connector-Anwendung bereitstellen und verfügbar machen wie alle anderen Anwendungen, die auf einem Webanwendungsserver ausgeführt werden sollen.

1. Stellen Sie mit den erforderlichen Berechtigungen eine Verbindung mit der Managementbenutzeroberfläche für den Webanwendungsserver her, um eine Anwendung bereitzustellen.
2. Folgen Sie den Anweisungen zu Ihrem Web-Applikationsserver, um diese Datei bereitzustellen und auszuführen: `<Unica Interact_home>/msgconnector/MessageConnector.war`
   Ersetzen Sie <*Unica Interact_home*> durch das tatsächliche Verzeichnis, in dem der Unica Interact-Laufzeitserver installiert ist.
3. Folgende Schritte sind bei der Konfiguration des Message Connectors mit JBOSS erforderlich
   a. Fügen Sie ein Datenbankmodul hinzu.
   b. Notieren Sie den Namen des Moduls, der zum Zeitpunkt der Modulerstellung angegeben wurde.

c. Wechseln Sie zum unkommentierten Abschnitt <INSTALL_DIR>/Interact/msgconnector/MessageConnector.war/WEB-INF/jboss-deployment-structure.xml in Abhängigkeit und geben Sie den Modulnamen so an, wie er zum Zeitpunkt der Modulerstellung angegeben wurde. Beispiel: - Das Modul, das zum Zeitpunkt der Konfiguration von JBoss für DB2 erstellt wurde, lautet ---> com.ibm, dann sollte jboss-depolyment-structure.xml Einträge haben, wie nachstehend gezeigt: <?xml version="1.0" encoding="UTF-8"?><deployment><dependencies><module name="com.ibm" export="true"/></dependencies></deployment></jboss-deployment-structure>

Der Message Connector kann jetzt verwendet werden. Nachdem Sie die Unica Interact Installation zum Erstellen der erforderlichen Daten konfiguriert haben, die der Message Connector verwendet, um Angebote zu unterbreiten, z. B. interaktive Kanäle, Strategien, Ablaufdiagramme, Angebote und so weiter, können Sie in den elektronischen Medien die Links erstellen, die der Message Connector annimmt.

Erstellen der Message Connector-Links

Sie müssen entsprechende Links erstellen und in Ihre Nachricht einbinden, wenn Sie den Message Connector verwenden möchten, um benutzerdefinierte Angebotsgrafiken und Landing-Pages bereitzustellen, wenn ein Endbenutzer mit den elektronischen Medien interagiert (z. B. durch Öffnen einer E-Mail-Nachricht) und sich durch das Angebot klickt. Dieser Abschnitt enthält eine Übersicht über die HTML-Tags, die Sie für diese Links benötigen.

Unabhängig davon, welches System Sie zum Generieren von abgehenden Nachrichten an die Endbenutzer verwenden, müssen Sie das HTML-Tagging generieren, das die entsprechenden Felder enthält (die in den HTML-Tags als Attribute angegeben werden), in denen die Informationen enthalten sind, die an den Interact-Laufzeitserver übergeben werden sollen. Folgen Sie den Anweisungen unten, um die Minimalanforderungen zu konfigurieren, die für eine Nachricht im Message Connector erforderlich sind.

Hinwies: Obwohl sich die Anweisungen hier speziell auf Nachrichten beziehen, die Message Connector-Links enthalten, können Sie dieses Verfahren auch zur Konfiguration verwenden, um Webseiten oder beliebigen anderen elektronischen Medien Links hinzuzufügen.

1. Sie müssen mindestens die folgenden Parameter verwenden, um den `IMG`-Link zu erstellen, der in der Nachricht angezeigt werden soll:
   - `msgID`, um die eindeutige ID für diese Nachricht anzugeben.
   - `linkID`, um die eindeutige ID für den Link in der Nachricht anzugeben.
   - `audienceID`, um die Kennung der Zielgruppe anzugeben, zu der der Empfänger der Nachricht gehört.

     Hinweis: Wenn sich die Zielgruppen-ID aus mehreren ID-Komponenten zusammensetzt, müssen alle Komponenten im Link enthalten sein.

   Sie können auch optionale Parameter einbeziehen, um die Zielgruppenebene, den Namen des interaktiven Kanals, den Namen des Interaktionspunkts, die URL mit der Position der Grafik und eigene benutzerdefinierte Parameter anzugeben, die nicht speziell vom Message Connector verwendet werden.
2. Optional können Sie auch einen `A`-Link erstellen, der den `IMG`-Link einschließt, damit der Benutzer auf das Bild klicken kann, um die entsprechende Seite mit dem Angebot im Browser zu laden.
   Der `A`-Link muss auch die drei oben genannten Parameter (`msgID`, `linkID`und `audienceID`) und alle optionalen Parameter (Zielgruppenebene, Name des interaktiven Kanals und Name des Interaktionspunkts) und benutzerdefinierten Parameter enthalten, die nicht speziell vom Message Connector verwendet werden.
   Hinweis: Der `A`-Link kann bei Bedarf auch eigenständig auf der Seite verwendet werden, obwohl er im Message Connector in den meisten Fällen auch einen `IMG`-Link enthält. Wenn der Link einen `IMG`-Link enthält, sollte der `IMG`-Link den gleichen Parametersatz enthalten wie der umschließende `A`-Link (einschließlich aller optionalen oder benutzerdefinierten Parameter).
3. Wenn alle Links korrekt definiert sind, können Sie die E-Mail-Nachrichten generieren und senden.

Beispiel-Links und weitere Informationen zu den verfügbaren Parametern finden Sie unter HTTP-Anforderungsparameter für die Tags "IMG" und "A" *(auf Seite 642)*

HTTP-Anforderungsparameter für die Tags "IMG" und "A"

Wenn Message Connector eine Anfrage erhält, entweder weil ein Endbenutzer eine E-Mail mit einem von Message Connector kodierten `IMG`-Tag geöffnet hat oder weil der Endbenutzer ein `A`-Tag durchgeklickt hat, analysiert er die in der Anfrage enthaltenen Parameter, um die entsprechenden Angebotsdaten zurückzugeben. Dieser Abschnitt enthält eine Liste der Parameter, die in der anfordernden URL enthalten sein können (entweder im `IMG`-Tag, das automatisch geladen wird, wenn ein in Tags eingeschlossenes Bild aufgerufen oder die E-Mail geöffnet wird, oder im `A`-Tag, das geladen wird, wenn sich der Benutzer durch die E-Mail-Nachricht zur angegebenen Website klickt).

Parameter

Wenn der Message Connector eine Anfrage empfängt, werden die in der Anfrage enthaltenen Parameter geparst. Diese Parameter können vollständig oder teilweise Folgendes enthalten:

Parametername	Syntax	Erforderlich?	Standardwert
`msgId`	Die eindeutige ID der E-Mail-Instanz oder Webseite.	Ja	Keine. Dies wird von dem System bereitgestellt, das die eindeutige Instanz der E-Mail-Nachricht oder der Webseite erstellt, die den Tag enthält.
`linkId`	Die eindeutige ID des Links in dieser E-Mail oder Webseite.	Ja	Keine. Dies wird von dem System bereitgestellt, das die eindeutige Instanz der E-Mail-Nachricht oder der Webseite erstellt, die den Tag enthält.
`audienceLevel`	Die Zielgruppenebene, zu der der Empfänger dieser Kommunikation gehört.	Nein	Der audienceLevel-Standardwert, der im `audienceLevels`-Element der Datei MessageConnectorConfig.xml angegeben ist.
`ic`	Der Name des interaktiven Zielkanals (IC)	Nein	Der Wert des `interactiveChannel`-Elements im Abschnitt `defaultParameterValues` der

Parametername	Syntax	Erfor-derlich?	Standardwert
			Datei MessageConnectorConfig.xml. Der Standardwert lautet "interactiveChannel".
`ip`	Der Name des Interaktionspunkts (IP)	Nein	Der Wert des `interactionPoint`-Elements im Abschnitt `defaultParameterValues` der Datei MessageConnectorConfig.xml. Der Standardwert lautet "headBanner".
`offerImageUrl`	Die URL der Zielangebotsgrafik für die IMG-URL in der Nachricht.	Nein	Keine.
`offerImageUrlAttr`	Der Name des Angebotsattributs mit der URL der Zielangebotsgrafik	Nein	Der Wert des `imageUrlAttribute`-Elements im Abschnitt `defaultParameterValues` der Datei MessageConnectorConfig.xml.
`offerLandingPage-Url`	Die URL der Landing-Page für das Zielangebot.	Nein	Keine.
`offerLandingPage-UrlAttr`	Der Name des Zielattributs mit der URL der Landing-Page für das Zielangebot.	Nein	Der Wert des `landingPageUrlAttribute`-Elements im Abschnitt `defaultParameterValues` der Datei MessageConnectorConfig.xml.
`contactEvent`	Der Name des Kontaktereignisses.	Nein	Der Wert des `contactEventName`-Elements im Abschnitt `defaultParameterValues` der Datei MessageConnectorConfig.xml. Der Standardwert lautet "contact".
`responseEvent`	Der Name des Annahmeereignisses.	Nein	Der Wert des `acceptEventName`-Elements im Abschnitt `defaultParameterValues` der Datei MessageConnectorConfig.xml. Der Standardwert lautet "accept".
`debug`	Das Debugflag. Setzen Sie diesen Parameter nur zur Fehlerbehebung und auf An-	Nein	Der Wert des `debugFlag`-Elements im Abschnitt `defaultParameterValues` der Datei

Parametername	Syntax	Erfor-derlich?	Standardwert
	weisung durch den technischen Support auf "true".		MessageConnectorConfig.xml. Der Standardwert lautet "false".
<audience id>	Die Zielgruppen-ID dieses Benutzers. Der Name dieses Parameters ist in der Konfigurationsdatei definiert.	Ja	Keine.

Wenn der Message Connector einen unbekannten Parameter empfängt (das heißt einen Parameter, der nicht in obiger Liste enthalten ist), gibt es zwei Möglichkeiten:

- Wenn ein unbekannter Parameter angegeben ist (z. B. "attribute", wie in `attribute="attrValue"`) und ein übereinstimmender Parameter mit dem gleichen Namen und dem Wortzusatz "Type" vorhanden ist (z. B. "attributeType", wie in `attributeType="string"`), erstellt der Message Connector einen übereinstimmenden Unica Interact-Parameter und übergibt diesen der Unica Interact-Laufzeit.

  Für den Typparameter sind folgende Werte zulässig:

  - Zeichenfolge
  - Zahl
  - datetime

    Für einen Typparameter "datetime" sucht der Message Connector auch nach einem Parameter mit dem gleichen Namen und dem Wortzusatz "Pattern" (z. B. "attributePattern"), falls ein gültiger Wert im Format für Datum/Uhrzeit vorhanden ist. Sie können zum Beispiel den Parameter `attributePattern="MM/dd/yyyy"` angeben.

    Hinweis: Wenn Sie den Parametertyp "datetime" angeben, ohne ein übereinstimmendes Datumsmuster anzugeben, wird der Wert verwendet, der in

der Message Connector-Konfigurationsdatei (in `<installation_directory>/msgconnector/config/MessageConnectorConfig.xml`) auf dem Unica Interact Server angegeben ist.

- Wenn ein unbekannter Parameter angegeben wird und kein übereinstimmender Typwert vorhanden ist, übergibt der Message Connector diesen Parameter an die URL für die Zielweiterleitung.

Der Message Connector übergibt alle unbekannten Parameter an den Unica Interact Laufzeitserver, ohne die Parameter zu verarbeiten oder zu speichern.

Beispielcode für den Message Connector

Der folgende `A`-Tag enthält ein Beispiel mit einem Satz von Message Connector-Links, die in einer E-Mail-Nachricht enthalten sein können:

```
<a
 href="http://www.example.com/MessageConnector/offerClickthru.jsp?msgId=234
   &linkId=1&userid=1&referral=xyz">
 <img
 src="http://www.example.com/MessageConnector/offerImage.jsp?msgId=234&link
Id=1
      &userid=1&incomeCode=3&incomeType=numeric"/>
</a>
```

In diesem Beispiel wird der `IMG`-Tag beim Öffnen der E-Mail-Nachricht automatisch geladen. Indem die Grafik aus der angegebenen Seite abgerufen wird, übergibt die Nachricht die Parameter für die eindeutige Nachrichten-ID (msgID), die eindeutige Link-ID (linkID) und die eindeutige Benutzer-ID (userid) zusammen mit den beiden zusätzlichen Parametern (incomeCode und incomeType) an die Unica Interact Laufzeit.

Der `A`-Tag stellt das HREF-Attribut (Hypertext Reference) bereit, das die Angebotsgrafik in einen Link zum Anklicken in der E-Mail-Nachricht umwandelt. Wenn der Endbenutzer die Grafik in der Nachricht anzeigt und sich dann zur Landing-Page durchklickt, werden die eindeutige Nachrichten-ID (`msgId`), die eindeutige Link-ID (`linkId`) und die eindeutige

Benutzer-ID (`userid`) jeweils zusammen mit allen zusätzlichen Parametern (`referral`) der URL für die Zielweiterleitung an den Server übergeben.

Informationen zum Unica Interact Web Connector

Der Unica Interact WebConnector (wird auch als JavaScript™ Connector oder JSConnector bezeichnet) bietet einen Service auf dem Unica Interact Laufzeitserver, mit dem JavaScript™ Unica Interact Java™-API aufrufen kann. Auf diese Weise können Webseiten die Unica Interact Echtzeit-Personalisierung von Angeboten aufrufen, indem nur der eingebettete JavaScript™-Code verwendet wird, ohne sich auf Web-Entwicklungssprachen (wie z. B. Java™, PHP, JSP und so weiter) verlassen zu müssen. So können Sie zum Beispiel ein kleines Snippet mit JavaScript™-Code auf jeder Seite Ihrer Website integrieren, um die von Unica Interact empfohlenen Angebote bereitzustellen. Dadurch wird bei jedem Seitenaufruf die Unica Interact API aufgerufen, um sicherzustellen, dass auf der geladenen Seite stets die besten Angebote für den Besucher der Website angezeigt werden.

Verwenden Sie den Unica Interact Web Connector in Situationen, in denen Sie den Besuchern auf einer Seite Angebote anzeigen möchten, ohne die serverseitige Anzeige der Seite programmatisch steuern zu können (wie das zum Beispiel mit PHP oder einem anderen serverbasierten Scripting der Fall wäre). Dazu können Sie im Seiteninhalt JavaScript™-Code integrieren, der vom Web-Browser des Besuchers ausgeführt wird.

Tipp: Die Dateien für den Unica Interact Web Connector werden auf dem Unica Interact Laufzeitserver automatisch im Verzeichnis `<Unica Interact_home>/jsconnector` installiert. Das Verzeichnis `<Unica Interact_home>/jsconnector` enthält die Datei ReadMe.txt mit wichtigen Informationen und Hinweisen zu den Funktionen des Web Connectors. Hier finden Sie auch Beispieldateien und den Quellcode des Web Connectors als Basis zur Entwicklung eigener Lösungen. Weitere Informationen finden Sie außerdem auch im Verzeichnis `jsconnector`.

Installieren des Web Connectors auf dem Laufzeitserver

Eine Instanz des Web Connectors wird automatisch mit dem Unica Interact Laufzeitserver installiert und standardmäßig aktiviert. Sie müssen jedoch einige Einstellungen ändern, bevor Sie den Web Connector konfigurieren und verwenden können.

Die Einstellungen, die geändert werden müssen, bevor Sie den auf dem Laufzeitserver installierten Web Connector verwenden können, werden der Konfiguration des Webanwendungsservers hinzugefügt. Aus diesem Grund muss der Webanwendungsserver erneut gestartet werden, nachdem Sie diese Schritte abgeschlossen haben.

1. Legen Sie für den Webanwendungsserver, auf dem der Unica Interact Laufzeitserver installiert ist, die folgenden Java™-Eigenschaften fest:

   `-DUI_JSCONNECTOR_ENABLE_INPROCESS=true`

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Ersetzen Sie `<jsconnectorHome>` durch den Pfad zum Verzeichnis `jsconnector` auf dem Laufzeitserver. Dieser lautet `<Unica Interact_Home>/jsconnector`.

   Die Art und Weise, wie die Java™-Eigenschaften festgelegt werden, hängt von Ihrem Webanwendungsserver ab. Beispiel: In WebLogic bearbeiten Sie die Datei `startWebLogic.sh` oder `startWebLogic.cmd`, um die Einstellung `JAVA_OPTIONS` zu aktualisieren, wie nachfolgend dargestellt:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/jsconnector"
   ```

   Im WebSphere® Application Server legen Sie diese Eigenschaft über die Administrationskonsole im Fenster der Java™ Virtual Machine fest.

   Weitere Informationen zum Einstellen spezifischer Java™-Eigenschaften finden Sie in der Dokumentation des Webanwendungsservers.
2. Der Webanwendungsserver muss jetzt neu gestartet werden, damit die neuen Java™-Eigenschaften übernommen werden.

Mit dem erfolgreichen Neustart des Webanwendungsservers ist die Installation des Web Connectors auf dem Laufzeitserver abgeschlossen. Als nächstes muss eine Verbindung

mit der Webseite zur Konfiguration unter `http://<host>:<port>/interact/jsp/WebConnector.jsp` hergestellt werden, wobei `<host>` den Unica Interact Namen des Laufzeitservers und `<port>` den Port angibt, den der Web Connector für das Listening verwendet, wie im Webanwendungsserver angegeben.

Installieren des Web Connectors als separate Webanwendung

Eine Instanz des Web Connectors wird automatisch mit dem Unica Interact Laufzeitserver installiert und standardmäßig aktiviert. Sie können den Web Connector jedoch auch als eigenständige Webanwendung implementieren (zum Beispiel in einem Webanwendungsserver auf einem separaten System) und für die Kommunikation mit dem fernen Unica Interact Laufzeitserver konfigurieren.

Diese Anleitung beschreibt, wie Sie den Web Connector als separate Webanwendung mit Zugriff auf einen fernen Unica Interact Laufzeitserver implementieren.

Bevor Sie den Web Connector implementieren können, müssen Sie den Unica Interact Laufzeitserver installiert haben. Außerdem benötigen Sie einen Webanwendungsserver auf einem anderen System mit Netzzugang (ohne Sperre durch eine Firewall) zum Unica Interact Laufzeitserver.

1. Kopieren Sie das Verzeichnis `jsconnector` mit den Web Connector-Dateien vom Unica Interact Laufzeitserver in das System, auf dem bereits der konfigurierte Webanwendungsserver (z. B. WebSphere® Application Server) ausgeführt wird. Das Verzeichnis `jsconnector` befindet sich in Ihrem Unica Interact-Installationsverzeichnis.
2. Konfigurieren Sie auf dem System, auf dem Sie die Web Connector-Instanz implementieren, die Datei `jsconnector/jsconnector.xml` in einem beliebigen Text- oder XML-Editor, um das Attribut `interactURL` zu ändern.

   Der Standardwert lautet `http://localhost:7001/interact`. Sie müssen den Wert ändern, damit er mit der URL des fernen Unica Interact Laufzeitservers übereinstimmt, z. B. `http://runtime.example.com:7011/interact`.

   Nachdem Sie den Web Connector implementiert haben, können Sie eine Webbenutzeroberfläche verwenden, um die weiteren Einstellungen in der Datei

`jsconnector.xml` anzupassen. Weitere Informationen finden Sie in Konfigurieren des Web Connectors (auf Seite 650).

3. Legen Sie für den Webanwendungsserver, auf dem Sie den Web Connector implementieren, die folgende Java™-Eigenschaft fest:

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Ersetzen Sie `<jsconnectorHome>` durch den tatsächlichen Pfad, in den Sie das Verzeichnis `jsconnector` auf dem Webanwendungsserver kopiert haben.

   Die Art und Weise, wie die Java™-Eigenschaften festgelegt werden, hängt von Ihrem Webanwendungsserver ab. Beispiel: In WebLogic bearbeiten Sie die Datei `startWebLogic.sh` oder `startWebLogic.cmd`, um die Einstellung `JAVA_OPTIONS` zu aktualisieren, wie nachfolgend dargestellt:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/Unica
   InteractFiles/jsconnector"
   ```

   Im WebSphere® Application Server legen Sie diese Eigenschaft über die Administrationskonsole im Fenster der Java™ Virtual Machine fest.

   Weitere Informationen zum Einstellen spezifischer Java™-Eigenschaften finden Sie in der Dokumentation des Webanwendungsservers.
4. Der Webanwendungsserver muss jetzt neu gestartet werden, damit die neue Java™-Eigenschaft übernommen wird.

   Warten Sie, bis der Startvorgang des Webanwendungsservers abgeschlossen ist, bevor Sie den Vorgang fortsetzen.
5. Stellen Sie mit den erforderlichen Berechtigungen eine Verbindung mit der Managementbenutzeroberfläche für den Webanwendungsserver her, um eine Anwendung bereitzustellen.
6. Folgen Sie den Anweisungen zum Bereitstellen des Webanwendungsservers und rufen Sie die folgende Datei auf:

   `jsConnector/jsConnector.war`

Der Web Connector wird jetzt in der Webanwendung implementiert. Wenn der vollständig konfigurierte und erneut gestartete Unica Interact Server ausgeführt wird, müssen Sie als Nächstes eine Verbindung mit der Webseite zur Konfiguration des Web Connectors unter

`http:// <host>: <port>/interact/jsp/WebConnector.jsp`herstellen. Dabei bezeichnet <*host*> das System, auf dem der Webanwendungsserver ausgeführt wird, auf dem Sie den Web Connector gerade implementiert haben, und `<port>` bezeichnet den Port, den der Web Connector für das Listening verwendet, wie im Webanwendungsserver angegeben.

Konfigurieren des Web Connectors

Die Konfigurationseinstellungen für den Unica Interact Web Connector werden in der Datei `jsconnector.xml` auf dem System gespeichert, auf dem der Web Connector bereitgestellt wird (dies kann der Unica Interact Laufzeitserver selbst oder ein separates System sein, auf dem ein Webanwendungsserver ausgeführt wird). Sie können die Datei `jsconnector.xml` direkt bearbeiten, indem Sie einen beliebigen Texteditor oder XML-Editor verwenden. Die meisten Konfigurationseinstellungen können jedoch auch erheblich leichter konfiguriert werden, indem Sie einfach die Seite zur Konfiguration des Web Connectors im Web-Browser aufrufen.

Bevor Sie die Webbenutzeroberfläche zum Konfigurieren des Web Connectors verwenden können, müssen Sie zunächst die Webanwendung installieren und implementieren, die den Web Connector bereitstellt. Auf dem Unica Interact Laufzeitserver wird automatisch eine Instanz des Web Connectors installiert, wenn Sie Unica Interact installieren und implementieren. Auf jedem anderen Webanwendungsserver müssen Sie die Web Connector-Webanwendung installieren und implementieren, wie in Installieren des Web Connectors als separate Webanwendung (auf Seite 648) beschrieben.

1. Öffnen Sie Ihren unterstützten Web-Browser und öffnen Sie eine URL, die der folgenden URL ähnlich ist:

   `http://<host>:<port>/interact/jsp/WebConnector.jsp`

   - Ersetzen Sie <host> durch den Server, auf dem der Web Connector ausgeführt wird, z. B. durch den Hostnamen des Laufzeitservers oder den Namen des Servers, auf dem Sie eine separate Instanz des Web Connectors bereitgestellt haben.

- Ersetzen Sie <port> durch die Portnummer, die der Web Connector für das Listening verwendet, um Verbindungen mit der Webanwendung herzustellen. Normalerweise kann der voreingestellte Standardport der Webanwendung übernommen werden.

2. Füllen Sie auf der anschließend angezeigten Seite "Konfiguration" die folgenden Abschnitte aus:

Tabelle 37. Zusammenfassung der Konfigurationseinstellungen für den Web Connector

Für jeden Abschnitt auf der Seite mit den Konfigurationseinstellungen für den Web Connector wird eine Kurzbeschreibung zur Verfügung gestellt, um zu beschreiben, wie Sie diesen Abschnitt verwenden. Ausführliche Informationen dazu finden Sie an anderer Stelle.

Abschnitt	Einstellungen
Basiseinstellungen	Auf der Seite mit den Basiseinstellungen können Sie das gesamte Verhalten des Web Connectors für die Website konfigurieren, auf der die in Tags eingeschlossenen Seiten aufgerufen werden sollen. Diese Einstellungen beinhalten die Basis-URL für die Website und Informationen über die Besucher, die Unica Interact verwenden, und ähnliche Einstellungen, die sich auf alle Seiten beziehen, die Sie mit dem Web Connector-Code in Tags einschließen möchten. Weitere Informationen finden Sie unter Basisoptionen zur Konfiguration des WebConnectors (auf Seite 654).
HTML-Anzeigetypen	Verwenden Sie die Seite HTML-Anzeigetypen, um den HTML-Code zu bestimmen, der für jeden Interaktionspunkt auf der Seite zur Verfügung gestellt wird. Sie können aus der Liste mit Standardvorlagen (FLT-Dateien) auswählen, die eine Kombination aus CSS-Code für Cascading Style Sheets, HTML-Code und JavaScript-Code für jeden Interaktionspunkt enthalten. Sie können die

Abschnitt	Einstellungen
	zur Verfügung gestellten Vorlagen verwenden und bei Bedarf anpassen oder eigene Vorlagen erstellen. Die Konfigurationseinstellungen auf dieser Seite beziehen sich auf den `interactionPoints` Abschnitt `jsconnector.xml` der Konfigurationsdatei. Weitere Informationen finden Sie unter HTML-Anzeigetypen zur Konfiguration von WebConnector (auf Seite 658).
Erweiterte Seiten	Auf der Seite 'Erweiterte Seiten' können Sie einem URL-Muster seitenspezifische Einstellungen zuweisen. Sie können zum Beispiel eine Seitenzuordnung einrichten, um für jede URL, die den Text "index.htm" enthält, die allgemeine Begrüßungsseite anzuzeigen, die spezielle Ereignisse zum Laden der Seite und definierte Interaktionspunkte für diese Zuordnung enthält. Die Konfigurationseinstellungen auf dieser Seite beziehen sich auf den `pageMapping` Abschnitt `jsconnector.xml` der Konfigurationsdatei. Weitere Informationen finden Sie unter Erweiterte Seiten zur Konfiguration von WebConnector (auf Seite 661).

3. Stellen Sie auf der Seite Basiseinstellungen sicher, dass die Website-weiten Einstellungen für die Installation gültig sind, und legen Sie optional den Debugmodus fest (nur empfohlen, wenn Sie die Fehlersuche zu einem Problem durchführen). Überprüfen Sie auch die Digital Analytics for On Premises Integration der Seiten-Tags und geben Sie die Standardeinstellungen für die Interaktionspunkte an, bevor Sie unter Konfigurationen auf den Link HTML-Anzeigetypen klicken.
4. Befolgen Sie diese Schritte auf der Seite HTML-Anzeigetypen, um für die Anzeige Vorlagen hinzuzufügen oder zu ändern, die die Interaktionspunkte auf der Webseite des Kunden definieren.

   Vorlagen für die Anzeige (FLT-Dateien) werden standardmäßig in `<jsconnector_home>/conf/html` gespeichert.

a. Wählen Sie in der Liste die FLT-Datei aus, die Sie überprüfen oder als Ausgangspunkt verwenden möchten, oder klicken Sie auf 'Typ hinzufügen', um eine neue, leere Vorlage für den Interaktionspunkt zu erstellen und zu verwenden.

   Neben der Vorlagenliste werden weitere Informationen zum Inhalt der Vorlage angezeigt, falls vorhanden.

b. Optional können Sie den Namen der Vorlage im Feld **Dateiname für diesen Anzeigetyp** ändern. Für eine neue Vorlage ändern Sie `CHANGE_ME.flt`, indem Sie einen beschreibenden Namen eingeben.

   Wenn Sie den Namen der Vorlage hier umbenennen, erstellt der Web Connector eine neue Datei mit diesem Namen, sobald Sie die Vorlage speichern. Vorlagen werden gespeichert, wenn Sie den Hauptteil des Textes ändern und dann zu einem anderen Feld navigieren.

c. Ändern oder vervollständigen Sie bei Bedarf die Informationen im HTML-Snippet, indem Sie CSS-Code für Style-Sheets, JavaScript™ und HTML-Code eingeben. Hinweis: Sie können auch Variablen eingeben, die während der Laufzeit durch Unica Interact Parameter ersetzt werden. Beispiel: `${offer.HighlightTitle}` wird an der angegebenen Position automatisch durch den Angebotstitel des Interaktionspunkts ersetzt.

   Verwenden Sie die Beispiele, die unter dem Feld HTML-Snippet angezeigt werden, um weitere Informationen zum Formatieren von CSS, JavaScript™ oder HTML-Codeblöcken zu erhalten.

5. Verwenden Sie bei Bedarf die Seite 'Erweiterte Seiten', um die Seitenzuordnungen einzurichten und festzulegen, wie mit bestimmten URL-Mustern auf den Seiten umgegangen werden soll.
6. Wenn Sie keine weiteren Konfigurationseigenschaften einrichten möchten, klicken Sie auf **Rollout der Änderungen durchführen**.
   Wenn Sie auf **Rollout der Änderungen durchführen** klicken, werden die folgenden Aktionen ausgeführt:

- Zeigt den Unica Interact Web Connector-Seitentag an, der den JavaScript™-Code enthält, den Sie auf der Web Connector-Seite kopieren und in den Webseiten einfügen können.
- Sichert die vorhandene Web Connector-Konfigurationsdatei auf dem Unica Interact Server (die Datei `jsconnector.xml` auf dem Server, auf dem der Web Connector installiert ist) und erstellt eine neue Konfigurationsdatei mit den von Ihnen definierten Einstellungen.

  Sicherungskonfigurationsdateien werden in *<jsconnector_home>/conf/archive/jsconnector.xml.<date>.<time>* als `jsconnector.xml.20111113.214933.750-0500` gespeichert (dabei entspricht 20111113 der Zeichenfolge für das Datum und 214933.750-0500 der Zeichenfolge für die Uhrzeit einschließlich der Zeitzone)

Sie haben jetzt die Konfiguration des Web Connectors abgeschlossen.

Um die Konfiguration zu ändern, können Sie entweder alle oben genannten Schritte erneut ausführen, um neue Werte einzugeben, oder die vorhandene Konfigurationsdatei(`<Unica Interact_home>/jsconnector/conf/jsconnector.xml`) in einem beliebigen Text- oder XML-Editor öffnen, um die vorhandenen Werte zu ändern.

Basisoptionen zur Konfiguration des WebConnectors

Verwenden Sie die Seite Basiseinstellungen der Seite zur Konfiguration des Web Connectors, um das Web Connector-Verhalten für die gesamte Website zu konfigurieren, auf der die in Tags eingeschlossenen Seiten aufgerufen werden sollen. Diese Einstellungen beinhalten die Basis-URL für die Website und Informationen über die Besucher, die Unica Interact verwenden, und ähnliche Einstellungen, die sich auf alle Seiten beziehen, die Sie mit dem Web Connector-Code in Tags einschließen möchten.

Website-weite Einstellungen

Die Website-weiten Einstellungen enthalten Konfigurationsoptionen für die globalen Einstellungen, die das Verhalten der gesamten Installation des konfigurierten Web Connectors beeinflussen. Sie können die folgenden Werte angeben:

Tabelle 38. Website-weite Einstellungen für die Web Connector-Installation

Einstellungen, die die gesamte Web Connector-Installation beeinflussen

Einstellung	Syntax	Funktional entsprechende Einstellung in js-connector.xml
Interact API-URL	Die Basis-URL des Unica Interact Laufzeitservers. **Anmerkung:** Diese Einstellung wird nur verwendet, wenn der Web Connector nicht innerhalb des Interact-Laufzeitservers ausgeführt wird (somit separat implementiert wurde).	*<interactURL>*
Web Connector-URL	Die Basis-URL, die zum Generieren der URL für die Klickabfolge verwendet wird.	*<jsConnectorURL>*
Name des interaktiven Kanals für die Ziel-Website	Der Name des interaktiven Kanals, den Sie auf dem Unica Interact Server definiert haben, der diese Seitenzuordnung darstellt.	*<interactiveChannel>*
Zielgruppenebene der Besucher	Die Unica Campaign Zielgruppenebene für die ankommenden Besucher; wird im API-Aufruf für die Unica Interact Laufzeit verwendet.	*<audienceLevel>*
Feldname der Zielgruppen-ID in der Profiltabelle	Name des audienceId-Feldes, das im API-Aufruf für Unica Interact verwendet wird. Hinweis: Zielgruppen-IDs für mehrere Felder werden gegenwärtig nicht unterstützt.	*<audienceIdField>*
Datentyp des Feldes Zielgruppen-ID	Der Datentyp des Feldes Zielgruppen-ID (entweder "numeric" oder "string"), der im API-Aufruf für Unica Interact verwendet wird.	*<audienceIdFieldType>*

Tabelle 38. Website-weite Einstellungen für die Web Connector-Installation

Einstellungen, die die gesamte Web Connector-Installation beeinflussen

(Fortsetzung)

Einstellung	Syntax	Funktional entsprechende Einstellung in js-connector.xml
Cookiename, der die Sitzungs-ID darstellt	Der Name des Cookies, das die Sitzungs-ID enthält.	*<sessionIdCookie>*
Cookiename, der die Besucher-ID darstellt	Der Name des Cookies, das die Besucher-ID enthält.	*<visitorIdCookie>*

Optionale Funktionen

Die Zusatzfunktionen sind optionale Konfigurationsoptionen und optionale globale Einstellungen für die Installation des konfigurierten Web Connectors. Sie können die folgenden Werte angeben:

Tabelle 39. Optionale Website-weite Einstellungen für die Web Connector-Installation

Optionale Einstellungen, die sich auf die gesamte Web Connector-Installation auswirken

Einstellung	Syntax	Funktional entsprechende Einstellung in js-connector.xml
Debugmodus aktivieren	Zeigt (mit `ja` oder `nein`), ob der spezielle Debugmodus verwendet wird. Wenn Sie diese Funktion aktivieren, enthält der vom Web Connector zurückgegebene Inhalt einen JavaScript-Aufruf mit einem Warnhinweis, der den Client über die gerade erfolgte Seitenzuordnung informiert. Der Client muss einen Eintrag in der unter <*authorizedDebugClients*> angegebenen Datei haben, um den Warnhinweis abzurufen.	<*enableDebugMode*>
Hostdatei für autorisierte Debug-Clients	Der Pfad zu einer Datei, die eine Liste mit qualifizierenden Hosts oder IP-Adressen (Internet Protocol) für den Debugmodus enthält. Der Hostname oder die IP-Adresse des Clients muss in der angegebenen Datei enthalten sein, damit die Debuginformationen erfasst werden.	<*authorizedDebugClients*>
Digital Analytics for On Premises Seitentag-Inte-	Zeigt (mit `ja` oder `nein`), ob der Web Connector den angegebenen Digital Analytics for On Premises-Tag am Ende des Seiteninhalts anhängen soll.	`<enable-NetInsight-Tagging>`

Tabelle 39. Optionale Website-weite Einstellungen für die Web Connector-Installation

Optionale Einstellungen, die sich auf die gesamte Web Connector-Installation auswirken (Fortsetzung)

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
gration aktivieren		
Digital Analytics for On Premises Tag HTML-Vorlagendatei	Die HTML/JavaScript-Vorlage, die verwendet wird, um einen Aufruf für den Digital Analytics for On Premises-Tag zu integrieren. Im Allgemeinen sollten Sie die Standardeinstellung übernehmen, solange Sie nicht aufgefordert werden, eine andere Vorlage bereitzustellen.	`<netInsight-Tag>`

HTML-Anzeigetypen zur Konfiguration von WebConnector

Verwenden Sie die Seite HTML-Anzeigetypen, um den HTML-Code zu bestimmen, der für jeden Interaktionspunkt auf der Seite zur Verfügung gestellt wird. Sie können aus der Liste mit Standardvorlagen (FLT-Dateien) auswählen, die eine Kombination aus CSS-Code für Cascading Style Sheets, HTML-Code und JavaScript™-Code für jeden Interaktionspunkt enthalten. Sie können die zur Verfügung gestellten Vorlagen verwenden und bei Bedarf anpassen oder eigene Vorlagen erstellen.

Anmerkung: Die Konfigurationseinstellungen auf dieser Seite beziehen sich auf den`interactionPoints` Abschnitt `jsconnector.xml` der Konfigurationsdatei.

Der Interaktionspunkt kann auch Platzhalter (Zonen) enthalten, in denen die Angebotsattribute automatisch abgelegt werden können. Beispiel: Wenn Sie `${offer.TREATMENT_CODE}` einschließen, wird dies während der Interaktion durch den Verfahrenscode ersetzt, der diesem Angebot zugeordnet ist.

Die Vorlagen auf dieser Seite werden automatisch aus den Dateien geladen, die im `<Unica Interact_home>/jsconnector/conf/html` Verzeichnis des Web Connector-Servers gespeichert sind. Alle neu erstellten Vorlagen werden ebenfalls in diesem Verzeichnis gespeichert.

Wenn Sie auf der Seite HTML-Anzeigetypen eine vorhandene Vorlage anzeigen oder ändern möchten, klicken Sie in der Liste auf die entsprechende `.flt`-Datei.

Um auf der Seite HTML-Anzeigetypen eine neue Vorlage zu erstellen, klicken Sie auf **Typ hinzufügen**.

Unabhängig von der Methode, die Sie zum Erstellen oder Ändern einer Vorlage verwenden, werden neben der Vorlagenliste die folgenden Informationen angezeigt:

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
Dateiname für diesen Anzei-	Der zugewiesene Name der Vorlage, die Sie bearbeiten. Der Name muss für das Betriebssystem zulässig sein, unter dem der Web Connector ausgeführt wird. Der Name darf zum Beispiel keinen Schrägstrich (/) enthalten, wenn Sie das Betriebssystem Microsoft™ Windows™verwenden. Wenn Sie eine neue Vorlage erstellen, ist in diesem Feld `CHANGE_ME.flt` voreingestellt. Ändern Sie diese	*<htmlSnippet>*

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
getyp	Voreinstellung, indem Sie einen aussagekräftigen Wert eingeben, bevor Sie den Vorgang fortsetzen.	
HTML-Snippet	Der spezifische Inhalt, den der Web Connector in den Interaktionspunkt auf der Webseite einfügen soll. Dieses Snippet kann HTML-Code, CSS-Formatierungsinformationen oder JavaScript™ zur Ausführung auf der Seite enthalten. Alle drei dieser Inhaltstypen müssen jeweils im Code BEGIN und END eingeschlossen werden, wie in den folgenden Beispielen dargestellt: • ${BEGIN_HTML} <your HTML content> ${END_HTML} • ${BEGIN_CSS} <your Interaction Point-specific stylesheet information> ${END_CSS} • ${BEGIN_JAVASCRIPT} <your Interaction Point-specific JavaScript code> ${END_JAVASCRIPT} Sie können auch mehrere vordefinierte Spezialcodes eingeben, die beim Laden der Seite automatisch ersetzt werden, zum Beispiel: • `${logAsAccept}` : Ein Makro übernimmt zwei Parameter (eine Ziel-URL und den TreatmentCode, der verwendet wird, um die Annahme des Angebots zu signalisieren) und ersetzt sie durch die URL für die Klickabfolge. • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}`	Keine funktional entsprechende Einstellung, weil das HTML-Snippet als eigenständige Datei außerhalb von jsconnector.xml bereitgestellt wird.

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
	• `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` Alle hier aufgelisteten Angebotscodes stellen Angebotsattribute dar, die in Unica Campaign in der Angebotsvorlage definiert sind, die der Marketier zum Erstellen der Angebote verwendet hat, die Unica Interact zurückgibt. Hinweis: Der Web Connector verwendet eine Vorlagenengine mit dem Namen FreeMarker. Diese bietet viele Zusatzoptionen, die beim Einrichten der Codes in den Seitenvorlagen hilfreich sein können. Weitere Informationen finden Sie in http://freemarker.org/docs/index.html.	
Beispiele für Spezialcodes	Enthält Beispiele für Typen von Spezialcodes, mit denen Sie zum Beispiel HTML-, CSS- und JAVASCRIPT-Blöcke oder Ablagezonen einfügen und kennzeichnen können, um auf spezifische Metadaten eines Angebots zu verweisen.	Keine funktional entsprechende Einstellung.

Die Änderungen an dieser Seite werden automatisch gespeichert, wenn Sie zu einer anderen Web Connector-Konfigurationsseite navigieren.

Erweiterte Seiten zur Konfiguration von WebConnector

Auf der Seite 'Erweiterte Seiten' können Sie einem URL-Muster seitenspezifische Einstellungen zuweisen. Sie können zum Beispiel eine Seitenzuordnung einrichten, um für jede eingehende URL, die den Text "index.htm" enthält, die allgemeine Begrüßungsseite

anzuzeigen, die spezielle Ereignisse zum Laden der Seite und definierte Interaktionspunkte für diese Zuordnung enthält.

Anmerkung: Die Konfigurationseinstellungen auf dieser Seite beziehen sich auf den`pageMapping` Abschnitt `jsconnector.xml` der Konfigurationsdatei.

Wenn Sie die Seite 'Erweiterte Seiten' verwenden möchten, um eine neue Seitenzuordnung zu erstellen, klicken Sie auf den Link **Seite hinzufügen** und geben Sie die erforderlichen Informationen für die Zuordnung an.

Seiteninformationen

Die Seiteninfo-Konfigurationsoptionen für die Seitenzuordnung definieren das URL-Muster, das als Trigger für diese Zuordnung verwendet wird, und einige zusätzliche Einstellungen für die Art und Weise, wie diese Seitenzuordnung von Unica Interact verarbeitet wird.

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
URL enthält	Dies ist das URL-Muster, das der Web Connector in den eingehenden Seitenanforderungen suchen soll. Beispiel: Wenn die anfordernde URL "mortgage.htm" enthält, können Sie das mit der entsprechenden Informationsseite abgleichen.	*<urlPattern>*
Anzeigename für diese Seite oder diesen Seitensatz	Ein beschreibender Name mit einer Beschreibung dieser Seitenzuordnung zur eigenen Verwendung, z. B. "Seite mit Informationen zur Hypothek".	*<friendlyName>*

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
Gibt Angebote auch als JSON-Daten zur JavaScript-Verwendung zurück	Eine Dropdown-Liste, mit der Sie angeben können, ob der Web Connector die Rohdaten des Angebots im Format JavaScript™ Object Notation (http://www.json.org/) am Ende des Seiteninhalts einbeziehen soll.	*<enableRawDataReturn>*

Ereignisse, die bei einem Besuch (onload) dieser Seite oder dieses Seitensatzes ausgelöst werden

Diese Konfigurationsoptionen für die Seitenzuordnung definieren das URL-Muster, das als Trigger für diese Zuordnung verwendet wird, und einige zusätzliche Einstellungen für die Art und Weise, wie diese Seitenzuordnung von Unica Interact verarbeitet wird.

Anmerkung: Die Konfigurationseinstellungen in diesem Abschnitt entsprechen dem Abschnitt `<pageLoadEvents>` der `jsconnector.xml`.

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
Einzelne Ereig-	Eine Liste mit Ereignissen, die für diese Seite oder diesen Seitensatz verfügbar sind. Die Ereignisse in dieser Liste sind in Unica Interact definiert. Wählen Sie mindestens ein Ereignis, das beim Laden der Seite ausgelöst werden soll. Für die Unica Interact API-Aufrufe gilt folgende Reihenfolge:	*<event>*

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
nisse	1. `startSession` 2. `postEvent` für jedes einzelne Ereignis, das beim Laden der Seite ausgelöst wird (sofern Sie die einzelnen Ereignisse in Unica Interact definiert haben) 3. Für jeden Interaktionspunkt: • `getOffers` • `postEvent(ContactEvent)`	

Interaktionspunkte (Positionen zum Anzeigen der Angebote) für diese Seite oder diesen Seitensatz

Mit diesen Konfigurationsoptionen für die Seitenzuordnung können Sie auswählen, welche Interaktionspunkte auf der Seite angezeigt werdenUnica Interact.

Anmerkung: Die Konfigurationseinstellungen in diesem Abschnitt entsprechen dem Abschnitt `<pageMapping> | <page> | <interactionPoints>` der `jsconnector.xml`.

Einstellung	Syntax	Funktional entsprechende Einstellung in js-connector.xml
Kontrollkästchen mit dem Namen des Interaktionspunkts	In diesem Abschnitt der Seite werden alle Interaktionspunkte angezeigt, die in der Konfigurationsdatei definiert sind. Wenn Sie das Kontrollkästchen neben dem Namen des Interaktionspunkts aktivieren, werden die verfügbaren Optionen angezeigt.	*<interactionPoint>*
HTML-Element-ID (Unica Interact nimmt die innerHTML-Einstellung vor)	Der Name des HTML-Elements, das den Inhalt für diesen Interaktionspunkt empfangen soll. Beispiel: Wenn Sie `<div id="welcomebanner">` auf der Seite angegeben haben, geben Sie `welcomebanner` (den ID-Wert) in dieses Feld ein.	*<htmlElementId>*
HTML-Anzeigetyp	Eine Dropdown-Liste, in der Sie den HTML-Anzeigetyp (die HTML-Snippets oder FLT-Dateien, die Sie zuvor auf einer anderen Web Connector-Konfigurationsseite definiert haben) für diesen Interaktionspunkt auswählen können.	*<htmlSnippet>*
Maximale Anzahl an Angeboten zur Präsentation (in einem Karus-	Die maximale Anzahl an Angeboten, die der Web Connector für diesen Interaktionspunkt vom Unica Interact Server abrufen soll. Dieses Feld ist optional und gilt nur für einen Interaktionspunkt, der die präsentierten Angebote regelmäßig aktualisiert, ohne die Seite erneut zu laden, z. B. im Karussellszenario, das mehrere Angebote abruft und jeweils immer nur ein Angebot präsentiert.	*<maxNumberOfOffers>*

Einstellung	Syntax	Funktional entsprechende Einstellung in jsconnector.xml
sell oder Daumenkino)		
Auszulösendes Ereignis, wenn das Angebot präsentiert wird	Der Name des Kontaktereignisses, das für diesen Interaktionspunkt übergeben werden soll.	*<contactEvent>*
Auszulösendes Ereignis, wenn das Angebot angenommen wird	Der Name des Annahmeereignisses, das in dem Moment, in dem auf den Angebotslink geklickt wird, für diesen Interaktionspunkt übergeben soll.	*<acceptEvent>*
Auszulösendes Ereignis, wenn das Angebot abgelehnt wird	Der Name des Ablehnungsereignisses, das für diesen Interaktionspunkt übergeben werden soll. **Anmerkung:** Diese Funktion wird derzeit noch nicht unterstützt	*<rejectEvent>*

Konfigurationsoptionen für den Web Connector

Im Allgemeinen können Sie die grafische Web Connector-Benutzeroberfläche verwenden, um die Web Connector-Einstellungen zu konfigurieren. Alle Einstellungen werden jedoch

auch in der Datei jsconnector.xml im Verzeichnis `jsconnector/conf` gespeichert. Hier werden die einzelnen Parameter beschrieben, die in der Konfigurationsdatei jsconnector.xml gespeichert sind.

Parameter und Beschreibungen

Die folgenden Parameter werden in der Datei `jsconnector.xml` gespeichert und für die Interaktionen im Web Connector verwendet. Es gibt zwei Möglichkeiten zum Ändern dieser Einstellungen:

- Verwenden der Webseite zur Konfiguration des Web Connectors. Die Seite ist automatisch verfügbar, nachdem Sie die Web Connector-Anwendung implementiert und gestartet haben. Um die Webseite zur Konfiguration zu verwenden, öffnen Sie in Ihrem Web-Browser eine URL, die der folgenden URL ähnlich ist: `http://<host>:<port>/interact/jsp/WebConnector.jsp`.

  Die Änderungen, die Sie in der Webseite zur Administration vornehmen, werden in der Datei `jsconnector.xml` auf dem Server gespeichert, auf dem Sie den Web Connector implementiert haben.
- Bearbeiten Sie die Datei `jsconnector.xml` direkt in einem beliebigen Text- oder XML-Editor. Stellen Sie sicher, dass Ihnen der Umgang mit XML-Tags und Werten vertraut ist, bevor Sie diese Methode verwenden.

  Anmerkung: Jedes Mal, wenn Sie die Datei `jsconnector.xml` manuell bearbeiten, können Sie diese Einstellungen erneut laden, indem Sie die Seite zur Administration des Web Connectors laden (unter `http://<host>:<port>/interact/jsp/jsconnector.jsp`) und dann auf **Konfiguration erneut laden** klicken.

Die folgende Tabelle beschreibt die Konfigurationsoptionen, wie Sie sie in der Datei `jsconnector.xml` einstellen können.

Tabelle 40. Konfigurationsoptionen für den Web Connector

Parametergruppe	Parameter	Syntax
`defaultPageBehavior`		
	`friendlyName`	Eine leicht lesbare Kennung, die anstelle des URL-Musters auf der Webseite zur Konfiguration des Web Connectors angezeigt wird.
	`interactURL`	Die Basis-URL des Interact-Laufzeitservers. Anmerkung: Dieser Parameter muss nur gesetzt werden, wenn Sie den Web Connector-Service (jsconnector) als implementierte Webanwendung ausführen. Es ist nicht erforderlich, diesen Parameter zu setzen, wenn der WebConnector automatisch als Teil des Interact-Laufzeitservers ausgeführt wird.
	`jsConnectorURL`	Die Basis-URL, die zum Generieren der URL für die Klickabfolge verwendet wird, wie z.B. `http://host:port/jsconnector/clickThru`
	`interactiveChannel`	Name des interaktiven Kanals, der diese Seitenzuordnung darstellt.
	`sessionIdCookie`	Name des Cookies, das die Sitzungs-ID enthält, die in den API-Aufrufen für Unica Interact verwendet wird.

Tabelle 40. Konfigurationsoptionen für den Web Connector (Fortsetzung)

Parametergruppe	Parameter	Syntax
	`visitorIdCookie`	Name des Cookies, das die Zielgruppen-ID enthält.
	`audienceLevel`	Die Zielgruppenebene der Kampagne, die für ankommende Besucher im API-Aufruf für die Unica Interact Laufzeit verwendet wird.
	`audienceIdField`	Name des `audienceId`-Feldes, das im API-Aufruf für die Unica Interact Laufzeit verwendet wird. **Anmerkung:** Anmerkung: Zielgruppen-IDs für mehrere Felder werden gegenwärtig nicht unterstützt.
	`audienceIdFieldType`	Der Datentyp [`numeric | string`], der für das Feld mit der Zielgruppen-ID im API-Aufruf für die Unica Interact Laufzeit verwendet wird
	`audienceLevelCookie`	Name des Cookies, das die Zielgruppenebene enthält. Dies ist optional. Wenn Sie diesen Parameter nicht setzen, verwendet das System die für `audienceLevel` definierte Einstellung.
	`relyOnExistingSession`	Wird im API-Aufruf für die Unica Interact Laufzeit verwendet. Im

Tabelle 40. Konfigurationsoptionen für den Web Connector (Fortsetzung)

Parametergruppe	Parameter	Syntax
		Allgemeinen ist dieser Parameter auf "true" gesetzt.
	`enableInteractAPIDebug`	Wird im API-Aufruf für die Unica Interact Laufzeit verwendet, um die Debuggerausgabe in den Protokolldateien zu aktivieren.
	`pageLoadEvents`	Das Ereignis, das übergeben wird, sobald diese bestimmte Seite geladen wird. Geben Sie ein oder mehrere Ereignisse innerhalb dieses Tags an, im ähnlichen Format wie `<event>event1</event>`.
	`interactionPointValues`	Alle Elemente in dieser Kategorie werden als Standardwerte für fehlende Werte in den IP-spezifischen Kategorien verwendet.
	`interactionPointValuescontactEvent`	Standardname des Kontaktereignisses, das für diesen bestimmten Interaktionspunkt übergeben werden soll.
	`interactionPointValuesacceptEvent`	Standardname des Annahmeereignisses, das für diesen bestimmten Interaktionspunkt übergeben werden soll.
	`interactionPointValuesrejectEvent`	Standardname des Ablehnungsereignisses, das für diesen bestimmten Interaktionspunkt übergeben

Tabelle 40. Konfigurationsoptionen für den Web Connector (Fortsetzung)

Parametergruppe	Parameter	Syntax
		werden soll. (Hinweis: Diese Funktion wird derzeit noch nicht unterstützt)
	`interactionPointValueshtmlSnippet`	Der Name der HTML-Vorlage, die für diesen Interaktionspunkt bereitgestellt wird.
	`interactionPointValuesmaxNumberOfOffers`	Standardwert für die maximale Anzahl an Angeboten, die Unica Interact für diesen Interaktionspunkt abruft.
	`interactionPointValueshtmlElementId`	Standardname des HTML-Elements, das den Inhalt für diesen Interaktionspunkt empfängt.
	`interactionPoints`	Diese Kategorie enthält die Konfiguration für jeden Interaktionspunkt. Für alle fehlenden Eigenschaften verwendet das System die Konfigurationen unter der Kategorie `interactionPointValues`.
	`interactionPointname`	Name des Interaktionspunkts (IP).
	`interactionPointcontactEvent`	Name des Kontaktereignisses, das für diesen bestimmten IP übergeben werden soll.
	`interactionPointacceptEvent`	Name des Annahmeereignisses, das für diesen bestimmten IP übergeben werden soll.

Tabelle 40. Konfigurationsoptionen für den Web Connector (Fortsetzung)

Parametergruppe	Parameter	Syntax
	`interactionPointrejectEvent`	Name des Ablehnungsereignisses, das für diesen bestimmten IP übergeben werden soll. (Hinweis: Diese Funktion wird derzeit noch nicht unterstützt.)
	`interactionPointhtmlSnippet`	Name der HTML-Vorlage, die für diesen IP bereitgestellt wird.
	`interactionPointmaxNumberOf-Offers`	Maximale Anzahl an Angeboten, die Unica Interact für diesen IP abruft
	`interactionPointhtmlElementId`	Name des HTML-Elements, das den Inhalt für diesen Interaktionspunkt empfängt.
	`enableDebugMode`	Boolesches Flag (akzeptable Werte: `true` oder `false`), um den speziellen Debug-Modus zu aktivieren. Wenn dieser Wert auf true gesetzt ist, enthält der vom Web Connector zurückgegebene Inhalt einen JavaScript™-Aufruf mit einem Warnhinweis, der den Client über die gerade erfolgte Seitenzuordnung informiert. Der Client muss einen Eintrag in der Datei `authorizedDebugClients` haben, um den Warnhinweis zu generieren.

Tabelle 40. Konfigurationsoptionen für den Web Connector (Fortsetzung)

Parametergruppe	Parameter	Syntax
	`authorizedDebugClients`	Eine vom speziellen Debugmodus verwendete Datei, die eine Liste mit qualifizierenden Hostnamen und IP-Adressen (Internet Protocol) für den Debugmodus enthält.
	`enableRawDataReturn`	Ein boolesches Flag (akzeptable Werte: (zulässige Werte: `true` oder `false`), um festzulegen, ob der Web Connector die Rohdaten des Angebots im JSON-Format am Ende des Inhalts anhängt.
	`enableNetInsightTagging`	Ein boolesches Flag (akzeptable Werte: `true` oder `false`), um festzulegen, ob der Web Connector einen Digital Analytics for On Premises-Tag am Ende des Inhalts anhängt.
	`apiSequence`	Stellt eine Implementierung der APISequence-Benutzeroberfläche dar, die die Reihenfolge der API-Aufrufe im Web Connector festlegt, wenn ein `pageTag` aufgerufen wird. Standardmäßig verwendet die Implementierung die Reihenfolge `StartSession`, `pageLoadEvents`, `getOffers` und `logContact`, wobei die letzten beiden spe-

Tabelle 40. Konfigurationsoptionen für den Web Connector (Fortsetzung)

Parametergruppe	Parameter	Syntax
		zifisch für jeden Interaktionspunkt sind.
	`clickThruApiSequence`	Stellt eine Implementierung der APISequence-Benutzeroberfläche dar, die die Reihenfolge der API-Aufrufe im Web Connector festlegt, wenn ein `clickThru` aufgerufen wird. Standardmäßig verwendet die Implementierung die Reihenfolge `StartSession` und `logAccept`.
	`netInsightTag`	Stellt die HTML- und JavaScript™-Vorlage dar, die verwendet wird, um einen Aufruf für den Digital Analytics for On Premises-Tag zu integrieren. Im Allgemeinen ist es nicht erforderlich, diese Option zu ändern.

Verwenden der Administratorseite in Web Connector

Der Web Connector enthält eine Administrationsseite, die einige Tools bereitstellt, die beim Verwalten und Testen der Konfiguration behilflich sind, die mit spezifischen URL-Mustern verwendet werden kann. Sie können die Administratorseite auch verwenden, um eine geänderte Konfiguration erneut zu laden.

Informationen zur Administratorseite

Sie können `http://`*`host`*`:`*`port`*`/interact/jsp/jsconnector.jsp` mit jedem unterstützten Web-Browser öffnen. Dabei bezeichnet host:port den Namen des Hosts, auf dem der Web

Connector ausgeführt wird, und den Port, der für das Listening von Verbindungen verwendet wird, z. B. `runtime.example.com:7001`

Es gibt mehrere Möglichkeiten, wie Sie die Administratorseite verwenden können:

Tabelle 41. Optionen für die Administratorseite im Web Connector

Verfügbare Optionen auf der Administratorseite im Web Connector

Option	Zweck
Konfiguration erneut laden	Klicken Sie auf den Link **Konfiguration erneut laden**, um alle auf dem Datenträger gespeicherten Konfigurationsänderungen erneut in den Speicher zu laden. Dies ist erforderlich, wenn Sie Änderungen direkt in der Web Connector-Konfigurationsdatei jsconnector.xml vorgenommen haben anstatt die Webseiten zur Konfiguration zu verwenden.
Konfiguration anzeigen	WebConnector-Konfiguration anzeigen, die dem URL-Muster im Feld **Konfiguration anzeigen** entspricht. Wenn Sie die URL einer Seite eingeben und auf **Konfiguration anzeigen** klicken, gibt der Web Connector die Konfiguration zurück, die das System aufgrund des zugeordneten Musters verwendet. Wird keine Übereinstimmung gefunden, wird die Standardkonfiguration zurückgegeben. Dies ist hilfreich, um zu testen, ob die richtige Konfiguration für eine bestimmte Seite verwendet wird.
Seitentag ausführen	Wenn Sie die Felder auf dieser Seite ausfüllen und dann auf **Seitentag ausführen** klicken, gibt der Web Connector das `pageTag`-Ergebnis zurück, das diesem URL-Muster entspricht. Dadurch wird der Aufruf eines Seitentags simuliert. Der Unterschied zwischen dem pageTag-Aufruf mit diesem Tool und der Verwendung einer echten Website liegt darin, dass alle Fehler oder Ausnahmen angezeigt werden, wenn Sie diese Administratorseite verwenden. Bei einer echten Website werden die

Tabelle 41. Optionen für die Administratorseite im Web Connector

Verfügbare Optionen auf der Administratorseite im Web Connector

(Fortsetzung)

Option	Zweck
	Ausnahmen nicht zurückgegeben, sondern nur in der Web Connector-Protokolldatei angezeigt.

Web Connector-Beispielseite

Zusammen mit dem Unica Interact Web Connector wird eine Beispieldatei mit dem Namen `WebConnectorTestPageSA.html` geliefert (im Verzeichnis `<Unica Interact_Home/jsconnector/webapp/html`), die demonstriert, wie viele Funktionen des Web Connectors in einer Seite markiert würden. Diese Seite wird auch im folgenden Beispiel dargestellt.

HTML-Beispielseite im 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">
//<![CDATA[
   var unicaWebConnectorBaseURL=
        "[CHANGE ME - http://host:port/<jsconnector>/pageTag]";
   var unicaURLData = "ok=Y";
   try {
    unicaURLData += "&url=" + escape(location.href)
   } catch (err) {}
   try {
    unicaURLData += "&title=" + escape(document.title)
   } catch (err) {}
   try {
    unicaURLData += "&referrer=" + escape(document.referrer)
   } catch (err) {}
   try {
    unicaURLData += "&cookie=" + escape(document.cookie)
   } catch (err) {}
   try {
    unicaURLData += "&browser=" + escape(navigator.userAgent)
```

```
  } catch (err) {}
  try {
   unicaURLData += "&screensize=" +
   escape(screen.width + "x" + screen.height)
  } catch (err) {}
  try {
   if (affiliateSitesForUnicaTag) {
       var unica_asv = "";
       document.write("<style id=\"unica_asht1\" type=\"text/css\"> "
    + "p#unica_ashtp a {border:1px #000000 solid; height:100px "
    + "!important;width:100px "
    + "!important; display:block !important; overflow:hidden "
    + "!important;} p#unica_ashtp a:visited {height:999px !important;"
    + "width:999px !important;} <\/style>");
       var unica_ase = document.getElementById("unica_asht1");
       for (var unica_as in affiliateSitesForUnicaTag) {
        var unica_asArr = affiliateSitesForUnicaTag[unica_as];
        var unica_ashbv = false;
        for (var unica_asIndex = 0; unica_asIndex <
    unica_asArr.length && unica_ashbv == false;
    unica_asIndex++)
  {
         var unica_asURL = unica_asArr[unica_asIndex];
         document.write("<p id=\"unica_ashtp\" style=\"position:absolute;
"
    + "top:0;left:-10000px;height:20px;width:20px;overflow:hidden; \
  margin:0;padding:0;visibility:visible;\"> \
  <a href=\"" + unica_asURL + "\">" + unica_as + " <\/a><\/p>");
         var unica_ae =
document.getElementById("unica_ashtp").childNodes[0];
         if (unica_ae.currentStyle) {
          if (parseFloat(unica_ae.currentStyle["width"]) > 900)
```

```
       unica_ashbv = true
          } else if (window.getComputedStyle) {
           if (parseFloat(document.defaultView.getComputedStyle
       (unica_ae, null).getPropertyValue("width")) > 900)
    unica_ashbv = true
          }
          unica_ae.parentNode.parentNode.removeChild(unica_ae.parentNode)
         }
         if (unica_ashbv == true) {
          unica_asv += (unica_asv == "" ? "" : ";") + unica_as
         }
        }
        unica_ase.parentNode.removeChild(unica_ase);
        unicaURLData += "&affiliates=" + escape(unica_asv)
    }
   } catch (err) {}
   document.write("<script language='javascript' "
     + " type='text/javascript' src='" + unicaWebConnectorBaseURL + "?"
  + unicaURLData + "'><\/script>");
  //]]&gt;
</script>
<style type="text/css">
/*<![CDATA[*/
   .unicainteractoffer {display:none !important;}
   /*]]&gt;*/
</style>
   <title>Sample Interact Web Connector Page</title>
   </head>
   <body>
   <!-- END: Interact Web Connector Page Tag -->
   <!--
```

```
##############################################################################
###
end of pageTag paste
##############################################################################
###
-->
   </body>
</html>
```

Kapitel 18. Integration von Unica Interact und Digital Recommendations

Unica Interact kann mit IBM Digital Recommendations integriert werden, um Unica Interact-basierte Produktempfehlungen bereitzustellen. Beide Produkte können Produktempfehlungen für Angebote liefern, jedoch unter Verwendung unterschiedlicher Methoden. Digital Recommendations verwendet das Webverhalten eines Besuchers (kollaborativer Filter), um Korrelationen zwischen Besuchern und empfohlenen Angeboten herzustellen. Unica Interact basiert auf dem bisherigen Verhalten, den Attributen und der Historie des Kunden und weniger auf Angeboten auf Anzeigeebene, wobei gelernt wird, welche Angebote am besten zum Verhaltensprofil eines Kunden passen (basierend auf demographischen und anderen Informationen über den Kunden). Angebotsannahmeraten helfen dabei, mithilfe von Selbstlernfunktionen ein Vorhersagemodell zu erstellen. Unica Interact nutzt die Vorteile beider Produkte, um so mithilfe eines persönlichen Profils Angebote zu definieren, die eine Kategorie-ID an Digital Recommendations übergeben. Anschließend werden Produktempfehlungen anhand der Beliebtheit (die „Weisheit der Vielen“) abgerufen, die dem Besucher als Teil des ausgewählten Angebots angezeigt werden. Auf diese Weise erhalten Kunden bessere Empfehlungen, die zu mehr Klickabfolgen und besseren Ergebnissen führen als die Verwendung nur eines der Produkte.

In den folgenden Abschnitten wird beschrieben, wie diese Integration funktioniert und wie Sie die bereitgestellte Beispielanwendung verwenden können, um eine benutzerdefinierte Angebotsintegration zu erstellen.

Übersicht über die Integration von Unica Interact mit Digital Recommendations

In diesem Abschnitt wird beschrieben, wie Unica Interact mit IBM Digital Recommendations integriert werden kann, um Unica Interact-basierte Produktempfehlungen bereitzustellen. Dies umfasst auch eine Beschreibung des Integrationsprozesses und der Mechanismen der Integration.

Die Integration von Unica Interact mit IBM Digital Recommendations erfolgt über eine REST-API (Representational State Transfer), die über die Digital Recommendations-Installation

verfügbar ist. Mithilfe der REST-API-Aufrufe mit der entsprechenden Kategorie-ID kann Unica Interact empfohlene Produkte abrufen und in die Angebotsinformationen auf der benutzerdefinierten Seite einbeziehen, die der Besucher anzeigt.

Wenn ein Besucher die URL der Webseite anzeigt (z. B. die Beispiel-JSP-Seite, die Bestandteil Ihrer Unica Interact-Installation ist), ruft die Seite Unica Interact auf, um ein Angebot abzurufen. Vorausgesetzt, dass das Angebot in Unica Interact mit den richtigen Parametern konfiguriert wurde, so treten im einfachsten Fall die folgenden Schritte ein:

1. Die Seitenlogik ermittelt die Kunden-ID des Besuchers.
2. Ein API-Aufruf an Unica Interact wird durchgeführt, der die erforderlichen Informationen übergibt, um ein Angebot für diesen Kunden zu erstellen.
3. Das zurückgegebene Angebot liefert die Webseite mit mindestens drei Attributen: die URL für das Bild des Angebots, die URL der Landing-Page, wenn der Kunde sich durchklickt, und die Kategorie-ID, um zu bestimmen, welche Produkte empfohlen werden.
4. Die Kategorie-ID wird dann für einen Digital Recommendations-Aufruf verwendet, um die empfohlenen Produkte abzurufen. Diese Produktmenge ist JSON-formatiert (JavaScript Object Notation) und nach bestverkauften Produkten in dieser Kategorie sortiert.
5. Das Angebot und die Produkte werden dann im Browser des Besuchers angezeigt.

Diese Integration ist nützlich, um Angebotsempfehlungen und Produktempfehlungen zu kombinieren. Zum Beispiel könnten Sie auf einer Webseite zwei Interaktionspunkte haben: einen für ein Angebot und einen für Empfehlungen, die zu diesem Angebot passen. Dazu führt die Webseite einen Aufruf an Unica Interact durch, um mithilfe einer Echtzeitsegmentierung das beste Angebot zu ermitteln (z. B. 10 % Rabatt auf alle Kleingeräte). Wenn die Seite das Angebot von Unica Interact erhält, enthält das Angebot die Kategorie-ID (in diesem Beispiel für Kleingeräte). Die Seite übergibt dann mithilfe eines API-Aufrufs die Kategorie-ID für Kleingeräte an Digital Recommendations und erhält als Antwort die besten Produktempfehlungen für diese Kategorie anhand der Beliebtheit.

In einem einfacheren Beispiel führt eine Webseite einen Aufruf an Unica Interact durch, nur um eine Kategorie herauszufinden (z. B. hochwertiges Besteck), die mit dem Kundenprofil

übereinstimmt. Anschließend übergibt die Seite die erhaltene Kategorie-ID an Digital Recommendations und erhält Produktempfehlungen für Besteck.

Voraussetzungen für die Integration

Bevor Sie die Integration von Digital Recommendations mit Unica Interact verwenden können, müssen Sie zunächst sicherstellen, dass die in diesem Abschnitt beschriebenen Voraussetzungen erfüllt sind.

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Sie sind mit der Verwendung der Unica Interact-API vertraut, die an einer anderen Stelle im Administratorhandbuch und in der Onlinehilfe dokumentiert ist.
- Sie sind mit der Digital Recommendations-REST-API vertraut, die in Ihrer Digital Recommendations-Dokumentation für Entwickler beschrieben ist.
- Sie verfügen über grundlegende Kenntnisse in HTML, JavaScript™, CSS und JSON (JavaScript™ Object Notation).

  JSON ist wichtig, weil die Digital Recommendations-REST-API die von Ihnen angeforderten Produktinformationen als Daten im JSON-Format zurückgibt.
- Sie sind mit serverseitiger Webseitencodierung vertraut, weil die mit Unica Interact bereitgestellte Demonstrationsanwendung JSP verwendet (JSP ist jedoch nicht erforderlich).
- Sie verfügen über ein gültiges Digital Recommendations-Konto und die Liste der Kategorie-IDs, die Unica Interact zum Abrufen von Produktempfehlungen verwenden soll (die am besten verkauften oder beliebtesten Produkte in der von Ihnen angegebenen Kategorie).
- Sie verfügen über den Link zur Digital Recommendations-REST-API (eine URL für Ihre Digital Recommendations-Umgebung).

  In der Beispielanwendung, die Bestandteil Ihrer Unica Interact-Installation ist, finden Sie ein Beispiel. Weitere Informationen finden Sie im Beispielcode unter Verwenden des Integrationsbeispielprojekts (auf Seite 686).

Konfigurieren eines Angebots mit Digital Recommendations-Integration

Bevor Ihre Webseite Digital Analytics Digital Recommendations aufrufen kann, um ein empfohlenes Produkt abzurufen, müssen Sie zunächst das Unica Interact-Angebot mit den erforderlichen Informationen konfigurieren, die an Digital Recommendations übermittelt werden sollen.

Um ein Angebot einzurichten, das mit Digital Recommendations verlinkt ist, müssen Sie zunächst darauf achten, dass die folgenden Bedingungen erfüllt sind:

- Stellen Sie sicher, dass Ihr Unica Interact-Laufzeitserver ordnungsgemäß eingerichtet ist und ausgeführt wird.
- Stellen Sie sicher, dass der Laufzeitserver eine Verbindung zum Digital Recommendations-Server aufbauen kann. Achten Sie auch darauf, dass Ihre Firewall ausgehende Standardwebverbindungen (Port 80) nicht verhindert.

Führen Sie die folgenden Schritte aus, um ein mit Digital Recommendations integriertes Angebot einzurichten.

1. Erstellen oder bearbeiten Sie ein Angebot für Unica Interact.

   Informationen zum Erstellen und Ändern von Angeboten finden Sie im Unica InteractBenutzerhandbuch und in der Unica Campaign-Dokumentation.
2. Achten Sie darauf, dass das Angebot neben den anderen enthaltenen Einstellungen die folgenden Angebotsattribute enthält:
   - Die URL (Uniform Resource Locator) mit dem Link zum Bild des Angebots.
   - Die URL mit dem Link zur Landing-Page für das Angebot.
   - Eine diesem Angebot zugeordnete Digital Recommendations-Kategorie-ID.

     Sie können die Kategorie-ID manuell aus Ihrer Digital Recommendations-Konfiguration abrufen. Unica Interact kann die Kategorie-ID-Werte nicht direkt abrufen.

   In der Demonstrationswebanwendung, die Bestandteil Ihrer Unica Interact-Installation ist, heißen diese Angebotsattribute `ImageURL`, `ClickThruURL`und `CategoryID`. Sie

können beliebige beschreibende Namen verwenden, solange Ihre Webanwendung mit den Werten übereinstimmt, die das Angebot erwartet.

Beispielsweise könnten Sie ein Angebot mit dem Namen "10PercentOff" definieren, das diese Attribute enthält, wobei die Kategorie-ID (wie aus Ihrer Digital Recommendations-Konfiguration abgerufen) `PROD1161127`, die URL des angeklickten Angebots `http://www.example.com/success`und die URL des für das Angebot anzuzeigenden Bildes `http://localhost:7001/sampleIO/img/10PercentOffer.jpg` ist (eine URL, die in diesem Fall lokal auf dem Unica Interact-Laufzeitserver liegt).

3. Definieren Sie die Verfahrensregeln für einen interaktiven Kanal, der dieses Angebot enthalten soll, und implementieren Sie den interaktiven Kanal wie gewohnt.

Das Angebot ist jetzt mit den Informationen definiert, die für die Integration mit Digital Recommendations erforderlich sind. Damit Digital Recommendations Unica Interact Produktempfehlungen bereitstellen kann, müssen Sie jetzt nur noch Ihre Webseiten so konfigurieren, dass sie die erforderlichen API-Aufrufe durchführen.

Wenn Sie Ihre Webanwendung so konfigurieren, dass sie Besuchern die integrierte Seite bereitstellt, müssen Sie darauf achten, dass die folgenden Dateien im Verzeichnis `WEB-INF/lib` enthalten sind:

- *`Interact_Home`*`/lib/interact_client.jar`, die zum Bearbeiten von Aufrufen von Ihrer Webseite an die Unica Interact-API erforderlich ist.
- *`Interact_Home`*`/lib/JSON4J_Apache.jar`, die zum Bearbeiten der Daten erforderlich ist, die vom Aufruf an die Digital Recommendations-REST-API zurückgegeben werden. Diese API gibt Daten im JSON-Format zurück.

Weitere Informationen zum Bereitstellen der Angebote für Ihre Kunden finden Sie unter Verwenden des Integrationsbeispielprojekts (auf Seite 686).

Verwenden des Integrationsbeispielprojekts

Jede Unica Interact-Laufzeitinstallation beinhaltet ein Beispielprojekt, das den Prozess der Integration von Digital Recommendations mit Unica Interact demonstriert. Das

Beispielprojekt stellt eine vollständige End-to-End-Demonstration zur Verfügung, wie Sie eine Webseite erstellen, die ein Angebot aufruft, das eine Kategorie-ID enthält. Diese Kategorie-ID wird dann an Digital Recommendations übergeben, um eine Liste mit empfohlenen Produkten abzurufen, die an den Interaktionspunkten der Seite dargestellt wird.

Übersicht

Sie können das enthaltene Beispielprojekt ohne Änderungen verwenden, wenn Sie den Integrationsprozess testen möchten. Sie können es auch als Ausgangspunkt verwenden, um eigene benutzerdefinierte Seiten zu entwickeln. Sie finden das Beispielprojekt in der folgenden Datei:

Interact_home`/samples/IntelligentOfferIntegration/MySampleStore.jsp`

Diese Datei enthält neben einem vollständigen, funktionsfähigen Beispiel des Integrationsprozesses auch umfassende Anmerkungen, die erklären, was Sie in Unica Interact einrichten müssen, was Sie in der `.jsp`-Datei anpassen müssen und wie Sie die Seite ordnungsgemäß implementieren, damit sie mit Ihrer Installation ausgeführt wird.

MySampleStore.jsp

Zur Vereinfachung ist die Datei „MySampleStore.jsp“ hier dargestellt. Dieses Beispiel wird in nachfolgenden Releases von Unica Interact u. U. aktualisiert. Verwenden Sie daher als Ausgangspunkt für alle Beispiele, die Sie möglicherweise benötigen, die Datei, die Bestandteil Ihrer Installation ist.

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page import="java.net.URL,
    java.net.URLConnection,
    java.io.InputStreamReader,
    java.io.BufferedReader,
    com.unicacorp.interact.api.*,
    com.unicacorp.interact.api.jsoverhttp.*,
    org.apache.commons.json.JSONObject,
    org.apache.commons.json.JSONArray" %>
```

```
<%

 /
*************************************************************************
******
 * 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
 (communication 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,audienceLe
vel,

 audienceColumnName,ip,customerId,debug,relyOnExistingSession,interactError
Msg);

 // 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:functi
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 %>"><img src="<%=offerImgURL %>"
 height="170"
          width="695" border="0"></a>
  </div>
 </div>


<% } else { %>
 No offer available.. <br> <br>
 <%=interactErrorMsg.toString() %>
<% } %>

<% if(products != null) { %>
 <!-- IntelligentOffer Products HTML -->
 <br><br><br> <br><br><br> <br><br><br> <br><br><br> <br>
 <div class="unicacarousel">
 <div class="unicacarousel_sizer">
  <ul class="unicacarousel_rotater">


 <% JSONArray recs = products.getJSONObject("io").getJSONArray("recs");
  if(recs != null)
  {
```

```
   for(int x=0;x< recs.length();x++)
   {
    JSONObject rec = recs.getJSONObject(x);
    if(rec.getString("Product Page") != null &&
              rec.getString("Product Page").trim().length()>0) {
    %>

    <li>
     <a href="<%=rec.getString("Product Page") %>"
 title="<%=rec.getString("Product Name") %>">
             <img src="<%=rec.getString("Image") %>"  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>
<% } %>

 </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());
    }
%>
```

Kapitel 19. Integration von Unica Interact und Digital Data Exchange

Mit Digital Data Exchange kann Ihre Website eine Verbindung zu Unica Interact herstellen, um eine leistungsfähige Omni-Channel-Ausführungsengine bereitzustellen, die die besten Angebote an die optimalen Kanäle übergibt und aus dem Angebotsfeedback lernt, um die Effektivität des Marketings kontinuierlich zu erhöhen.

Sie können dieses Tool verwenden, wenn Ihr Marketing-Team Unica Interact zur Verwaltung von Omni-Channel-Angeboten verwendet und diese individuell gestalteten Intelligent Offers auf Ihre Websites erweitern möchte.

IBM Digital Data Exchange integriert Marketinglösungen von und anderen Anbietern mit digitalen Kundeninformationen über eine API für die Syndikation von Echtzeitdaten und eine unternehmensweite Tagmanagementlösung.

Ohne IBM Digital Data Exchange sind die Anbieter beim Verbinden von Unica Interact mit ihrer Webseite und beim Aufrufen der Unica Interact-API von verschiedenen Webseiten von der IT abhängig. Mit IBM Digital Data Exchange können Anbieter die IT umgehen und eine direkt Verbindung zu IBM Digital Data Exchange herstellen, um IBM Digital Data Exchange-Tags auf verschiedenen Webseiten einzuschließen.

Voraussetzungen

Bevor Sie die Integration von Unica Interact mit Digital Data Exchange verwenden können, müssen Sie zunächst sicherstellen, dass die in diesem Abschnitt beschriebenen Voraussetzungen erfüllt sind.

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind.

- Sie sind mit der Unica Interact-JavaScript-API vertraut, die an einer anderen Stelle im Administratorhandbuch und in der Onlinehilfe dokumentiert ist.
- Sie sind mit Digital Data Exchange-Tagging und Seitengruppen vertraut.
- Sie verfügen über ein gültiges Digital Data Exchange-Konto.
- Die Datei `interactapi.js` wird öffentlich per Hosting bereitgestellt, damit auf sie in den Einstellungen **Anbieter** zugegriffen werden kann.

Unica Interact mit IBM Digital Data Exchange in Website integrieren

Anhand der folgenden Schritte können Sie Unica Interact unter Verwendung von Digital Data Exchange in Ihre Website integrieren.

1. Geben Sie den Speicherort der `Interactapi.js`-Datei an.
    a. Navigieren Sie zu **Anbieter > Anbietereinstellungen** in Digital Data Exchange.
    b. Wählen Sie Unica Interact in der Dropdown-Liste **Anbieter** aus.
    c. Geben Sie unter **Bibliothekspfad** die URL ein, unter der Sie die `Interactapi.js` gehostet haben. Schließen Sie in diese URL nicht das Protokoll (http oder https) ein.
    d. Fügen Sie unter **Pfad zu öffentlichem REST-Servlet** den Pfad zum REST-Servlet hinzu.
2. Navigieren Sie in Digital Data Exchange zu **Verwalten > Globale Einstellungen**, um den Objektnamen anzugeben, der als Seitenkennung unter **Eindeutige Seitenkennung** verwendet werden soll. Als Objektnamen können Sie zum Beispiel `digitalData.pageInstanceID` festlegen.
3. Fügen Sie die `eluminate.js`-Datei und eine Kennung auf der Webseite ein, auf der Digital Data Exchange die Tags einfügen soll. Sie müssen für jede Webseite eine eindeutige Kennung festlegen, damit die verschiedenen Seiten von Digital Data Exchange unterschieden werden können.

    Sie können zum Beispiel das folgende Script zur Homepage hinzufügen.

```
<!-- 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. Erstellen Sie in Digital Data Exchange Tags, Codesegmente, Funktionen und weitere Elemente, die Sie auf der Webseite hinzufügen möchten.
5. Erstellen Sie Seitengruppen, um zu definieren, was auf jeder Seite abgelegt werden soll.

Unica Interact-Tags in Digital Data Exchange

Mit den Digital Data Exchange-Standardtags können Sie Variationen der Tags definieren, die für Webseiten geeignet sind, auf denen Daten von anderen Standorten dargestellt werden. Sobald sie definiert sind, werden diese Tags zur Unica Interact-Tagliste hinzugefügt. Tags müssen nicht Felder aufweisen, die definiert werden müssen, müssen nicht über erforderliche Tagfelder verfügen und können direkt verwendet werden.

Die folgenden Unica Interact-Tags sind in Digital Data Exchange unter **Tags**verfügbar.

- Sitzung beenden
- Angebot abrufen
- Bibliothek laden
- Ereignis senden
- Zielgruppe festlegen
- Sitzung starten

Wenn Sie Unica Interact-Tags verwenden möchten, bearbeiten Sie die Tags, um Tagfeld, Methode, Objektname, Datentyp und Änderungswert für jeden Unica Interact-Tag zu definieren.

Für die Tags "Ereignis senden", "Zielgruppe festlegen" und "Sitzung starten" sind benutzerdefinierte Tagfelder zulässig. Verwenden Sie das Symbol zum Hinzufügen von Tagfeldern und klicken Sie auf das Symbol "Bearbeiten", um den angepassten Parameter zu definieren. Der Prozess ist zwar derselbe wie bei jeder Parameterdefinition, der Name des

Parameters kann jedoch bearbeitet werden und muss aus dem Parameternamen, einem Doppelpunkt und dem Datentyp des Parameters bestehen. Die Reihenfolge der angepassten Parameter im Tag kann mithilfe der Abwärts- und Aufwärtspfeile geändert werden.

Die Tags können auch an JavaScript-Funktionen oder HTML-Objekte gebunden werden, sodass sie nach dem Auslösen der Funktion oder einem HTML-Objektereignis ausgelöst werden.

Sitzung beenden

Der Tag "Sitzung beenden" markiert das Ende einer Websitzung.

Für den Tag "Sitzung beenden" stehen die folgenden Tagfelder zur Verfügung.

Tabelle 42. Tags "Sitzung beenden"

Tags "Sitzung beenden"

Tagfeld	Syntax
*Sitzungs-ID	Gibt die Sitzungs-ID an.
Name der Callback-Funktion bei Erfolg	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Beenden der Sitzung erfolgreich ist.
Name der Callback-Funktion bei Fehler	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Beenden der Sitzung nicht erfolgreich ist.

Alle **Tagfelder** mit der Markierung ***** sind erforderlich.

Angebot abrufen

Mit dem Tag "Angebot erhalten" können Sie Angebote vom Laufzeitserver anfordern.

Für den Tag "Angebot erhalten" stehen die folgenden Tagfelder zur Verfügung.

Tabelle 43. Tags "Angebot erhalten"

Tags "Angebot erhalten"

Tagfeld	Syntax
*Sitzungs-ID	Gibt die Sitzungs-ID an.
*Name des Interaktionspunkts	Gibt den Namen des Interaktionspunkts an, auf den von dieser Methode verwiesen wird. Dieser Name muss exakt mit dem Namen des im interaktiven Kanal definierten Interaktionspunkts übereinstimmen.
*Erforderliche Anzahl	Gibt die Anzahl der erforderlichen Angebote an.
Name der Callback-Funktion bei Erfolg	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Abrufen der Angebote erfolgreich ist.
Name der Callback-Funktion bei Fehler	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Abrufen der Angebote nicht erfolgreich ist.

Alle **Tagfelder** mit der Markierung ***** sind erforderlich.

Der Tag "Angebot erhalten" muss einer Seitengruppe zugeordnet sein, für deren Container `Default` eingestellt ist.

Bibliothek laden

Bei Verwendung des Tags "Bibliothek laden" wird die Unica Interact-JavaScript-Bibliothek in den Anfangsabschnitt der Seite geladen.

Für den Tag "Bibliothek laden" werden keine Parameter bereitgestellt. Die Position der Bibliothek wird aus `Bibliothekspfad` unter **Anbietereinstellungen** abgerufen. Sie muss in einer Seitengruppe eingeschlossen sein, in der ein Container verwendet wird, für den `Head`

eingestellt ist; außerdem muss sie auf jeder Seite ausgeführt werden, auf der das Unica Interact-Tagging angewendet wird.

Wichtig: Keiner der anderen Tags funktioniert, wenn der Tag "Bibliothek laden" nicht eingeschlossen ist. Die Datei interact.js wird nicht geladen, wenn dieser Tag nicht eingeschlossen ist.

Ereignis senden

Mit dem Tag "Ereignis senden" kann jedes Ereignis ausgeführt werden, das im interaktiven Kanal definiert ist.

Für den Tag "Ereignis senden" stehen die folgenden Tagfelder zur Verfügung.

Tabelle 44. Tags "Ereignis senden"

Tags "Ereignis senden"

Tagfeld	Syntax
*Sitzungs-ID	Gibt die Sitzungs-ID an.
*Ereignisname	Gibt den Namen des Ereignisses an. Der Name des Ereignisses muss mit dem im interaktiven Kanal definierten Ereignisnamen übereinstimmen. Bei diesem Namen braucht die Groß-/Kleinschreibung nicht berücksichtigt zu werden.
Name der Callback-Funktion bei Erfolg	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Senden des Ereignisses erfolgreich ist.
Name der Callback-Funktion bei Fehler	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Senden des Ereignisses nicht erfolgreich ist.

Alle **Tagfelder** mit der Markierung * sind erforderlich.

Optionale Parameter können mit der Funktion für benutzerdefinierte Tagfelder hinzugefügt werden. Benutzerdefinierte Tagnamen müssen aus dem Parameternamen, einem Doppelpunkt und dem Datentyp bestehen.

Zielgruppe festlegen

Mit dem Tag "Zielgruppe festlegen" können die Zielgruppen-ID und die Ebene für einen Besucher festgelegt werden.

Für den Tag "Zielgruppe festlegen" stehen die folgenden Tagfelder zur Verfügung.

Tabelle 45. Tags "Zielgruppe festlegen"

Tags "Zielgruppe festlegen"

Tagfeld	Syntax
*Sitzungs-ID	Gibt die Sitzungs-ID an.
*Zielgruppen-ID	Gibt die Zielgruppen-ID an. Die Namen müssen mit den physischen Spaltennamen aller Tabellen übereinstimmen, in denen die Zielgruppen-ID enthalten ist. Die Zielgruppen-ID darf nicht mehr als 17 signifikante Ziffern enthalten. Wenn eine Zielgruppen-ID mehr als 17 signifikante Ziffern enthält, muss sie untergliedert oder in eine Zeichenfolge umgewandelt werden.
*Zielgruppenebene	Definiert die Zielgruppenebene.
Name der Callback-Funktion bei Erfolg	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Festlegen der Zielgruppe erfolgreich ist.
Name der Callback-Funktion bei Fehler	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Festlegen der Zielgruppe nicht erfolgreich ist.

Alle **Tagfelder** mit der Markierung ***** sind erforderlich.

Optionale Parameter können mit der Funktion für benutzerdefinierte Tagfelder hinzugefügt werden. Benutzerdefinierte Tagnamen müssen aus dem Parameternamen, einem Doppelpunkt und dem Datentyp bestehen.

Sitzung starten

Mit dem Tag "Sitzung starten" wird eine Websitzung erstellt und definiert.

Für den Tag "Sitzung starten" stehen die folgenden Tagfelder zur Verfügung.

Tabelle 46. Tags "Sitzung starten"

Tags "Sitzung starten"

Tagfeld	Syntax
*Sitzungs-ID	Gibt die Sitzungs-ID an.
*Interact-Kanal	Definiert den Namen des interaktiven Kanals, auf den diese Sitzung verweist. Dieser Name muss exakt mit dem in Campaign definierten Namen des interaktiven Kanals übereinstimmen.
*Zielgruppen-ID	Gibt die Zielgruppen-ID an. Die Namen müssen mit den physischen Spaltennamen aller Tabellen übereinstimmen, in denen die Zielgruppen-ID enthalten ist.
*Zielgruppenebene	Definiert die Zielgruppenebene.
*Vorhandene Sitzung als Grundlage	Definiert, ob diese Sitzung eine neue oder eine vorhandene Sitzung verwendet.
*Debug	Aktiviert oder inaktiviert die Daten zur Fehlerbehebung.

Tabelle 46. Tags "Sitzung starten"

Tags "Sitzung starten"

(Fortsetzung)

Tagfeld	Syntax
Name der Callback-Funktion bei Erfolg	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Starten der Sitzung erfolgreich ist.
Name der Callback-Funktion bei Fehler	Definiert den Namen der Funktion, die aufgerufen werden soll, wenn die Methode zum Starten der Sitzung nicht erfolgreich ist.

Alle **Tagfelder** mit der Markierung ***** sind erforderlich.

Optionale Parameter können mit der Funktion für benutzerdefinierte Tagfelder hinzugefügt werden. Benutzerdefinierte Tagnamen müssen aus dem Parameternamen, einem Doppelpunkt und dem Datentyp bestehen.

Das "Sitzung starten"-Tag sollte einer Seitengruppe zugewiesen werden, deren Container auf `Standard` gesetzt ist.

Beispiel für Tageinstellungen

An diesem Beispiel wird eine einfache Konfiguration der Einstellungen der Tags "Sitzung starten", "Ereignis senden", "Angebot erhalten" und "Sitzung beenden" veranschaulicht.

Für jeden Tag können Sie die Werte der Tagfelder über das Cookie mit der Cookiemethode oder über das JavaScript-Objekt mit der Methode "javascriptobject" abrufen.

Diese Tags unterstützen weitere Parameter, die in diesem Beispiel nicht dargestellt werden.

Beispieltageinstellungen für "Sitzung starten"

Klicken Sie auf**Tags > IBM Tags > Interact > Typ: Starten Sie die Sitzung**, um ein Start-Session-Tag zu erstellen. Bearbeiten Sie den Tag mit den folgenden Einstellungen.

Einstellungen für Sitzungs-ID

- **Methode**: `Konstante`
- **Konstante**: `5555`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Interaktiver Kanal"

- **Methode**: `Konstante`
- **Konstante**: `WSCDemo`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Zielgruppen-ID"

- **Methode**: `Konstante`
- **Konstante**: `USERS_ID,2002,numeric`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Zielgruppenebene"

- **Methode**: `Konstante`
- **Konstante**: `WSCUserId`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Vorhandene Sitzung als Grundlage"

- **Methode**: `Konstante`
- **Konstante**: `False`
- **Datentyp**: `Boolesch`
- **Modifikator**: `<null>`

Debug

- **Methode**: `Konstante`
- **Konstante**: `True`
- **Datentyp**: `Boolesch`
- **Modifikator**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Erfolg"

- **Methode**: `Nicht zugewiesen`
- **Wert**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Fehler"

- **Methode**: `Nicht zugewiesen`
- **Wert**:`<null>`

Beispieltageinstellungen für "Angebot erhalten"

Klicken Sie auf**Tags > IBM Tags > Interact > Typ:** Angebote abrufen, um ein "Angebot erhalten"-Tag zu erstellen Bearbeiten Sie den Tag mit den folgenden Einstellungen.

Einstellungen für Sitzungs-ID

- **Methode**: `Konstante`
- **Konstante**: `5555`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Name des Interaktionspunkts"

- **Methode**: `Konstante`
- **Konstante**: `AuroraHomepageHeaderBannerLeft`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Erforderliche Anzahl"

- **Methode**: `Konstante`
- **Konstante**: `1`
- **Datentyp**: `integer`
- **Modifikator**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Erfolg"

- **Methode**: `Konstante`
- **Konstante**: `onOfferReturnSuccess`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Fehler"

- **Methode**: `Konstante`
- **Konstante**: `onOfferReturnError`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Beispieltageinstellungen für "Ereignis senden"

Klicken Sie auf**Tags > IBM Tags > Interact > Typ:** Ereignis senden, um ein "Ereignis senden"-Tag zu erstellen. Bearbeiten Sie den Tag mit den folgenden Einstellungen.

Einstellungen für Sitzungs-ID

- **Methode**: `Konstante`
- **Konstante**: `5555`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Ereignisname"

- **Methode**: `Konstante`
- **Konstante**: `ACCEPTOFFER`

- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Erfolg"

- **Methode**: `Konstante`
- **Konstante**: `onSuccessTestFunction`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Fehler"

- **Methode**: `Konstante`
- **Konstante**: `onErrorTestFunction`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für zusätzliches Parameterfeld

- **Tagfeld**: `UACIOfferTrackingCode:string`
- **Methode**: `JavaScriptObject`
- **Objektname**: `oa.treatmentCode`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Beispieltageinstellungen für "Sitzung beenden"

Klicken Sie auf**Tags > IBM Tags > Interact > Typ:** Sitzung beenden, um ein "Sitzung beenden" -Tag zu erstellen. Bearbeiten Sie den Tag mit den folgenden Einstellungen.

Einstellungen für Sitzungs-ID

- **Methode**: `Konstante`
- **Konstante**: `5555`
- **Datentyp**: `Zeichenfolge`
- **Modifikator**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Erfolg"

- **Methode**: `Nicht zugewiesen`
- **Wert**: `<null>`

Einstellungen für "Name der Callback-Funktion bei Fehler"

- **Methode**: `Nicht zugewiesen`
- **Wert**: `<null>`

Beispielfunktionen

Für die Funktionen, die für die Einstellungen von "Name der Callback-Funktion bei Erfolg" und "Name der Callback-Funktion bei Fehler" verwendet werden, müssen Sie nur den Funktionsnamen angeben, wenn Sie einen neuen Tag erstellen, falls die Funktion bereits auf der Webseite vorhanden ist.

Sie können auch die Digital Data Exchange-Dienstprogramme verwenden, um Funktionen zu erstellen und zu den Webseiten hinzuzufügen.

Im folgenden Beispiel wird dargestellt, wie ein Angebot angezeigt werden soll, das von Unica Interact auf der Webseite zurückgegeben wird. Sie müssen dieses Script auf der Seite einschließen oder den Codeausschnitt aus Digital Data Exchange verwenden, um es einzufügen.

```
<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>
```

Integrationskonfiguration überprüfen

Mit dem Digital Data Exchange-Testtool und der Datei `Interact.log` können Konfigurationsproblem behoben werden.

Mit dem Digital Data Exchange-Testtool können Sie die Enzyklopädie überprüfen, um festzustellen, ob die Konfiguration wie erwartet funktioniert. Klicken Sie zum Öffnen des Testtools auf **Bereitstellung > Testtool** in Digital Data Exchange.

Sie können die Datei `Interact.log` anzeigen, um die Details der verschiedenen Unica Interact-API-Aufrufe zu überprüfen, die ausgeführt wurden. Fügen Sie die Callback-Funktion

bei Erfolg und die Callback-Funktion bei Fehler zu jedem Tag hinzu, um ein Debugging für die unterschiedlichen Aufrufe auszuführen.

Chapter 20. Integration von Unica Interact und Unica Journey

Unica Interact kann in Journey integriert werden, sodass eine kontinuierliche Kommunikation mit Benutzern basierend auf den Eingaben von Interact aufgebaut werden kann. Die Interact-Segmente oder -Zielgruppeninformationen können zu Journey übertragen werden und ermöglichen so einen kontinuierlichen Kundendialog. In der Interact-Anwendung wird die neue Kapazität hinzugefügt, um die Zielgruppeninformationen für Journey zu veröffentlichen. Diese Option wird über eine Journey-Konfiguration in ausgelösten Nachrichten auf ausgehender Kanalebene aktiviert, indem Sie die Möglichkeit bietet, Interact-Felder Journey-Feldern und einem neuen ausgehenden Kafka-Kanal mit Journey zuzuordnen.

In den folgenden Abschnitten wird beschrieben, wie diese Integration funktioniert.

Übersicht

Bei ausgelösten Nachrichten kann Interact die Journey Funktionen zur ständigen Kommunikation mit Benutzern verwenden. Mit den ausgelösten Nachrichten können die Administratoren die Ereignisse/Ereignismuster zusammen mit anderen Bedingungen definieren, nachdem den Benutzern ein Angebot unterbreitet wurde. Mit Interact kann die Zuordnung der Journey Felder zu den Zielgruppenfeldern und den Angebotsattributen ermöglicht werden. Die Interact Laufzeit kann eine Kommunikation-Journey mit Benutzern auslösen, indem die Zielgruppendetails über den ausgehenden Kafka-Kanal, der auf dem Bildschirm 'Ausgelöste Nachrichten' ausgewählt wurde, an das Journey System weitergegeben werden.

- Die Feldzuordnung von Interact und Journey kann auf der Registerkarte 'Gateway' im interaktiven Kanal definiert werden, wenn ein Gateway vom Typ 'Journey Outbound' ausgewählt wird. Die konfigurationsbasierte Zuordnung wurde entfernt und wird nicht mehr unterstützt.
- Der von Benutzern konfigurierte ausgehende Kanal steht in der Liste Ausgelöste Nachrichten Kanal zur Auswahl.

- Die Feldzuordnungdetails von Interact und Journey für alle Kanäle werden durch die Bereitstellung des interaktiven Kanals an Interact Laufzeit übertragen.
- Die Kafka-Verbindungsdetails müssen über die Gateway-Konfigurationsparameter in der Interact Laufzeit konfiguriert werden.
- Sollte das Ereignis oder Ereignismuster mit der postEvent Interact API übereinstimmen und die anderen Bedingungen der ausgelösten Nachrichten ebenfalls erfüllt sind, löst das System die ausgehende Nachricht mit den Zielgruppenfeldern gemäß der Feldzuordnung von Interact und Journey über den Journey Kafka Kanal aus.

Die Feldzuordnung von Interact und Journey

Interact stellt einen Mechanismus bereit, um die Felder von Interact und Journey zuzuordnen und die zu sendenden Informationen zu identifizieren. Bei der Auswahl von 'Ausgehende Journey', können die Feldzuordnungen von Journey auf dem Reiter 'Gateway' eines interaktiven Kanals definiert werden. Bei der Auswahl von 'Ausgehende Journey', können diese Feldzuordnungen von Journey auf dem Reiter 'Gateway' eines interaktiven Kanals definiert werden. Die konfigurationsbasierte Zuordnung wurde entfernt und wird nicht mehr unterstützt.

Details von Journey (vor Version 12.1.0.3)

Unter outboundChannels ist eine Kategorie namens „Journey“ verfügbar, die die Zuordnungsinformationen von Interact Journey definiert.

Die Kategorie Journey umfasst die folgenden Parameter:

- Neuer Kategoriename: Name der Konfiguration.
- Name: Der Name des ausgehenden Kanals, der in der Liste 'Ausgelöste Nachrichten' angezeigt wird.
- EntrySourceCode: Dieses Feld ist ein Journey-spezifisches Pflichtfeld, das die Quelle der eingehenden Daten identifiziert und eine oder mehrere Journeys auslöst. EntrySourceCode ist ein Teil der ausgehenden Nachricht von Interact an Journey.

- DataDefinition: Name der Datendefinition für diese Zuordnung. Dieses Feld dient nur zu Informationszwecken in Interact, damit die Benutzer die Feldzuordnungen identifizieren können. Das Feld wird in Interact zur Nachrichtenverarbeitung nicht verwendet.

Feldzuordnung

Sobald die Basisdetails von Journey gespeichert werden, wird unter Journey eine neue Kategorie namens "FieldMapping" angezeigt.

Die Kategorie FieldMapping repliziert die Datendefinition in Journey. Die Kategorie ermöglicht es, ein Journey Feld gleich dem Datendefinitionsfeld in Journey zu definieren.

Neuer Kategoriename: Der Feldname von Interact, der diesem Journey Feld zugeordnet ist. Das Format des Feldnamens von Interact beträgt "Präfix.Feldname"; bei Interact wird die Groß-/Kleinschreibung nicht beachtet. Bei den Feldnamen in Journey, muss die Groß- und Kleinschreibung beachtet werden. Stellen Sie daher sicher, dass die entsprechende Groß-/Kleinschreibung für die Feldzuordnung verwendet wird. Das Präfix kann zwei mögliche Werte haben.

- **ANGEBOT** – Wird das Feld Journey einem Angebotsattribut zugeordnet, muss dem Feldnamen/Attributnamen das Präfix "ANGEBOT" vorangestellt werden.
- **PROFILE** – Wird das Feld Journey einem Profilattribut zugeordnet, muss dem Feldnamen von Interact das Präfix "PROFILE" vorangestellt werden.

Note: Die Echtzeitattribute und Sitzungsparameter müssen ebenfalls als Profilattribute betrachtet werden und müssen mit dem Präfix "PROFILE" vorangestellt werden.

Den Feldern/Parametern/Attributen von Interact, die nicht angebotsspezifisch sind, muss das Präfix "PROFILE" vorangestellt werden.

FieldName : Der Feldname von Journey, dem der Wert des oben erwähnten Felds von Interact zugeordnet und durch eine ausgehende Nachricht an Journey gesendet wird. Der hier erwähnte Feldname von Journey ist Teil der ausgehenden Nachrichten.

DataType: Es stehen drei Datentypen zur Auswahl, gleich wie bei der Datendefinition von Journey.

- String
- Numerisch
- Datum und Uhrzeit

Der Datentyp wird zur Validierung und Formattierung vom Interact-Feldwert verwendet, bevor er an Journey gesendet wird.

DefaultValue: Wird der Interact-Feldwert nicht verfügbar, weist das System dem Journey Feld den Standardwert zu.

Pflichtfeld: Die Interact Laufzeit überprüft, ob das Journey Feld als Pflichtfeld definiert ist. Wird das Feld als Pflichtfeld definiert und der entsprechende Interact-Feldwert nicht verfügbar ist, wird ein Fehler protokolliert und die Nachricht verworfen.

Note: Wird defaultValue für ein solches Feld definiert, verwendet das System den Standardwert, anstatt die ausgehende Nachricht zu verwerfen.

DatetimeFormat: Diese Eigenschaft gilt nur für den Datentyp datetime. Das ist ein datetime Format, das den Interact-Feldwert formatiert.

MaxLength: Diese Eigenschaft gilt nur für den Datentyp String. Dies bietet eine zusätzliche Überprüfung des Feldwerts. Wird MaxLength definiert und der entsprechende Interact-Feldwert MaxLength überschreitet, wird ein Fehler protokolliert und die Nachricht verworfen.

Die Benutzer müssen für jedes Feld von Journey eine Feldzuordnung konfigurieren, die ein Teil der ausgehenden Nachricht an Journey ist.

Sie können die Feldzuordnungen von Interact und Journey mithilfe der Optionen von ausgehenden Journey definieren. Diese sind auf dem Reiter Gateway im interaktiven Kanal verfügbar.

- Es ermöglicht den Vermarktern, die vorhandene Datendefinition von Journey auszuwählen.
- Es ermöglicht den Benutzern, die Eingabequellen auszuwählen, die in Journeys vom Typ Unica Interact vorhanden sind.
- Nachdem die Eingabequelle und die Datendefinition ausgewählt und die Schaltfläche Abrufen angeklickt wurden, werden die Details der ausgewählten und unter den Journey Feldern ausgefüllten Datendefinition angezeigt. Dies enthält Details zu Journey Feldern wie z.B. Name, Datentyp, Pflichtfeld / wichtiges Feld für Journeys.
- Für das Journey Feld können die Benutzer die Interact Felder zuordnen. Dies können entweder das Profilfeld, RTA oder das Angebotsattribut aus Interact sein.
- Sollte das Journey Feld ein Pflichtfeld sein, ist der Standardwert erforderlich. Falls der Wert aus den Interact-Feldern zur Laufzeit nicht abgerufen werden kann, wird der Standardwert verwendet und an Journey gesendet.

Ausgelöste Nachrichten

Der Bildschirm Ausgelöste Nachrichten listet die verfügbaren ausgehenden Kanäle zur Auswahl auf. Die ausgehenden Journey-Kanäle sind auch in der Liste der Kanäle für ausgelöste Nachrichten sichtbar.

Interact-Laufzeitkonfigurationen

-Gateway

Alle Journey-spezifischen Konfigurationen für die Kafka-Verbindungsdetails sind unter `Affinium|interact|triggeredMessage|gateways` konfiguriert. Unter Gateways steht eine neue Vorlage mit dem Namen "Journey" zur Verfügung, die verwendet werden muss, wenn die Informationen des ausgehenden Kanals an das Journey-System gesendet werden müssen. Im Folgenden sind die obligatorischen Konfigurationen aufgeführt, die erforderlich sind, damit das ausgehende Journey-Gateway funktioniert.

- Kafka-Verbindungsdetails: Die Kafka-Verbindungdetails müssen als Parameter für den Journey-Gateway konfiguriert werden. Die Parameter müssen alle erforderlichen Details enthalten, wie z. B. die Kafka-Broker-URL, das Kafka-Topic, Authentifizierungsparameter usw., um die Interact-Laufzeit so zu aktivieren, dass

ausgehende Nachrichten erfolgreich an Journey gesendet werden können. Um die für die Konfiguration erforderlichen-Parameter zu finden, lesen Sie den Abschnitt Unica Interact Laufzeitumgebung - Konfigurationseigenschaften *(on page 420)*, der Details zum Erstellen von Kafka Producer enthält. Im Folgenden sind die Beispielparameter aufgeführt, die Sie im Falle von 'Authentifizierung=keine' festlegen müssen.

- providerURL: Kafka_server_ip:port
- Thema: Kafka_topicname
- Authentifizierung: Kafka-Authentifizierungsmodus
- validationTimeoutMillis: Dies wird zum Zeitpunkt der Validierung der ausgehenden Nachrichten verwendet. Sie können jeden geeigneten Wert in Millisekunden entsprechend Ihrer Anforderung festlegen.
- deliveryTimeoutMillis: Dies wird zum Zeitpunkt des Versendens der Nachricht verwendet. Sie können jeden geeigneten Wert in Millisekunden entsprechend Ihrer Anforderung festlegen.

Implementierung

Die Reisekonfiguration für die Zuordnungsinformationen der Interact-Journey-Felder ist Teil des Bereitstellungspakets für interaktive Kanäle und wird bei der Bereitstellung des "interaktiven Kanals" an Interact Runtime übertragen.

Chapter 21. Integration von Unica Interact und Unica Deliver

Unica Deliver ist eine webbasierte, unternehmensweite Lösung für Marketingnachrichten, mit der Sie ausgehende Massen-Nachrichten wie E-Mail, SMS, Push und WhatsApp sowie Kampagnen von Transaktionsnachrichten durchführen können.

Interact verfügt über umfangreiche Funktionen, um den Benutzern in Echtzeit die bestmöglichen Angebote zu unterbreiten. Unica Interact kann mit Deliver integriert werden, sodass eine kontinuierliche Kommunikation mit Benutzern abhängig von den Angaben von Interact aufgebaut werden kann. Die Zielgruppeninformationen können von Interact an Deliver übertragen werden; dadurch wird eine kontinuierliche Kundenkommunikation ermöglicht. Die Integration von Interact und Deliver zielt darauf ab, die ausgehenden Funktionen von Deliver zu nutzen, um einen Benutzer mit den entsprechenden Angeboten abhängig von seinen Ereignissen und Aktivitäten zu konfigurieren und zu kontaktieren.

Die Integration von Deliver und Interact wird über die Registerkarte 'Gateway' auf der Benutzeroberfläche des interaktiven Kanals aktiviert. Dies bietet die Möglichkeit, der Deliver-Kommunikationstyp zusammen mit den relevanten Deliver-Vorlagen auszuwählen und die Interact Felder den Deliver Feldern zuzuordnen.

Übersicht

Die Deliver Funktionen können von Interact verwendet werden, um über die ausgelösten Nachrichten ständig mit den Benutzern zu kommunizieren. Mit den ausgelösten Nachrichten können die Administratoren die Ereignisse oder Ereignismuster zusammen mit anderen Bedingungen für ein den Benutzern zu unterbreitendes Angebot definieren.

Interact ermöglicht es den Administratoren, den Nachrichtentyp und die erforderliche Deliver-Vorlage auf der Registerkarte 'Quelle' auszuwählen. Derzeit werden E-Mail-, SMS-, MobileApp- und WhatsApp-Nachrichten unterstützt. Die in der Vorlage vorhandenen Deliver Felder können den Zielgruppenfeldern von Interact und Angebotsattributen zugeordnet werden.

Die Interact Laufzeit kann eine Kommunikationsnachricht für die Benutzer mit personalisierten Feldwerten über den im Bildschirm 'Ausgelöste Nachrichten' ausgewählten Ausgangskanal an das Deliver System auslösen.

- Die Feldzuordnung von Interact und Deliver kann auf der Registerkarte 'Gateway' im interaktiven Kanal definiert werden, wenn ein Gateway vom Typ 'Deliver Outbound' ausgewählt wird.
- Die für die Zuordnung vorhandenen Deliver-Felder basieren auf dem Quelltyp und dem ausgewählten zugehörigen Deliver-Dokument. Die Benutzer können die Vorlage über die Registerkarten 'Delivery Messaging Editor' und 'Quick Builder' erstellen.
- Der von Benutzern konfigurierte ausgehende Kanal steht in der Liste Ausgelöste Nachrichten Kanal zur Auswahl.
- Die Feldzuordnungdetails von Interact und Deliver für alle Kanäle werden durch die Bereitstellung des interaktiven Kanals an Interact Runtime übertragen.
- Die TMS-API Verbindungsdetails von Deliver müssen über die Deliver-Vorlage in der Interact-Laufzeit konfiguriert werden.

Sollte das Ereignis oder Ereignismuster mit der postEvent Interact API übereinstimmen und die anderen Bedingungen der ausgelösten Nachrichten ebenfalls erfüllt sind, löst das System die ausgehende Nachricht mit den personalisierten Zielgruppenfeldern und Angebotsattributen gemäß der Feldzuordnung von Interact und Deliver über den Deliver Kanal aus.

Zuordnung von Interact und Deliver

Interact stellt einen Mechanismus bereit, um die Felder von Interact und Deliver zuzuordnen und die zu sendenden Informationen zu identifizieren. Bei der Auswahl von 'Ausgehende Deliver', können die Feldzuordnungen von Deliver auf dem Reiter 'Gateway' eines interaktiven Kanals definiert werden. Die zuzuordnenden Felder von Deliver werden abhängig von dem Quelltyp und der ausgewählten Vorlage abgerufen.

Feldzuordnung von Deliver

Nachdem die grundlegenden Gateway Details auf dem Reiter 'Allgemein' gespeichert werden, können die Administratoren mit dem Reiter 'Quelle' fortfahren.

Hier können Administratoren den Quelltyp für den Nachrichtentyp auswählen, über den die Benutzer kontaktiert werden. Derzeit werden E-Mail, SMS, MobileApp und WhatsApp unterstützt.

Nachdem der Quelltyp ausgewählt wurde, können Benutzer aus der Liste von verfügbaren Vorlagen auswählen. Diese Vorlagen werden aus dem 'Messaging Editor' oder Quick Builder (E-Mail Vorlagen) in Deliver erstellt. Nachdem eine bestimmte Deliver Vorlage ausgewählt wurde, wird alle der Vorlage zugeordneten Felder aufgelistet.

Ausgelöste Nachrichten

Der Bildschirm Ausgelöste Nachrichten listet die verfügbaren ausgehenden Kanäle zur Auswahl auf. Die in der Konfiguration erstellten und ausgehenden Deliver-Kanäle sind auch in der Liste der Kanäle für ausgelöste Nachrichten sichtbar.

Interact-Laufzeitkonfigurationen

-Gateway

Alle Deliver-spezifischen Konfigurationen für die Deliver TMS-API-Details werden unter `Affinium|interact|triggeredMessage|gateways` konfiguriert. Unter dem Gateway-Tab steht eine neue Vorlage mit dem Namen "Deliver" zur Verfügung, die verwendet werden muss, wenn die Informationen des ausgehenden Kanals an das System gesendet werden müssen. Im Folgenden sind die obligatorischen Parameter angegeben, die in der Vorlage "Deliver" erforderlich sind.

- `deliverURL`: Stellen Sie die Transaktions-API-URL bereit.
- `username`: Benutzername für das gehostete Deliver-Konto.
- `dataSourceName`: Datenquelle für das gehostete Deliver-Konto.

Implementierung

Die Deliver-Konfiguration für die Zuordnungsinformationen der Interact-Deliver-Felder ist Teil des Bereitstellungspakets für IC und wird bei der Bereitstellung des "interaktiven Kanals" an Interact Runtime übertragen.

Chapter 22. Gateways konfigurieren

Verwenden Sie Gateways für ausgelöste Nachrichten, um Angebotsinformationen von ausgehenden Kanälen zu senden.

Sie können die folgenden ausgehenden Gateways mit ausgelösten Nachrichten verwenden. Diese Gateways sind unter "Affinium|interact|triggeredMessage|gateways" verfügbar.

In Interact können Sie Informationen über eingehende Gateways erhalten, die unter "Affinium|interact|activityOrchestrator|" verfügbar sind. Die Gateway Dateien sind unter `INTERACT_HOME/conf/gateways/` verfügbar. Die zugehörigen Konfigurationen sind ab sofort unter den Knoten "Affinium | Campaign | Partitionen | Partition1 | Interact | OutboundChannels" für die Entwurfszeit und unter "Affinium | Interact | TriggeredMessage | Gateways", "Affinium | Interact | ActivityOrchestrator | Gateways" verfügbar.

- Unica Interact Eingehendes Gateway für IBM Universal Behavior Exchange
- Unica Interact Ausgehendes Gateway für IBM Universal Behavior Exchange
- Unica Interact Email (Transact) ausgehendes Gateway für IBM Marketing Cloud
- Unica Interact Ausgehendes Gateway für IBM Mobile Push Notification

Die Zuordnung für diese Gateways kann auf der Registerkarte 'Gateway' unter 'Interaktiver Kanal' definiert werden. Wenn Sie die auf Eigenschaftsdateien basierende Zuordnung weiterhin verwenden möchten, müssen Sie die folgenden JVM Parameter festlegen.

Für ausgehende Gateways

`OUTBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES`. Zum Beispiel:

```
DOUTBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES=EMail
```

Für eingehende Gateways

`INBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES`. Zum Beispiel:

```
DOUTBOUND_GATEWAYS_USING_MAPPING_FROM_PROPERTIES=UBX
```

Sollte es mehr als ein Gateway geben, fügen Sie die durch Kommas getrennten Gateway Namen hinzu.

Verwendung von Unica Interact Inbound Gateway für IBM Universal Behavior Exchange

Um das Unica Interact Inbound Gateway für IBM Universal Behavior Exchange zu verwenden, müssen Sie einen Endpunkt und ein Ereignis im UBX-System erstellen, Interact konfigurieren und einen UBX-Abonnentenendpunkt konfigurieren.

Verwenden Sie die folgenden Konfigurationen als Beispiel für Ihre Konfiguration.

A. Endpunkt und Ereignis in UBX erstellen

Sie können diesen Endpunkt und dieses Ereignis als Beispiel verwenden.

Führen Sie die folgenden Schritte aus, um einen Endpunkt und ein Ereignis in UBX zu erstellen.

1. Verwenden Sie den REST API-Client, um die Anforderungen an UBX zu übertragen.
2. Registrieren Sie einen Endpunkt in UBX mit JSON. Beispiel:

```
Method Call: PUT URL: https://ubx-qa1-api.adm01.com/v1/endpoint
 Headers: Content-Type: application/json Accept-Charset: UTF-8
 Authorization: Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
 (Note: This is the Auth Key generated from
 the UBX UI.) 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. Registrieren Sie einen Ereignistyp in UBX mit JSON. Beispiel:

```
Event Registration for Interact Event in UBX Method Call:
 POST URL: https://ubx-qa1-api.adm01.com/v1/eventtype
 Headers: Content-Type: application/json Accept-Charset: UTF-8
 Authorization: Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
 Note: This is the Auth Key generated from the UBX UI.)
 Bearer 912586bf-190d-48f9-8488-26f1bf532ef3 Body { "name":
 "recommendedOffers", "description": "recommended offers by OMO",
 "code": "recommendedOffers" }
```

4. Übertragen Sie ein Ereignis an UBX mit JSON. Beispiel:

```
{ "channel" : "mobile", "identifiers" : [ { "name" :
"interactprofileid", "value" : "55" } ], "events" : [ { "code" :
"recommendedOffers", "timestamp" : "2015-12-28T20:16:12Z" } ] }
```

B. Konfigurieren von Unica Interact für Unica Interact Inbound Gateway für IBM Universal Behavior Exchange

Führen Sie die folgenden Schritte aus, um Unica Interact zu konfigurieren.

1. Fügen Sie in der Konfigurationseigenschaft **Interact | activityOrceshtrator | Empfänger** einen neuen Empfänger hinzu. Legen Sie **Typ**auf `Kafka` oder `Benutzerdefiniert` fest. Wenn Sie `Benutzerdefiniert` auswählen, geben Sie **ClassName** und **ClassPath** ein. Wenn Sie `Kafka` auswählen, lassen Sie **ClassPath** und **ClassName** leer.
2. Fügen Sie die Parameter `providerUrl`, `topic`, `authentication`, `group.id` und `zookeeper.connect` für Ihren Empfänger hinzu.
3. In der Konfigurationseigenschaft **Interact | activityOrceshtrator | Gateways** ist die UBX-Kategorie standardmäßig verfügbar.
4. • Verwendung von auf Eigenschaftendateien basierenden Zuordnungen.
    - Erstellen Sie einen neuen Ordner, z. B. den Ordner `Interactubx12`, im Verzeichnis `<Interact_Home>\conf\inbound\UBX` und kopieren Sie die Eigenschaftendateien in diesen neuen Ordner. Der Ordnername muss mit dem Namen des Subscriber-Endpunkts übereinstimmen, den Sie in UBX erstellt haben.

- Konfigurieren Sie die Datei interactEventNameMapping.properties

  Verwenden Sie diese Datei, um den Wert des Nutzdatenereignisfelds, das in der Datei interactEventPayloadMapping.properties als [EventName] definiert ist, dem Interact-Ereignisnamen zuzuordnen. Die Datei interactEventNameMapping.properties befindet sich im Verzeichnis `<Install dir>\conf\inbound\UBX`. `{UBX event name}={Interact event name}` Beispiel: recommendedOffers=recommendedOffers

  Sollten die Nutzdaten aus einer bestimmten Quelle unterstützt werden, kann diese Datei auch im Verzeichnis `<Install dir>\conf\inbound\UBX\{source}` gespeichert werden. Der Wert für die Quelle muss mit dem Wert des Quellenfelds in den Ereignisnutzdaten von UniversalBehavior Exchange übereinstimmen, der in der Regel der Endpunktname von Universal Behavior Exchange ist. Sollten die Daten über bestimmte Versionen unterstützt werden, kann diese Datei auch im Verzeichnis `<Install dir>\conf\inbound\UBX\{source}\version-{version}` gespeichert werden. Der Wert für 'version' muss mit dem Wert des Versionsfelds in den Ereignisnutzdaten von Universal Behavior Exchange übereinstimmen.

  Zur Unterstützung mehrerer Instanzdaten von Universal Behavior Exchange kann diese Datei auch im Verzeichnis `<Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID}` gespeichert werden. Der Wert für clientID muss mit dem Wert von clientID in den Ereignisnutzdaten von Universal Behavior Exchange übereinstimmen.
- Konfigurieren Sie die Datei interactEventPayloadMapping.properties

  Verwenden Sie die Datei interactEventPayloadMapping.properties, um das eingehende Feld den Interact-API-Parametern zuzuordnen. Die Datei interactEventPayloadMapping.properties befindet sich im Verzeichnis `<Install dir>\conf\inbound\UBX`.

Interact API Parameter: Der Wert muss mit einer Feldtypdefinition starten und von einem statischen Wert gefolgt werden, wenn der Wert in Anführungszeichen steht, oder er muss von einem Feldnamen aus den Nutzdaten gefolgt werden. (FIELD_TYPE)"STATIC_VALUE" oder (FIELD_TYPE)PAYLOAD_FIELD_NAME. FIELD_TYPE kann String, Numeric oder DateTime sein.

Zum Beispiel:

```
[SessionID]=(String)interactprofileid
 [EventName]=(String)code
 [AudienceIDFieldNames]=(String)"change_me" [AudienceIDField
Values]=(String)interactprofileid
 [AudienceLevel]=(String)"change_me" [InteractChannel]=(Stri
ng)"change_me"
```

Ereignisdaten: Diese Eigenschaften werden verwendet, um die Ereignisattribute zuzuordnen, die in Ihrem Kanal für ausgehende Kommunikation verwendet werden können. Die linke Seite enthält die Variablennamen, Sie Sie in Ihrem Kanal für abgehende Kommunikation verwenden. Der Wert muss mit einer Feldtypdefinition starten und von einem statischen Wert gefolgt werden, wenn der Wert in Anführungszeichen steht, oder er muss von einem Feldnamen aus den Nutzdaten gefolgt werden. (FIELD_TYPE)"STATIC_VALUE" oder (FIELD_TYPE)PAYLOAD_FIELD_NAME. FIELD_TYPE kann String, Numeric oder DateTime sein.

Sollten die Nutzdaten aus einer bestimmten Quelle unterstützt werden, kann diese Datei auch im Verzeichnis `<Install dir>\conf\inbound\UBX\{source}` gespeichert werden. Der Wert für die Quelle muss mit dem Wert des Quellenfelds in den Ereignisnutzdaten von Universal Behavior Exchange übereinstimmen, der in der Regel der Endpunktname von Universal Behavior Exchange ist. Sollten die Daten über bestimmte Versionen unterstützt werden, kann diese Datei auch im Verzeichnis

`<Install dir>\conf\inbound\UBX\{source}\version-{version}` gespeichert werden. Der Wert für 'version' muss mit dem Wert des Versionsfelds in den Ereignisnutzdaten von Universal Behavior Exchange übereinstimmen. Zur Unterstützung mehrerer Instanzdaten von Universal Behavior Exchange kann diese Datei auch im Verzeichnis `<Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID}` gespeichert werden. Der Wert für clientID muss mit dem Wert von clientID in den Ereignisnutzdaten von Universal Behavior Exchange übereinstimmen.

- Starten Sie den Interact Server erneut.
- Erstellen Sie einen interaktiven Kanal und fügen Sie ihm ein Ereignis hinzu.

- UI-basierte Zuordnungen verwenden:

  Verwenden Sie die Option **Allgemein-Eingehend**, die auf der Registerkarte **Gateway** im interaktiven Kanal verfügbar ist. Erstellen Sie ein Gateway mit dem Namen 'UBX' und definieren Sie die Ereigniszuordnungen.

  Es ist erforderlich, dass die von UBX eingehende Anfrage den Header-Parameter ICName enthält. Die Zuordnungen aus dem spezifischen interaktiven Kanal werden basierend auf diesem Header Wert initialisiert.

5. Führen Sie die in den folgenden Abschnitten C und D genannten Schritte aus.
6. Stellen Sie den interaktiven Kanal bereit.
7. Übertragen Sie ein Ereignis an UBX aus einem Publisher oder zum Testen mit einem REST API-Client.

   Beispielereignis:

   ```
   { "channel" : "mobile", "identifiers" : [ { "name" :
   "interactprofileid", "value" : "55" } ], "events" : [ { "code" :
   "recommendedOffers", "timestamp" : "2015-12-28T20:16:12Z" } ] }
   ```

8. Prüfen Sie das Unica Interact-Protokoll, um zu sehen, ob das Ereignis für ausgelöste Nachrichten ausgelöst wird. Sie können dieses Ereignis für die weitere Verarbeitung

in Mustern / ausgelösten Nachrichten gemäß Ihrem speziellen Anwendungsfall verwenden

Weitere Informationen finden Sie auch in den Endpunktprotokollen

C. Konfiguration von Unica Interact Eingehendes Gateway für IBM Universal Behavior Exchange

Sie sollten die Anweisungen auch für die folgenden Konfigurationen verwenden.

- UBX-Endpunkt mit Kafka
- Endpunktdatei ubxInboundEndpoint.properties
- Endpunktdatei inboundTopicNameConfig.properties
- Endpunktdatei log4j2.xml

Die Datei ubxInboundEndpoint.properties

Mit der Datei ubxInboundEndpoint.properties konfigurieren Sie, wohin die Ereignisnutzdaten von Universal Behavior Exchange in Unica Interact Inbound Gateway für IBM Universal Behavior Exchange gesendet werden sollen. Die Datei lubxInboundEndpoint.properties befindet sich im Verzeichnis `<gateway endpoint install dir on the application server>`.

- **providerURL** erforderlich: Eine Liste von Host- oder Port-Paaren, die für die Herstellung der ersten Verbindung mit dem Kafka-Cluster verwendet werden sollen. Diese Liste muss das Format host1:port1 haben.
- **Authentifizierung**: Standardmäßig keine - Für Informationen über verschiedene Authentifizierungsmethoden und die zugehörige Eigenschaften, siehe Abschnitt **Interact | activityOrchestrator | Empfänger**.
- **group.id**: Erforderlich - Beliebige String (Beispiel: InteractTMGateway)
- **zookeeper.connect** Optional - `<host>:<port>` (Beispiel: localhost:2181)

Ein Neustart der Webanwendung des Gateway-Endpunkts (ubxInboundEndpoint.war) ist auf dem Web-Server oder Anwendungsserver erforderlich.

- UBX-Endpunkt mit Kafka
- Endpunktdatei ubxInboundEndpoint.properties
- Endpunktdatei inboundTopicNameConfig.properties
- Endpunktdatei log4j.properties

Die Datei inboundTopicNameConfig.properties

Der Endpunkt von Unica Interact Inbound Gateway für IBM Universal Behavior Exchange sendet das Ereignis durch Schreiben in einem Kafka-Thema an Interact. Verwenden Sie die Datei inboundTopicNameConfig.properties, um den Themennamen anzugeben, für den Daten veröffentlicht werden.

Beispiel:

```
topic=UBXTopic
```

Ein Neustart der Webanwendung des Gateway-Endpunkts (ubxInboundEndpoint.war) ist auf dem Web-Server oder Anwendungsserver erforderlich, damit Änderungen in dieser Datei wirksam werden.

Die Datei log4j2.xml

Verwenden Sie die Datei log4j2.xm, um verschiedene Protokollebenen für den Endpunkt zu konfigurieren. Die Datei log4j2.xm befindet sich im <Installationsverzeichnis des Gateway-Endpunkts auf dem Anwendungsserver>.

Beschreibung

Legen Sie die Protokollebene für com.ibm.web.offerorchestration.inbound.common und com.ibm.web.offerorchestration.inbound.ubx entsprechend fest.

D. Bereitstellen des Unica Interact Inbound Gateways für IBM Universal Behavior Exchange und Endpunkt

- Der eingehende Gateway-WAR von Unica Interact befindet sich am Speicherort `<Interact_Home>\UBXInboundEndpoint\ubxInboundEndpoint.war`.
- Kopieren Sie diese WAR-Datei zusammen mit dem Ordner conf auf von Unica unterstützte App-Servern. Dieser Server übergibt Daten an das eingehende Interact-Inbound-Kafka-Thema, das später von Unica Interact Inbound Gateway für IBM Universal Behavior Exchange verarbeitet wird.

Der Endpunkt von Unica Interact Eingehenden Gateway für IBM Universal Behavior Exchange ist so konfiguriert, dass er Anforderungen von Universal Behavior Exchange akzeptiert und an Unica Interact Inbound Gateway für IBM Universal Behavior Exchange sendet.

Sie müssen die folgenden Tasks ausführen, um den Subscriber-Gateway-Endpunkt von Universal Behavior Exchange zu konfigurieren:

1. Konfigurieren Sie eine neue Java-Systemeigenschaft (`-DubxInboundEndpointConfigPath`), indem Sie die Konfigurationsdatei auf dem Webserver oder in der Administrationskonsole des Anwendungsservers bearbeiten.

   Die Eigenschaft `-DubxInboundEndpointConfigPath` muss auf das Endpunkt-Installationsverzeichnis auf dem Server verweisen (d. h. conf Ordner, der im obigen Abschnitt kopiert wird).

   Dieses Verzeichnis enthält Konfigurationsdateien für die Ziel-Kafka-Umgebung und verschiedene Protokollstufen für den Endpunkt. Zum Beispiel `-DubxInboundEndpointConfigPath=c: \ubxInboundEndpoint`.
2. Stellen Sie die Webarchivdatei des Endpunkts (ubxInboundEndpoint.war) von Unica Interact eingehenden Gateway für IBM Universal Behavior Exchange vom Installationsverzeichnis bereit, wie in der Web-Server- oder Anwendungsserverdokumentation beschrieben.

Sie können prüfen, ob der Endpunkt ordnungsgemäß installiert wurde, indem Sie die folgende Adresse in einen beliebigen Browser eingeben und nach der Nachricht suchen.

`UBX-Endpunkt ist UP`.

```
http://[Server]:[Port]/[ContextRoot]/UBXEndPoint
```

Note: Sie müssen den öffentlich zugänglichen Endpunkt von Unica Interact eingehenden Gateway für IBM Universal Behavior Exchange schützen, indem Sie notwendige Firewallregeln hinzufügen, um nur HTTP-Anforderungen von IBM Universal Behavior Exchange Server zu akzeptieren.

Beispielsweise können Sie mithilfe der folgenden Anweisungen den Endpunkt von Unica Interact eingehenden Gateway für IBM Universal Behavior Exchange auf WebSphere Application Server konfigurieren und bereitstellen.

1. Öffnen Sie die Administrationskonsole.
2. Wählen Sie **Server > (Servertypen aufklappen) > server_name > (Java™ und Prozessmanagement aufklappen) > Prozessdefinition > Java Virtual Machine** aus.
3. Fügen Sie in den generischen JVM-Argumenten die Eigenschaft `-DubxInboundEndpointConfigPath=<Universal Behavior Exchange Subscriber Gateway endpoint install dir on the application server>` hinzu. Fügen Sie z.B. die Eigenschaft `-DubxInboundEndpointConfigPath=C:\ubxInboundEndpoint` hinzu.
4. Klicken Sie auf **OK**, um die Änderungen in der Masterkonfiguration zu speichern.
5. Starten Sie den Anwendungsserver erneut.

Stellen Sie den Endpunkt in WebSphere Anwendungsserver bereit.

1. Melden Sie sich an der Administrationskonsole an.
2. Gehen Sie zu **Anwendungen > Anwendungstypen > WebSphere Enterprise Anwendungen**. Klicken Sie auf **Installieren**.
3. Suchen Sie mithilfe der Option **Auf die Anwendungsinstallation vorbereiten** nach der zu installierenden Endpunkt-WAR-Datei (`ubxInboundEndpoint.war`) und klicken Sie dann auf **Weiter**.

4. Auf den nachfolgenden Seiten, klicken Sie auf **Weiter**, um zu **Stammkontext für Webmodule zuordnen** zu navigieren.
5. Verwenden Sie die **Stammkontext für Webmodule zuordnen**, um die Stammkontext zu finden und ändern Sie den Wert in `/UBXEndPoint`. Dies wird dann den Stammkontext. Klicken Sie auf **Weiter**.
6. Klicken Sie auf **Fertig stellen**.
7. Nachdem die Anwendung installiert wird, klicken Sie auf **Speichern**, um die Änderungen in der Masterkonfiguration beizubehalten.
8. Markieren Sie in den aufgelisteten und installierten Anwendungen das Kontrollkästchen für ubxInboundEndpoint_war und klicken Sie auf **Start**, um den Ladevorgang zu starten.

Note:

UBXInboundEndpoint wird standardmäßig aktualisiert, um Kafka zu unterstützen. UBXInboundEndpoint wird standardmäßig mit Interact geliefert.

Wenn Sie UBXInboundEndpoint bereits mit JMS Queue verwenden, können Sie die vorhandene UBXInboundEndpoint-WAR-Datei weiterhin verwenden.

Verwendung von Unica Interact Outbound Gateway für IBM Universal Behavior Exchange

Sie können Unica Interact Outbound Gateway für IBM Universal Behavior Exchange nur verwenden, wenn Sie Unica Interact, UBX und das Gateway konfigurieren.

Verwenden Sie die folgenden Konfigurationen als Beispiel für Ihre Konfiguration.

Wenn Sie UBX als Kanal für abgehende Nachrichten verwenden, fungiert Unica Interact als Publisher-Endpunkt, der Ereignisse an UBX veröffentlicht. Von UBX aus können diese Ereignisse an den Subscriber gesendet werden.

Bevor Sie mit der Konfiguration beginnen, fordern Sie ausgehenden Zugriff auf die Hostmaschine an. Der Netzzugriff muss für die Hostmaschine aktiviert sein.

Endpunkte und Ereignisse in UBX registrieren

Als Voraussetzung muss ein entsprechender Endpunkt bei UBX registriert werden. Wenden Sie sich an Akustik, um Endpunkte und Ereignisse in UBX zu erstellen.

Konfigurieren von Unica Interact und des Gateways

1. Unter dem Ordner **Interact**, geben Sie die Zeitüberschreitung in der Datei `httpConnectionConfig.properties` an.
   Zum Beispiel:

   ```
   connectTimeoutMs=180000
   ```

   Wenn OMO für die Verwendung einer HTTP-Verbindung konfiguriert wird, kann optional ein HTTP-Proxy mit Authentifizierung zwischen Interact und dem Endpunkt konfiguriert werden. Um den Proxy für ausgehende Gateways zu aktivieren, aktualisieren Sie die Werte der folgenden Eigenschaften.
   - proxyHost=<`IP-Adresse des Proxy-Servers`>
   - proxyPort=<`Listening Port des Proxy-Servers`>
   - targetUsername=<`Benutzername für die Verbindung mit dem Proxyserver. Leer lassen, wenn keine Authentifizierung erforderlich ist`>
   - targetPassword=<`Kennwort für die Verbindung mit dem Proxyserver. Leer lassen, wenn keine Authentifizierung erforderlich ist`>

   Zum Beispiel:

   ```
   authKey=912586bf-190d-48f9-8488-26f1bf532ef3 [Auth Key
   used to register publisher endpoint and event in UBX]
   interactProfileIdFieldName=interactprofileid [Field name from the
   ubxContentMapping.properties file]
   ```

2. Verwendung von auf Eigenschaftendateien basierenden Zuordnungen
   - Aktualisieren Sie die Werte für `interactprofileid` und `eventName` in der Datei `ubxContentMapping.properties`.

     Sie können den Ereignisnamen in drei Formaten übergeben: Wenn der Wert in Anführungszeichen steht, ist er ein statischer Wert. Wenn der

Wert das Format `offer.offerAttributeName` aufweist, wird er dem Angebotsattribut offerAttributeName zugeordnet. Wenn der Wert das Format `profile.profileAttributeName` aufweist, wird er dem Profilattribut profileAttributeName zugeordnet. Der Wert des Ereignisnamens sollte mit dem Code übereinstimmen, mit dem das Ereignis in UBX registriert wurde. Hierbei wird zwischen Groß- und Kleinschreibung unterschieden. Starten Sie den Anwendungsserver erneut.

Zum Beispiel:

```
eventName="abandoned_shopping_carts" eventName=offer.Card
 eventName=profile.EMAIL
```

- UI-basierte Zuordnungen verwenden.
  - Erstellen Sie in einem interaktiven Kanal ein Gateway mit dem Namen 'UBX' vom Typ Allgemein - Ausgehend.
  - Erstellen Sie eine Kanaleigenschaft mit dem Eigenschaftsnamen interactProfileIdFieldName und ein Interact Feld als einen erforderlichen Feldnamen für den Endpunkt. Beispiel: interactProfileIdFieldName=interactprofileid .
  - Unter dem Abschnitt Zuordnung können Sie den eventName in den folgenden drei Formaten definieren:

    eventName=abandoned_shopping_carts (Standardwert kann zugewiesen werden, um den statischen eventName zu erhalten)

    eventName=offer.Card (Der Wert wird aus Angebotsattributen abgerufen)

    eventName=profile.EMAIL (Der Wert wird aus Profilattributen abgerufen)

3. Unter der Konfigurationseigenschaft **Interact | triggeredMessage | Kanal** , fügen Sie einen Kanal hinzu.
4. Definieren Sie denselben Kanal in Designzeit unter **Campaign | Partitionen | Partition [n] |Interact | outboundChannels**.
5. Erstellen Sie eine Regel für ausgelöste Nachrichten mit einem Ereignisnamen, der den Kanal verwendet, den Sie in den vorherigen Schritten hinzugefügt haben.

6. Stellen Sie den interaktiven Kanal bereit.
7. Starten Sie auf dem API-Testclient die Sitzung für den interaktiven Kanal, in der die Regel für ausgelöste Nachrichten konfiguriert wird, sowie das Post-Ereignis, mit dem das Angebot für UBX ausgelöst wird.

Verwendung von Unica Interact Outbound Gateway für IBM Mobile Push Notification

Um dieses ausgehende Mobile Push- bzw. Publisher-Gateway zu verwenden, müssen Sie Unica Interact, IBM Marketing Cloud und das Gateway konfigurieren.

Verwenden Sie die folgenden Konfigurationen als Beispiel für Ihre Konfiguration.

IBM Marketing Cloud konfigurieren

1. Stellen Sie sicher, dass Sie ein IBM Marketing Cloud-Konto mit Push-Zugriff haben. Notieren Sie zudem Ihre Client-ID, Ihren geheimen Clientschlüssel und Ihren Aktualisierungstoken.
2. Erstellen Sie auf der Registerkarte **Daten** eine neue Datenbank. Fügen Sie eine neue `Mobile Benutzer-ID` zur Datenbank zusammen mit den Standardfeldern hinzu.
3. Suchen Sie auf der Registerkarte **Suchen** mithilfe des Felds `Mobile Benutzer-ID`. Bewegen Sie die Maus über das erste Feld `Keine E-Mail`. Die `Empfänger ID` wird unten im Browserfenster angezeigt. Fügen Sie diese `Empfänger ID` zur Unica Interact Profiltabelle hinzu.

Konfigurieren von Unica Interact Outbound Gateway für IBM Mobile Push Notification

1. Konfigurieren Sie die Datei `silverpopEngagePushConfig.properties`.

   Zum Beispiel:

   ```
   OauthServiceURL=<protocol>://<hostname>/<other_information>
    pushServiceURL=<protocol>://<hostname>/<other_information>
   ```

2. Konfigurieren Sie die Datei `silverpopEngagePushContentMapping.properties`.

   Note:
   Werden UI-basierte Zuordnungen verwendet, führen Sie die folgenden Aktionen aus.

   a. Erstellen Sie in einem interaktiven Kanal ein Gateway mit dem Namen 'MobilePush' vom Typ Allgemein - Ausgehend.
   b. Erstellen Sie die folgenden Zuordnungen unter dem Abschnitt Zuordnung.

   Zum Beispiel:

```
Interact Profile table attributes: appKey=appKey
 engageRecipientId=recipientId mobileUserId=mobileUserId
 deviceType=deviceType Interact Offer attributes:
 simpleSubject=simpleSubjectAttr simpleMessage=simpleMessageAttr
 simpleActionData=simpleActionDataAttr
 simpleActionType=simpleActionTypeAttr
 simpleActionLabel=simpleActionLabelAttr
 personalizeAttributeList=personalizeAttributeList contentId=ContentID
 campaignId=campaignId
```

Konfiguration von Unica Interact

1. Erstellen Sie die folgenden Angebotsattribute.

```
simpleActionDataAttr: string simpleActionLabelAttr: String
 simpleActionTypeAttr: string simpleMessageAttr: string
 simpleSubjectAttr: string contentID: string campaignId=string
 personalizeAttributeList=string
```

2. Erstellen Sie eine Angebotsvorlage mit den Angebotsattributen und den folgenden Angebotswerten.

```
simpleActionDataAttr: www.ibm.com simpleActionLabelAttr: Open URL
 simpleActionTypeAttr: url simpleMessageAttr: <Enter your message
 text here> simpleSubjectAttr: <Enter subject here> contentID:
 ID of the push message template that is created in Engage.
 PersonalizeAttributeList: Eine durch Kommas getrennte Liste
 von Wertpaaren der Attributnamen, die Sie unter dem Abschnitt
 personalizationDefaults der an Engage zu sendenden Nutzdaten einfügen
 möchten.
```

Wenn Sie das Attribut `contentID` verwenden, werden die anderen `simple..` Attribute ignoriert, weil die gesamten Details von der Engage-Vorlage abgerufen werden.

Beispiel: personalizedAttributeList

```
personalizeAttributeList=discount=10,Offercost=20
 campaignId=campaignname that you want to use for this campaign.
```

3. Ihre Profiltabelle enthält die folgenden Spalten und Werte.

```
appKey: gcsTQo6v79 recipientId: 13472242 deviceType: android or ios
```

4. Navigieren Sie zu `INTERACT_HOME/conf/gateways/outbound/common`.
   Unter dem Ordner **Interact**, geben Sie die Zeitüberschreitung in der Datei `httpConnectionConfig.properties` an.
   Zum Beispiel:

```
connectTimeoutMs=6000
```

Wenn OMO für die Verwendung einer HTTP-Verbindung konfiguriert wird, kann optional ein HTTP-Proxy mit Authentifizierung zwischen Interact und dem Endpunkt konfiguriert werden. Um den Proxy für ausgehende Gateways zu aktivieren, aktualisieren Sie die Werte der folgenden Eigenschaften.

- proxyHost=`<IP-Adresse des Proxy-Servers>`
- proxyPort=`<Listening Port des Proxy-Servers>`

    - targetUsername=<`Benutzername für die Verbindung mit dem Proxyserver. Leer lassen, wenn keine Authentifizierung erforderlich ist`>
    - targetPassword=<`Kennwort für die Verbindung mit dem Proxyserver. Leer lassen, wenn keine Authentifizierung erforderlich ist`>
5. Erstellen Sie einen Kanal und einen Handler unter **Interact | triggeredMessage** und verwenden Sie das [Mobile_Push] Gateway, das Sie oben in diesem Kanal erstellt haben. Dieser Kanal wird in der ausgelösten Nachricht verwendet, um Push-Nachrichten zu senden.
6. Erstellen Sie einen interaktiven Kanal und fügen Sie eine ausgelöste Nachricht hinzu, die das Angebot verwendet, das Sie zuvor für die Auslöseregel erstellt haben.
7. Stellen Sie den interaktiven Kanal bereit.
8. Führen Sie vom API-Testclient aus eine `startSession` Methode für den interaktiven Kanal durch, in der die Regel für ausgelöste Nachrichten konfiguriert wird, sowie die `postEvent` Methode, mit der das Angebot für Mobile Push ausgelöst wird.
9. Prüfen Sie die Unica Interact-Protokolle, um sicherzustellen, dass der Push erfolgreich gesendet wurde. Der Statuscode `202` bedeutet eine erfolgreiche Übermittlung.

Verwendung von Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud

Sie können diese Integration in Silverpop, Unica Interact und Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud verwenden, um ausgelöste E-Mail-Angebote an Ihre Kunden zu senden.

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Erstellen Sie eine Profiltabelle für die Zielgruppe "Kunde" mit der Spalte "E-Mail". Verwenden Sie diese Profiltabelle für Ihren interaktiven Kanal.
- Fordern Sie an, dass der Netzzugriff auf die Hostmaschine für Ihren Kanal für abgehende Nachrichten aktiviert wird.

Hinzufügen eines Dispatchers für die Gateway-Integration

Der Dispatcher fügt Ihr Angebot einer Warteschlange für Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud hinzu, damit Ihre Angebots-E-Mail versendet werden kann.

Sie müssen einen Dispatcher hinzufügen, um HCL Email (Transact) Outbound Gateway für IBM Marketing Cloud verwenden zu können.

1. Navigieren Sie in den Konfigurationseigenschaften zu **Interact | triggeredMessage | Dispatchers | <dispatcherName>**.
2. Fügen Sie einen **Neuen Kategorienamen** für Ihren Dispatcher hinzu.
3. Wählen Sie einen **Typ** aus. Sie können aus "InMemoryQueue", "JMSQueue" und "Benutzerdefiniert" auswählen.
4. Geben Sie den **className** ein.
5. Geben Sie den **classPath** ein.

Parameter "OMO-conf_outbound_common_httpConnectionConfig" konfigurieren

Navigieren Sie zum `INTERACT_HOME/conf/gateways/outbound/common`: Geben Sie unter dem Ordner **Interact** in der Datei `httpConnectionConfig.properties` die Zeitüberschreitung an.

Beispiel: `connectTimeoutMs=60000` Wenn OMO für die Verwendung einer HTTP-Verbindung konfiguriert ist, kann optional ein HTTP-Proxy mit Authentifizierung zwischen Interact und dem Endpunkt konfiguriert werden. Um den Proxy für ausgehende Gateways zu aktivieren, aktualisieren Sie die Werte der folgenden Eigenschaften.

- proxyHost=`<IP-Adresse des Proxy-Servers>`
- proxyPort=`<Listening Port des Proxy-Servers>`
- targetUsername=`<Benutzername für die Verbindung mit dem Proxyserver. Leer lassen, wenn keine Authentifizierung erforderlich ist>`
- targetPassword=`<Kennwort für die Verbindung mit dem Proxyserver. Leer lassen, wenn keine Authentifizierung erforderlich ist>`

Parameter "OMO-conf_outbound_silverpop_silverpopConfig" konfigurieren

Legen Sie in der Datei `silverpopConfig.properties` die Werte für `OauthServiceURL`, `xmlAPIServiceURL`, `clientID`, `clientSecret`, und `refreshToken` fest. Wenden Sie sich an Ihren Marketing Cloud-Administrator, um kundenspezifische Werte aus der Datei `transact.xml` abzurufen.

Parameter "OMO-conf_outbound_silverpop_silverpop ContentMapping" konfigurieren

Sie müssen den Parameter "OMO-conf_outbound_silverpop_silverpopContentMapping" für Ihr Gateway konfigurieren.

Wird die auf der Eigenschaftendatei basierende Zuordnung verwendet, aktualisieren Sie die Eigenschaftendatei mit den folgenden Zuordnungen.

- Wird UI-basierte Zuordnung verwendet, erstellen Sie ein Gateway mit dem Namen 'EMail'.
- Unter dem Abschnitt Zuordnung, erstellen Sie die Zuordnungen für die folgenden Eigenschaften.

In der Datei `silverpopContentMapping.properties`, setzen Sie die Werte für Ihre Inhaltszuordnung ein.

a. Setzen Sie die Eigenschaft `campaignId` ein. Der Wert für diese Eigenschaft ist der Name eines Angebotsattributs, das in Ihren Angebotsvorlagen angegeben wird.
b. Setzen Sie die `email` Eigenschaft ein. Der Wert für diese Eigenschaft ist der Spaltenname in Ihrer Profiltabelle. Fügen Sie Ihrer Profiltabelle eine Spalte `email` hinzu und geben Sie die E-Mail-IDs an. Dies sind die E-Mail-IDs der Empfänger.
c. Definieren Sie Ihre Angebotsattribute in `AdditionalOfferPfAttributesUsedInEmail`. Diese Eigenschaft legt die Attribute Ihrer Angebotsvorlage fest, die für die Mailingvorlage benötigt werden. Sie können

`additionalProfilePfAttributesUsedInEmail` verwenden, um Felder aus Ihrer Profiltabelle zu definieren. Mit `*` können Sie alle Angebotsattribute und Spaltenwerte berücksichtigen.

Parameter "deliveryTimeoutMillis" konfigurieren

Legen Sie den Parameter `deliveryTimeoutMills` fest, um das Unica Interact-Serverzeitlimit zum Herstellen einer Verbindung zum Marketing Cloud-Server zu erhöhen.

1. Navigieren Sie in den Konfigurationseigenschaften zu **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | deliveryTimeoutMillis**.
2. Legen Sie den **Wert** fest. Sie könnten den **Wert** beispielsweise auf 60000 festlegen. Dadurch würde das Serverzeitlimit auf 60000 Millisekunden erhöht werden.

Hinzufügen eines Kanal-Handlers für das Unica Interact E-Mail (Transact) Outbound-Gateway für die IBM Marketing Cloud

Fügen Sie einen Kanal-Handler in der Unica InteractLaufzeitumgebung hinzu.

1. Navigieren Sie in den Konfigurationseigenschaften zu **Interact | triggeredMessage | Kanäle | <SilverpopChannelName> | <handlerName>**.
2. Fügen Sie einen **neuen Kategorienamen** für Ihren Kanal-Handler hinzu.
3. Legen Sie den Namen des Dispatchers fest, den Sie zuvor hinzugefügt haben.
4. Legen Sie den Namen des Gateways fest.
5. Legen Sie den **Modus** fest. Wenn "Failover" ausgewählt wird, wird dieser Handler nur verwendet, wenn keiner der Handler mit höheren Prioritäten, die in diesem Kanal definiert sind, ein Angebot senden konnte. Wenn **Addon** ausgewählt wird, wird dieser Handler unabhängig davon verwendet, ob andere Handler erfolgreich Angebote gesendet haben.
6. Legen Sie die **Priorität** für diesen Handler fest.

Hinzufügen eines Kanals für abgehende Nachrichten für Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud

Fügen Sie in der Unica Interact-Designumgebung einen Kanal für abgehende Nachrichten hinzu.

1. Navigieren Sie in den Konfigurationseigenschaften zu **Campaign | Partitionen | Partition[N] | Interact | outboundChannels**.
2. Fügen Sie einen **Neuen Kategorienamen** für Ihren Kanal für abgehende Nachrichten hinzu.
3. Fügen Sie einen **Namen** für Ihren Kanal für abgehende Nachrichten hinzu. Stellen Sie sicher, dass der Kanalname mit dem Kanalnamen übereinstimmt, den Sie in der Konfigurationseigenschaft **Interact | triggeredMessage | Kanäle | <SilverpopChannelName>** hinzugefügt haben.

Konfigurieren des Transaktionsmailing mit Unica Interact Email (Transact) Outbound Gateway für IBM Marketing Cloud

Sie müssen Ihr Transaktionsmailing konfigurieren, um Ihr E-Mail-Angebot versenden zu können.

1. Klicken Sie in Marketing Cloud (Transact) auf **Daten > Datenbank erstellen**. Klicken Sie anschließend auf **Erstellen**, um eine Profiltabelle zu erstellen. Sie können die Profiltabelle auch dort importieren, wo Sie die E-Mail-Spalte hinzugefügt haben.
2. Klicken Sie auf **Automatisierung > Transaktionsnachrichten > Gruppe erstellen**. Wählen Sie **Transaktion** für den **Ereignisauslöser** aus. Zudem müssen Sie die Datenquelle auswählen, die Sie zuvor erstellt haben. Klicken Sie auf **Speichern & Aktivieren**.

   Das über Marketing Cloud gesendete Angebot sollte das gleiche Attribut enthalten, das Sie für die `campaignId` in der Datei `silverpopContentMapping.properties` festgelegt haben. Der Wert für dieses Angebotsattribut ist die `campaignId`, die für die automatisierte Nachrichtengruppe generiert wird.

3. Klicken Sie auf **Inhalt > Mailings erstellen** und wählen Sie die Inhaltsquelle aus dem vorherigen Schritt aus. Geben Sie den Hauptteil des Mailings ein. Klicken Sie auf **Automatisieren**. Wählen Sie **Mailing der vorhandenen Gruppe automatisierter Nachrichten zuweisen** aus. Klicken Sie auf **Übergeben & Aktivieren**.

   Die Betreffzeile und der Hauptteil des Mailings können mithilfe von Angebots- und Profilattributen personalisiert werden. Verwenden Sie die Syntax `%%Attribute_Name%%` zum Definieren von Attributen.
4. Der Marketing Cloud-Server akzeptiert nur Übermittlungen ausgehender Gateways über IP-Adressen, die im Vorhinein eingerichtet wurden. Um eine IP-Adresse hinzuzufügen, navigieren Sie zu **Einstellungen > Org.-Admin > Sicherheitseinstellungen > Zugriffsbeschränkungen**.
5. Wenn Sie WebSphere Application Server verwenden, müssen Sie das SSL-Zertifikat für Marketing Cloud importieren. Dies ist bei WebLogic-Benutzern nicht erforderlich.
   a. Navigieren Sie auf der WebSphere Application Server-Konsole zu **SSL-Zertifikat und Schlüsselverwaltung > Schlüsselspeicher und Zertifikat > NodeDefaultTrustStore > Unterzeichnerzertifikate > Von Port abrufen**.
   b. Legen Sie den Host und den Port fest.
   c. Starten Sie den WebSphere Application Server erneut.