IBM Unica Interact

バージョン 8 リリース 6

2012 年 5 月 25 日

管理者ガイド

注

本書および本書で紹介する製品をご使用になる前に、 277 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM Unica Interact (製品番号 5725-D22) バージョン 8 リリース 5 モディフィケーション 0、および、新しいエディションで明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM Unica Interact
Version 8 Release 6
May 25, 2012
Administrator's Guide

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.6

目次

第 1 章 IBM Unica Interact の管理 . . . 1

Interact キーの概念 1
オーディエンス・レベル 1
設計環境. 2
イベント. 2
インタラクティブ・チャネル 2
インタラクティブ・フローチャート. 3
インタラクション・ポイント 3
オファー. 3
プロファイル 3
ランタイム環境 4
ランタイム・セッション 4
タッチポイント 4
処理ルール 4
Interact アーキテクチャー 5
Interact ネットワークに関する考慮事項 5

第 2 章 IBM Unica Interact ユーザーの構成 9

ランタイム環境ユーザーの構成 9
設計環境ユーザーの構成 9
設計環境のアクセス権の例 11

第 3 章 Interact データ・ソースの管理 13

Interact データ・ソースを使用した作業 13
データベースおよびアプリケーション 14
Campaign システム・テーブル 15
ランタイム・テーブル. 15
テスト実行テーブル 16
動的に作成されたテーブルに使用されるデフォルト・データ型のオーバーライド. 17
デフォルトのデータ型をオーバーライドするには 18
動的に作成されたテーブルのデフォルトのデータ型 18
プロファイル・データベース 19
学習テーブル. 21
クロスセッション・レスポンス・トラッキングのコンタクト履歴. 22
Interact 機能スクリプトの使用 22
コンタクトとレスポンスの履歴のトラッキングについて. 22
コンタクトとレスポンスのタイプの構成. . . . 23
追加のレスポンス・タイプ 23
ランタイム環境のステージング・テーブルから Campaign の履歴テーブルへのマッピング . . . 25
コンタクトとレスポンスの履歴モジュール用に JMX 監視を構成するには. 27
クロスセッション・レスポンス・トラッキングについて. 28
クロスセッション・レスポンス・トラッキングのデータ・ソース構成 29
クロスセッション・レスポンス・トラッキング用のコンタクトおよびレスポンス履歴テーブルの構成 29
クロスセッション・レスポンス・トラッキングを有効にするには 32
クロスセッション・レスポンス・オファーの照合 32
ランタイム環境でのデータベース・ロード・ユーティリティーの使用 35
ランタイム環境でデータベース・ロード・ユーティリティーを使用可能にするには 36

第 4 章 オファーの提供. 39

オファーの適格性 39
候補オファーのリストの生成 39
マーケティング・スコアの計算. 41
学習の影響 41
オファーの非表示について 42
オファー非表示テーブルを有効にするには . . . 42
オファー非表示テーブル 42
グローバル・オファーと個別の割り当て. 43
デフォルトのセル・コードを定義するには . . . 43
UACI_ICBatchOffers テーブルを定義するには . . 44
グローバル・オファー・テーブルについて . . . 44
グローバル・オファー・テーブルを有効にするには 45
グローバル・オファー・テーブル 45
スコア・オーバーライド・テーブルについて . . 47
スコア・オーバーライド・テーブルを有効にするには. 48
スコア・オーバーライド・テーブル 48
Interact 組み込み学習の概要. 51
Interact の学習の理解 51
学習モジュールを有効にするには 53
学習属性 53
学習属性を定義するには 54
動的学習属性を定義するには 55
外部学習を有効にするには 56

第 5 章 Interact API の理解 57

Interact API データ・フロー. 57
単純なインタラクション計画の例 61
Interact API 統合の設計 64
考慮事項 65

第 6 章 IBM Unica Interact API の管理 67

ロケールと Interact API 67
JMX 監視について 67
RMI プロトコルを使用した JMX 監視を使用するように Interact を構成するには. 68

JMXMP プロトコルを使用した JMX 監視を使用するように Interact を構成するには 68
jconsole スクリプトの使用 68
JMX 属性 69
JMX 操作 77

第 7 章 IBM Unica Interact API のクラスとメソッド 79
Interact API クラス 79
HTTP における Java 直列化の前提条件 79
SOAP の前提条件 80
API JavaDoc 80
API の例について 80
セッション・データを使用した作業 81
InteractAPI クラスについて 82
endSession 82
executeBatch 83
getInstance 84
getOffers 85
getOffersForMultipleInteractionPoints 86
getProfile 88
getVersion 89
postEvent 90
setAudience 92
setDebug 94
startSession 95
予約パラメーター 98
AdvisoryMessage クラスについて 100
getDetailMessage 100
getMessage 101
getMessageCode 101
getStatusLevel 102
AdvisoryMessageCode クラスについて 102
アドバイザリー・メッセージ・コード 102
BatchResponse クラスについて 104
getBatchStatusCode 104
getResponses 105
コマンド・インターフェースについて 106
setAudienceID 106
setAudienceLevel 107
setDebug 108
setEvent 108
setEventParameters 109
setGetOfferRequests 110
setInteractiveChannel 111
setInteractionPoint 111
setMethodIdentifier 112
setNumberRequested 112
setRelyOnExistingSession 113
NameValuePair インターフェースについて 113
getName 113
getValueAsDate 114
getValueAsNumeric 114
getValueAsString 115
getValueDataType 115
setName 116
setValueAsDate 116
setValueAsNumeric 117
setValueAsString 117
setValueDataType 117
オファー・クラスについて 118
getAdditionalAttributes 118
getDescription 119
getOfferCode 119
getOfferName 120
getScore 120
getTreatmentCode 121
OfferList クラスについて 121
getDefaultString 121
getRecommendedOffers 122
レスポンス・クラスについて 123
getAdvisoryMessages 123
getApiVersion 123
getOfferList 124
getAllOfferLists 124
getProfileRecord 125
getSessionID 125
getStatusCode 126

第 8 章 ExternalCallout API について 127
IAffiniumExternalCallout インターフェース 127
EXTERNALCALLOUT で使用する Web サービスを追加するには 128
getNumberOfArguments 128
getValue 128
initialize 129
shutdown 130
ExternalCallout API の例 130
IInteractProfileDataService インターフェース . . . 131
プロファイル・データ・サービスで使用するデータ・ソースを追加するには 132

第 9 章 IBM Unica Interact ユーティリティー 133
配置実行ユーティリティー (runDeployment.sh/.bat) 133

第 10 章 学習 API について 137
外部学習を有効にするには 138
ILearning インターフェース 139
initialize 139
logEvent 139
optimizeRecommendList 140
reinitialize 141
shutdown 141
IAudienceID インターフェース 142
getAudienceLevel 142
getComponentNames 142
getComponentValue 143
IClientArgs 143
getValue 143
IInteractSession 143
getAudienceId 143

getSessionData 144
IInteractSessionData インターフェース 144
getDataType 144
getParameterNames 144
getValue 144
setValue 145
ILearningAttribute 145
getName 145
ILearningConfig 145
ILearningContext 146
getLearningContext 146
getResponseCode 146
IOffer 147
getCreateDate. 147
getEffectiveDateFlag 147
getExpirationDateFlag 147
getOfferAttributes 147
getOfferCode 148
getOfferDescription 148
getOfferID 148
getOfferName 148
getUpdateDate 148
IOfferAttributes 149
getParameterNames 149
getValue 149
IOfferCode インターフェース 149
getPartCount 149
getParts 150
LearningException 150
IScoreOverride 150
getOfferCode 150
getParameterNames 150
getValue 151
ISelectionMethod 151
ITreatment インターフェース 151
getCellCode 152
getCellId 152
getCellName 152
getLearningScore 152
getMarketerScore 153
getOffer 153
getOverrideValues 153
getPredicate 153
getPredicateScore 154
getScore 154
getTreatmentCode 154
setActualValueUsed 154
学習 API の例 155

付録 A. IBM Unica Interact WSDL 159

付録 B. Interact ランタイム環境の構成プロパティー 167

Interact | 全般 167
Interact | 全般 | learningTablesDataSource . . . 167
Interact | 全般 | prodUserDataSource 169
Interact | 全般 | systemTablesDataSource 171
Interact | 全般 | testRunDataSource 176
Interact | 全般 | contactAndResponseHistoryDataSource 178
Interact | 全般 | idsByType 179
Interact | フローチャート 180
Interact | フローチャート | ExternalCallouts | [ExternalCalloutName] 181
Interact | フローチャート | ExternalCallouts | [ExternalCalloutName] | パラメーター・データ (Parameter Data) | [parameterName] 182
Interact | 監視 182
Interact | プロファイル 183
Interact | プロファイル | オーディエンス・レベル | [AudienceLevelName] 185
Interact | プロファイル | オーディエンス・レベル | [AudienceLevelName] | 未加工 SQL によるオファー (Offers by Raw SQL) 188
Interact | プロファイル | オーディエンス・レベル | [AudienceLevelName] | プロファイル・データ・サービス | [DataSource] 190
Interact | offerserving 191
Interact | offerserving | 組み込み学習の構成 (Built-in Learning Config) 192
Interact | offerserving | 外部学習構成 (External Learning Config) 193
Interact | offerserving | 外部学習構成 (External Learning Config) | パラメーター・データ (Parameter Data) | [parameterName] 194
Interact | サービス (services) 194
Interact | サービス (services) | contactHist . . . 194
Interact | サービス (services) | contactHist | キャッシュ. 195
Interact | サービス (services) | contactHist | fileCache 196
Interact | サービス (services) | defaultedStats . . 196
Interact | サービス (services) | defaultedStats | キャッシュ 196
Interact | サービス (services) | eligOpsStats . . . 197
Interact | サービス (services) | eligOpsStats | キャッシュ 197
Interact | サービス (services) | eventActivity . . 198
Interact | サービス (services) | eventActivity | キャッシュ 198
Interact | サービス (services) | customLogger . . 198
Interact | サービス (services) | customLogger | キャッシュ 199
Interact | サービス (services) | responseHist. . . 199
Interact | サービス (services) | responseHist | キャッシュ 200
Interact | サービス (services) | responseHist | fileCache 200
Interact | サービス (services) | crossSessionResponse 201
Interact | サービス (services) | crossSessionResponse | キャッシュ 202

Interact | サービス (services) | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode 202
Interact | サービス (services) | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode . 204
Interact | サービス (services) | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode 205
Interact | サービス (services) | threadManagement | contactAndResponseHist 206
Interact | サービス (services) | threadManagement | allOtherServices 207
Interact | サービス (services) | threadManagement | flushCacheToDB 208
Interact | sessionManagement. 209

付録 C. Interact 設計環境の構成プロパティー 211

Campaign | パーティション | パーティション[n] | レポート 211
Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking 213
Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource] . . . 217
Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking | contactTypeMappings 218
Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking | responseTypeMappings. 219
Campaign | パーティション | パーティション[n] | Interact | レポート. 220
Campaign | パーティション | パーティション[n] | Interact | 学習 220
Campaign | パーティション | パーティション[n] | Interact | 学習 | learningAttributes | [learningAttribute] 224
Campaign | パーティション | パーティション[n] | Interact | 配置 224
Campaign | パーティション | パーティション[n] | Interact | serverGroups | [serverGroup] 224
Campaign | パーティション | パーティション[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL] 225
Campaign | パーティション | パーティション[n] | Interact | フローチャート 225
Campaign | パーティション | パーティション[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers . 226
Campaign | パーティション | パーティション[n] | Interact | whiteList | [AudienceLevel] | offersBySQL . 226
Campaign | パーティション | パーティション[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride. 227
Campaign | パーティション | パーティション[n] | サーバー (server) | 内部 (internal). 227
Campaign | 監視 230

付録 D. クライアント・サイドでのリアルタイム・オファーのカスタマイズ. . . 233

Interact Message Connector について. 233
Message Connector のインストール 234
Message Connector リンクの作成 241
Interact Web Connector について 244
ランタイム・サーバーへの Web Connector のインストール 245
別個の Web アプリケーションとしての Web Connector のインストール 246
Web Connector の構成 247
Web Connector 管理ページの使用. 260
Web Connector のサンプル・ページ 261

付録 E. Interact と Intelligent Offer の統合製品の推奨 265

Interact と Intelligent Offer の統合の概要 265
統合の前提条件. 266
Intelligent Offer 統合のためのオファーの構成. . . 267
統合サンプル・プロジェクトの使用 268

IBM Unica 技術サポートへの連絡 . . . 275

特記事項. 277

商標 279

第 1 章 IBM Unica Interact の管理

Interact の管理は、複数の作業で構成されています。以下のような作業があります (以下に限定されるわけではありません)。

- ユーザーおよび役割の保守
- データ・ソースの保守
- Interact のオファー配信機能 (オプション) の構成
- ランタイム環境のパフォーマンスのモニターと保守

Interact の管理を開始する前に、作業を容易にするために、Interact の処理に関するいくつかの主要な概念について、理解しておく必要があります。この後のセクションでは、Interact に関連する管理作業について説明します。

このガイドの 2 番目のパートでは、Interact で使用可能なアプリケーション・プログラミング・インターフェース (API) (Interact API、ExternalCallout API、および学習 API) について説明します。

Interact キーの概念

このセクションでは、Interact を使用して作業を行う前に理解しておく必要がある、いくつかのキーの概念について説明します。

オーディエンス・レベル

オーディエンス・レベルは、キャンペーンのターゲットにできる ID の集合です。例えば、一連のキャンペーンでは、オーディエンス・レベルとして、「世帯」、「見込み客」、「顧客」、「アカウント」などを使用できます。これらの各レベルは、キャンペーンで使用可能なマーケティング・データの特定の視点を表すものです。

オーディエンス・レベルは、通常は階層として編成されます。上記の例を使用すると、次のようになります。

- 「世帯」は階層の最上位にあり、各世帯には、複数の顧客と 1 人以上の見込み客を含めることができます。
- 「顧客」は階層の次の段階にあり、それぞれの顧客は複数のアカウントを持つことができます。
- 「アカウント」は、階層の最下位にあります。

その他、より複雑なオーディエンス階層の例としては、企業間取引の環境があります。その場合にはオーディエンス・レベルとして、業種、企業、部署、グループ、個人、アカウントなどが必要になるかもしれません。

これらのオーディエンス・レベルには、互いに「1 対 1」、「多対 1」、「多対多」などの異なる関係が存在する場合があります。オーディエンス・レベルを定義すると、このような概念を Campaign で表すことができるので、ユーザーは、ターゲティングで利用するためにこれら異なるオーディエンス間の関係を管理できま

す。例えば、1 つの世帯に複数の見込み客がいる場合には、メール配信を各世帯につき 1 人の見込み客だけに限定することもできます。

設計環境

Interact の構成のほとんどは、設計環境で行います。設計環境では、イベント、インタラクション・ポイント、スマート・セグメント、および処理ルールを定義します。これらのコンポーネントを構成したら、ランタイム環境に配置します。

設計環境は Campaign Web アプリケーションと共にインストールされます。

イベント

イベントとは訪問者が実行するアクションのことです。これにより、訪問者のセグメントへの分類、オファーの提示、またはデータのロギングなど、ランタイム環境におけるアクションがトリガーされます。

イベントは最初にインタラクティブ・チャネルに作成されてから、`postEvent` メソッドを使用する Interact API 呼び出しによってトリガーされます。イベントにより、Interact 設計環境に定義されている以下の 1 つ以上のアクションが行われる可能性があります。

- 再セグメントのトリガー
- オファー・コンタクトをログに記録
- オファー承認をログに記録
- オファー拒否をログに記録

また、イベントを使用して、テーブルへのデータのロギング、学習へのデータの組み込み、または個々のフローチャートのトリガーなど、`postEvent` メソッドで定義されるアクションをトリガーすることもできます。

イベントは、設計環境では便宜上、カテゴリーにまとめることができます。カテゴリーには、ランタイム環境では機能上の目的はありません。

インタラクティブ・チャネル

インタラクティブ・チャネルは、インターフェースの方式がインタラクティブ・ダイアログである場合の、Campaign におけるタッチポイントを表します。このソフトウェア表現は、インタラクティブ・マーケティングに関係のあるオブジェクト、データ、およびサーバー・リソースをすべて統合するために使用されます。

インタラクティブ・チャネルは、インタラクション・ポイントとイベントを定義するために使用するツールです。インタラクティブ・チャネルのレポートには、そのインタラクティブ・チャネルの「分析」タブからアクセスすることもできます。

インタラクティブ・チャネルには、運用ランタイム・サーバーとステージング・サーバーの割り当ても含まれます。運用ランタイム・サーバーとステージング・サーバーが 1 セットのみの場合にイベントおよびインタラクション・ポイントをまとめる場合、あるいは、顧客対応システムでイベントとインタラクション・ポイントを分ける場合に、インタラクティブ・チャネルをいくつか作成できます。

インタラクティブ・フローチャート

インタラクティブ・フローチャートは、Campaign バッチ・フローチャートに関連しますが、少し異なります。インタラクティブ・フローチャートでは、バッチ・フローチャートと同じ主要機能 (顧客をセグメントと呼ばれるグループに分ける) を実行します。ただし、インタラクティブ・フローチャートの場合、グループはスマート・セグメントとなります。Interact は、動作イベントまたはシステム・イベントで訪問者の再セグメントが必要であると示された場合に、これらのインタラクティブ・フローチャートを使用して、プロファイルをセグメントに割り当てます。

インタラクティブ・フローチャートには、バッチ・フローチャート・プロセスのサブセットと、少数のインタラクティブ・フローチャート固有のプロセスが含まれます。

注: インタラクティブ・フローチャートは、Campaign セッションでのみ作成できます。

インタラクション・ポイント

インタラクション・ポイントとは、オファーを提示するタッチポイントにある場所のことです。インタラクション・ポイントには、ランタイム環境に提示対象となる他のコンテンツがない場合の、デフォルトの充てんコンテンツが含まれます。

インタラクション・ポイントは複数のゾーンに分けることができます。

オファー

オファーとは、さまざまな方法で配信できる単一のマーケティング・メッセージのことです。

Campaign では、1 つ以上のキャンペーンで使用できるオファーを作成します。

オファーは以下のように再使用可能です。

- さまざまなキャンペーンにおいて
- さまざまな時点で
- さまざまな担当者グループ (セル) に対して
- さまざまな「バージョン」として (オファーのパラメーター化された項目を変更)

フローチャート内のターゲット・セルには、コンタクト・プロセスのいずれかを使用してオファーを割り当て、オファーを受けた顧客とそのオファーに応じた顧客に関するデータを取り込んでキャンペーン結果をトラッキングします。

プロファイル

プロファイルは、ランタイム環境で使用される顧客データ・セットです。このデータは、顧客データベースで使用可能な顧客データのサブセット、またはリアルタイムで収集されるデータ、あるいはこの 2 つを組み合わせたものにすることができます。このデータは次の目的に使用されます。

- リアルタイム・インタラクション・シナリオで 1 つ以上のスマート・セグメントに顧客を割り当てる。

セグメント化で使用するオーディエンス・レベルごとにプロファイル・データ・セットが必要です。例えば、場所でセグメント化する場合、所有しているすべての顧客の住所情報から郵便番号のみを含めることができます。

- オファーをカスタマイズする。
- 学習用にトラッキングする属性として。

  例えば、訪問者の婚姻区分、および特定のオファーを受け入れる各区分の訪問者数をモニターするように、Interact を構成できます。これで、ランタイム環境でその情報を使用して、オファーの選択を絞り込むことができます。

このデータは、ランタイム環境では読み取り専用です。

ランタイム環境

ランタイム環境はタッチポイントに接続され、インタラクションを行います。ランタイム環境は、タッチポイントに接続された 1 つ以上のランタイム・サーバーで構成できます。

ランタイム環境では、設計環境から入手した情報を Interact API と併用して、オファーをタッチポイントに提示します。

ランタイム・セッション

ランタイム・セッションは、タッチポイントへの訪問者ごとにランタイム・サーバー上に存在します。このセッションでは、ランタイム環境での訪問者のセグメントへの割り当ておよびオファーの推奨に使用する、訪問者のすべてのデータを保持します。

ランタイム・セッションは、`startSession` 呼び出しの使用時に作成できます。

タッチポイント

タッチポイントとは、顧客と対話できるアプリケーションまたは場所のことです。タッチポイントには、顧客がコンタクトを開始する (「インバウンド」インタラクション) チャネルや、顧客にコンタクトを取る (「アウトバウンド」インタラクション) チャネルがあります。その一般的な例として、Web サイトやコール・センター・アプリケーションがあります。 Interact API を使用すれば、Interact をタッチポイントと統合し、顧客にタッチポイントでのアクションに応じてオファーを提示できます。タッチポイントは顧客対応システム (CFS) ともいいます。

処理ルール

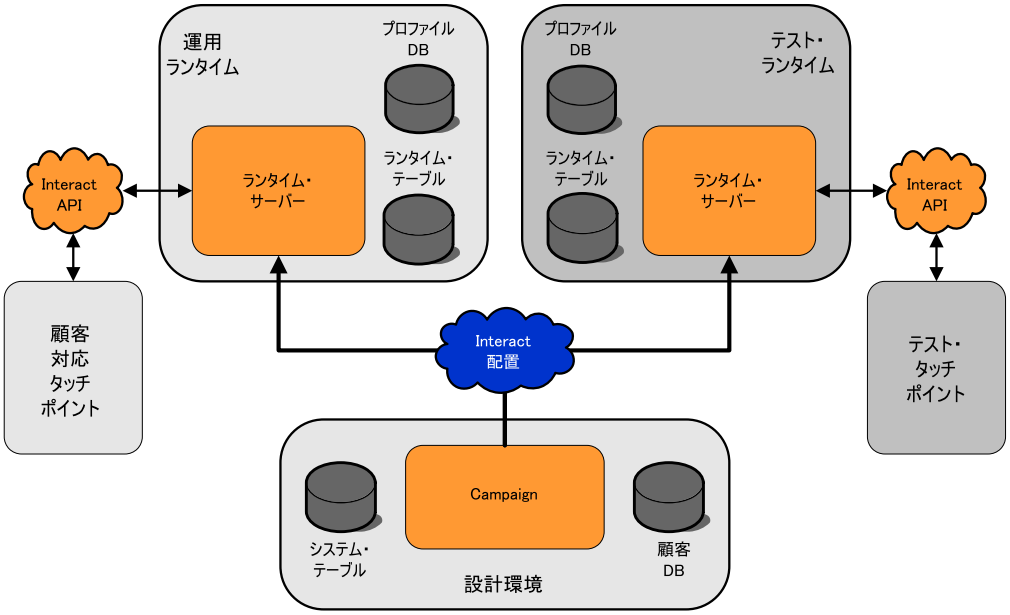
処理ルールに従って、オファーをスマート・セグメントに割り当てます。これらの割り当ては、処理ルールでオファーに関連付けられる、顧客定義ゾーンによってさらに制約されます。例えば、「ログイン」ゾーンにスマート・セグメントを割り当てる 1 つのオファー・セットがあり、「購入後」ゾーンに同じセグメントの異なるオファー・セットがある場合があります。処理ルールは、キャンペーンのインタラクション方法タブに定義されます。

各処理ルールにはマーケティング・スコアも含まれます。顧客が複数のセグメントに割り当てられているため、複数のオファーが適用可能な場合に、Interact がどのオファーを推奨するかを定義する際にマーケティング・スコアが役立ちます。ランタ

イム環境でどのオファーを推奨するかは、学習モジュール、オファー非表示リスト、およびグローバル・オファー割り当てと個々のオファー割り当てに影響を受ける可能性があります。

Interact アーキテクチャー

Interact 環境は、最低 2 つの主要なコンポーネント (設計環境とランタイム環境) で構成されます。オプションで、テスト・ランタイム・サーバーも含まれる場合があります。次の図は、ハイレベルなアーキテクチャーの概要を示しています。



Interact の構成のほとんどは、設計環境で行います。設計環境は、Campaign Web アプリケーションとともにインストールされ、Campaign システム・テーブルとご使用の顧客データベースを参照します。設計環境を使用して、その API で使用するインタラクション・ポイントおよびイベントを定義します。

ランタイム環境で顧客との対話を処理する方法を設計および構成したら、そのデータをテスト用にステージング・サーバー・グループに配置するか、あるいは顧客とのリアルタイムでの対話用に運用ランタイム・サーバー・グループに配置します。

Interact API は、タッチポイントとランタイム・サーバーの間の接続を提供します。ユーザーは、設計環境で Interact API を使用して作成されたオブジェクト (インタラクション・ポイントおよびイベント) を参照し、それらを使用してランタイム・サーバーにある情報を要求します。

Interact ネットワークに関する考慮事項

Interact の運用インストールは、最低でも 2 つのマシンに対して行われます。複数の Interact ランタイム・サーバーと分散データベースが含まれるようなボリュームの大きな運用環境では、数十台のマシンにインストールされる場合もあります。最良のパフォーマンスを得るために、ネットワーク・トポロジーに関して考慮の必要な要件がいくつか存在します。

- 例えば次のように、Interact API のセッションの開始と終了が同じ呼び出しの中で行われる実装環境の場合:

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

  ロード・バランサーと Interact ランタイム・サーバーの間のセッション・パーシスタンス (スティッキー・セッション) を有効にする必要はありません。Interact ランタイム・サーバーのセッション管理をローカル・キャッシュ・タイプ用に構成することができます。
- 例えば次のように、Interact API のセッションの開始と終了に、複数の呼び出しが使用される実装環境の場合:

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

  Interact ランタイム・サーバーでロード・バランサーを使用しているのであれば、そのロード・バランサー (スティッキー・セッションとも呼ばれます) に対していくつかのタイプのパーシスタンスを有効にする必要があります。それが可能でない場合、あるいはロード・バランサーを使用していない場合は、Interact サーバーのセッション管理を、分散 cacheType 用に構成します。分散キャッシュを使用している場合は、すべての Interact ランタイム・サーバーがマルチキャストを使用して通信できるようにする必要があります。同じマルチキャスト IP アドレスとポートを使用している Interact サーバー間の通信によってシステム・パフォーマンスが低下することがないように、ネットワークをチューニングする必要がある場合もあります。スティッキー・セッションを使用するロード・バランサーの方が、分散キャッシュを使用するよりも良いパフォーマンスを得られます。
- 分散 cacheType を使用しているサーバー・グループが複数ある場合は、それぞれ固有のマルチキャスト・ポートを使用する必要があります。サーバー・グループごとに固有のマルチキャスト・ポートおよびアドレスを使用するとなお良いでしょう。
- 最良のパフォーマンスを得るには、ランタイム環境の Interact サーバー、Marketing Platform、ロード・バランサー、およびタッチポイントを地理的に同じ場所に置いてください。設計時とランタイムで地理的に別の場所を使用することができますが、配置が遅くなることが予測されます。
- Interact の運用サーバー・グループとそれに関連するタッチポイントの間には、高速のネットワーク接続 (最低 1 GB) を使用します。
- 配置作業を完了するには、設計時からランタイムへのアクセスに http または https が必要です。配置を可能にするには、任意のファイアウォールか、またはその他のネットワーク・アプリケーションを構成する必要があります。配置の規模が大きい場合は、設計環境とランタイム環境の間の HTTP タイムアウトの長さを拡張する必要がある場合もあります。
- コンタクトとレスポンスの履歴モジュールには、設計時データベース (Campaign システム・テーブル) へのアクセス権とランタイム・データベース (Interact ランタイム・テーブル) へのアクセス権が必要です。このデータ転送が行われるようにするには、データベースとネットワークを適切に構成する必要があります。

テストまたはステージング・インストールでは、設計時とランタイムの Interact を同じマシンにインストールすることができます。このシナリオは、運用環境では推奨されません。

第 2 章 IBM Unica Interact ユーザーの構成

Interact では、ランタイム環境ユーザーと設計環境ユーザーの 2 セットのユーザーの構成が必要です。

- **ランタイム・ユーザー**は、ランタイム・サーバーで作業するように構成された Marketing Platform で作成されます。
- **設計時ユーザー**は、Campaign ユーザーです。設計チームのさまざまなメンバーのセキュリティーを、Campaign の場合と同様に構成します。

ランタイム環境ユーザーの構成

Interact をインストールしたら、最低 1 人の Interact ユーザー (ランタイム環境ユーザー) を構成する必要があります。

ランタイム環境ユーザーには、ランタイム・テーブルへのアクセス権があります。このユーザー名とパスワードは、インタラクティブ・チャネルの配置時に使用します。ランタイム・サーバーは、データベースの資格情報に対して Web アプリケーション・サーバーの JDBC 認証を使用するため、ランタイム環境ユーザーにランタイム環境のデータ・ソースを追加する必要はありません。

重要: 同じサーバー・グループに属するすべてのランタイム・サーバーが、同じユーザー資格情報を共有する必要があります。ランタイム・サーバーごとに別個の Marketing Platform インスタンスがある場合、それぞれに同じユーザーおよびパスワードを作成する必要があります。

データベース・ロード・ユーティリティーを使用している場合は、ランタイム・テーブルを、ランタイム環境ユーザーのログイン資格情報を持つデータ・ソースとして定義する必要があります。このデータ・ソースの名前は、`systemTablesDataSource` にする必要があります。

JMXMP プロトコルを使用する JMX 監視のセキュリティーを有効にする場合、JMX 監視セキュリティー用に別のユーザーが必要になる場合があります。

設計環境ユーザーの構成

設計環境ユーザーは、Campaign ユーザーです。設計環境ユーザーは、Campaign の役割のアクセス権を構成するのと同じ方法で構成します。

インタラクティブ・フローチャートの編集権限を持つ任意の Campaign ユーザーに、テスト実行テーブルのデータ・ソースへのアクセス権を付与します。

Interact がインストールおよび構成されている場合、デフォルトのグローバル・ポリシーおよび新規ポリシーに対して以下の追加オプションを使用することができます。一部の設計環境ユーザーには、一部の Campaign 権限 (カスタム・マクロなど) も必要ですので注意してください。

カテゴリー	権限
キャンペーン	• キャンペーン・インタラクション方法の表示 — キャンペーンのインタラクション方法タブを表示することができます。ただし、編集することはできません。 • キャンペーン・インタラクション方法の編集 — インタラクション方法タブに対して変更を加えることができます (処理ルールを含む)。 • キャンペーン・インタラクション方法の削除 — インタラクション方法タブをキャンペーンから削除することができます。インタラクティブ・チャネルの配置にインタラクション方法が組み込まれている場合、インタラクション方法タブの削除は制限されます。 • キャンペーン・インタラクション方法の追加 — 新規インタラクション方法タブをキャンペーンに作成することができます。 • キャンペーン・インタラクション方法配置の開始 — インタラクション方法タブに配置または配置解除のマークを付けることができます。
インタラクティブ・チャネル	• インタラクティブ・チャネルの配置 — インタラクティブ・チャネルを Interact ランタイム環境に配置することができます。 • インタラクティブ・チャネルの編集 — インタラクティブ・チャネルの「サマリー」タブに変更を加えることができます。 • インタラクティブ・チャネルの削除 — インタラクティブ・チャネルを削除することができます。インタラクティブ・チャネルが配置されている場合、インタラクティブ・チャネルの削除は制限されます。 • インタラクティブ・チャネルの表示 — インタラクティブ・チャネルを表示することができます。ただし、編集することはできません。 • インタラクティブ・チャネルの追加 — 新規インタラクティブ・チャネルを作成することができます。 • インタラクティブ・チャネル・レポートの表示 — インタラクティブ・チャネルの「分析」タブを表示することができます。 • インタラクティブ・チャネルの子オブジェクトの追加 — インタラクション・ポイント、ゾーン、イベント、およびカテゴリーを追加することができます。

カテゴリー	権限
セッション	• インタラクティブ・フローチャートの表示 — インタラクティブ・フローチャートをセッションに表示することができます。 • インタラクティブ・フローチャートの追加 — 新規インタラクティブ・フローチャートをセッションに作成することができます。 • インタラクティブ・フローチャートの編集 — インタラクティブ・フローチャートに変更を加えることができます。 • インタラクティブ・フローチャートの削除 — インタラクティブ・フローチャートを削除することができます。インタラクティブ・フローチャートが割り当てられているインタラクティブ・チャネルが配置されている場合、そのインタラクティブ・フローチャートの削除は制限されます。 • インタラクティブ・フローチャートのコピー — インタラクティブ・フローチャートをコピーすることができます。 • インタラクティブ・フローチャートのテスト実行 — インタラクティブ・フローチャートのテスト実行を開始することができます。 • インタラクティブ・フローチャートの確認 — インタラクティブ・フローチャートを表示したり、設定を表示するためにプロセスを開いたりすることができます。ただし、変更を加えることはできません。 • インタラクティブ・フローチャートの配置 — インタラクティブ・フローチャートに配置または配置解除のマークを付けることができます。

設計環境のアクセス権の例

例えば、2 つの役割を作成できます。1 つはインタラクティブ・フローチャートの作成者のための役割で、1 つは対話戦略の定義者のための役割です。各セクションに、その役割に付与されるアクセス権をリストします。

インタラクティブ・フローチャートの役割

カスタム・マクロ

- カスタム・マクロの追加
- カスタム・マクロの編集
- カスタム・マクロの使用

ユーザー定義項目

- ユーザー定義項目の追加
- ユーザー定義項目の編集
- ユーザー定義項目の使用

フローチャート・テンプレート

- テンプレートの貼り付け

セグメント・テンプレート

- セグメントの追加
- セグメントの編集

セッション

- セッション・サマリーの表示
- インタラクティブ・フローチャートの表示
- インタラクティブ・フローチャートの追加
- インタラクティブ・フローチャートの編集
- インタラクティブ・フローチャートのコピー
- インタラクティブ・フローチャートのテスト実行
- インタラクティブ・フローチャートの展開

対話戦略の役割

Campaign

- キャンペーン要約の表示
- キャンペーン・ターゲット・セルの管理
- キャンペーン対話戦略の表示
- キャンペーン対話戦略の編集
- キャンペーン対話戦略の追加
- キャンペーン対話戦略展開の開始

オファー

- オファーの要約の表示

セグメント・テンプレート

- セグメント・サマリーの表示

セッション

- インタラクティブ・フローチャートのレビュー

第 3 章 Interact データ・ソースの管理

Interact を正常に機能させるにはデータ・ソースがいくつか必要になります。これらのデータ・ソースには、Interact を機能させるために必要な情報を含むものや、ユーザーのデータを含むものがあります。

以下のセクションでは、Interact データ・ソースについて説明します。この説明にはデータ・ソースを適切に構成するために必要な情報、およびデータ・ソースを維持するためのいくつかの推奨事項が含まれます。

Interact データ・ソースを使用した作業

Interact が機能するには、いくつかのデータのセットが必要です。

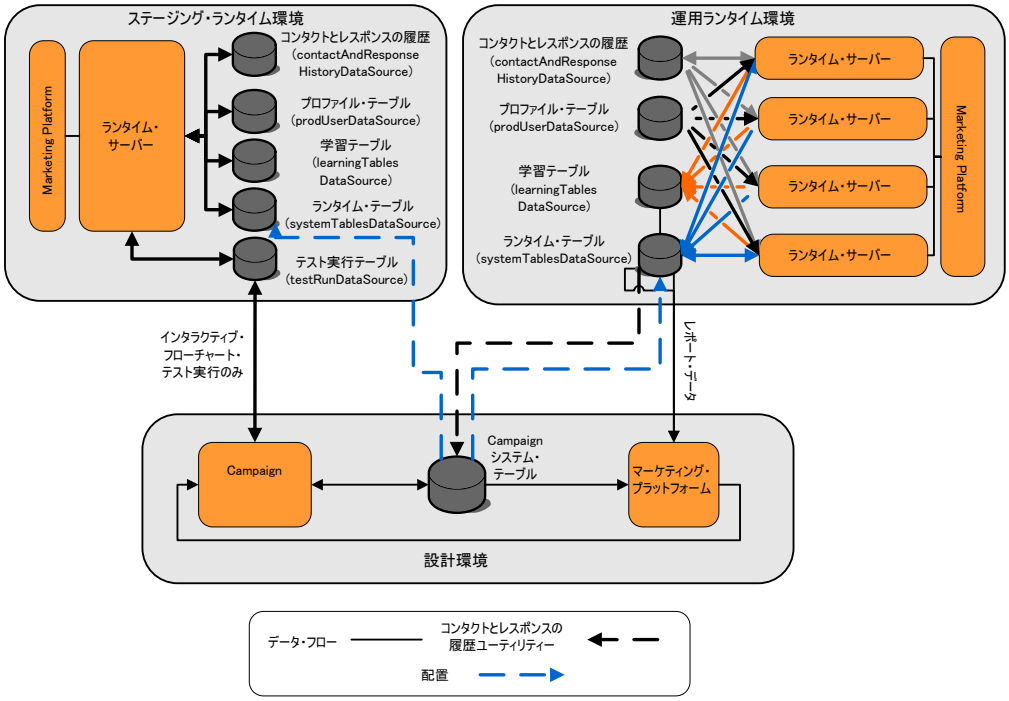
- **Campaign システム・テーブル** — Campaign のすべてのデータだけでなく、Campaign システム・テーブルには、設計環境で作成する Interact コンポーネント (処理ルールやインタラクティブ・チャネルなど) のデータが含まれます。設計環境と Campaign システム・テーブルは、同じ物理データベースおよびスキーマを共有します。
- **ランタイム・テーブル** — (`systemTablesDataSource`) には、設計環境からの配置データ、コンタクトとレスポンスの履歴のステージング・テーブル、およびランタイム統計が含まれます。
- **プロファイル・テーブル** — (`prodUserDataSource`) には、インタラクティブ・フローチャートが訪問者を適切なスマート・セグメントに配置するために必要とするすべての顧客データ (リアルタイムで収集された情報だけではない) が含まれます。完全にリアルタイムのデータに依存している場合、プロファイル・テーブルは必要ありません。プロファイル・テーブルを使用している場合は、インタラクティブ・チャネルによって使用されるオーディエンス・レベルごとに最低 1 つのプロファイル・テーブルを保持する必要があります。

  プロファイル・テーブルには、補完オファー・サービスに使用されるテーブル (オファー非表示、スコア・オーバーライド、およびグローバル・オファーと個々のオファーの割り当てのためのテーブルなど) も含めることができます。
- **テスト実行テーブル** — (`testRunDataSource`) には、インタラクティブ・フローチャートが訪問者をスマート・セグメントに配置するために必要とするすべてのデータのサンプル (対話中にリアルタイムで収集されるデータの例など) が含まれます。これらのテーブルが必要なのは、設計環境のテスト実行サーバー・グループとして指定されているサーバー・グループのみです。
- **学習テーブル** — (`learningTablesDataSource`) には、組み込み学習ユーティリティーによって収集されるすべてのデータが含まれます。これらのテーブルには、動的属性を定義するテーブルが含まれる場合があります。学習を使用していない場合、あるいは作成した外部の学習ユーティリティーを使用している場合、学習テーブルは必要ありません。
- **クロス・セッション・レスポンスのコンタクトとレスポンスの履歴** — (`contactAndResponseHistoryDataSource`) は、Campaign のコンタクト履歴テーブ

ルか、またはそれらのコピーです。クロス・セッション・レスポンス機能を使用していない場合、これらのコンタクト履歴テーブルを構成する必要はありません。

データベースおよびアプリケーション

以下の図は、考えられる Interact データ・ソースとそれらが IBM® Unica アプリケーションにどのように関連しているかを示しています。



- テスト実行テーブルには、Campaign およびテスト実行サーバー・グループの両方からアクセスされます。
- テスト実行テーブルは、インタラクティブ・フローチャートのテスト実行にのみ使用されます。
- Interact API を含む、配置のテストにランタイム・サーバーを使用する場合、ランタイム・サーバーはデータのプロファイル・テーブルを使用します。
- コンタクトおよびレスポンス履歴モジュールを構成すると、モジュールはバックグラウンド ETL (Extract、Transform、Load) プロセスを使用して、データをランタイム・ステージング・テーブルから Campaign コンタクトおよびレスポンス履歴テーブルに移動させます。
- レポート機能により、学習テーブル、ランタイム・テーブル、および Campaign のシステム・テーブルのデータが照会され、Campaign にレポートが表示されます。

運用ランタイム環境とは異なるテーブル・セットを使用するようにテスト・ランタイム環境を構成する必要があります。ステージング・テーブルと運用テーブルを区別することで、テスト結果を実際の結果と区別することができます。コンタクトおよびレスポンス履歴モジュールは常にデータを実際の Campaign コンタクトおよびレスポンス履歴テーブル (Campaign にはテスト用のコンタクトおよびレスポンス履

歴テーブルはありません) に挿入することに注意してください。テスト・ランタイム環境用の別個の学習テーブルがあり、レポートに結果を表示する場合は、テスト環境用の、学習レポートを実行する IBM Cognos® BI の別個のインスタンスが必要です。

Campaign システム・テーブル

設計環境をインストールするときには、Campaign システム・テーブル内に、Interact 固有の新規テーブルも作成します。

コンタクトとレスポンスの履歴モジュールを有効にすると、このモジュールは、コンタクトとレスポンスの履歴を、ランタイム・テーブル内のステージング・テーブルから Campaign システム・テーブル内のコンタクトとレスポンスの履歴テーブルにコピーします。デフォルトのテーブルは UA_ContactHistory、UA_DtlContactHist、および UA_ResponseHistory ですが、コンタクトとレスポンスの履歴モジュールは、そのコンタクトとレスポンスの履歴テーブル用に Campaign でマップされるテーブルを使用します。

グローバル・オファー・テーブルとスコア・オーバーライド・テーブルを使用してオファーを割り当てる場合で、そのインタラクティブ・チャネルの処理ルールに含まれていないオファーを使用している場合は、Campaign システム・テーブル内の UACI_ICBatchOffers テーブルへの入力が必要になる可能性があります。

ランタイム・テーブル

複数のオーディエンス・レベルがある場合は、オーディエンス・レベルごとにコンタクトおよびレスポンス履歴データのステージング・テーブルを作成する必要があります。

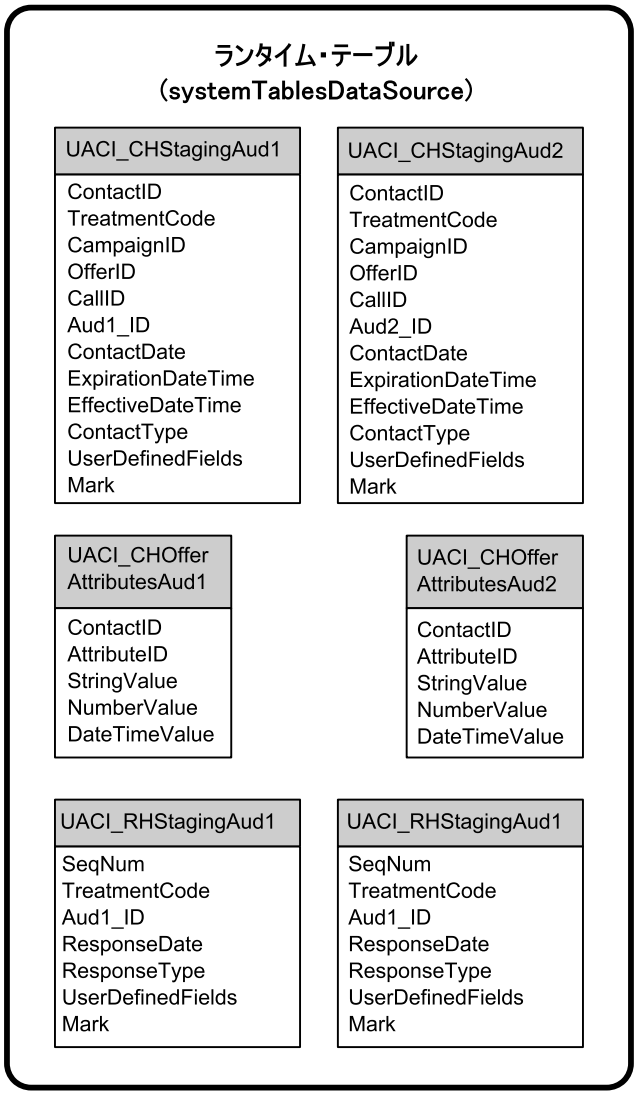
SQL スクリプトを実行すると、デフォルトのオーディエンス・レベルの以下のテーブルが作成されます。

- UACI_CHStaging
- UACI_CHOfferAttrib
- UACI_RHStaging

ランタイム・テーブルには、オーディエンス・レベルごとにこれら 3 つのテーブルのコピーを作成する必要があります。

Campaign のコンタクトおよびレスポンス履歴テーブルにユーザー定義の項目がある場合は、UACI_CHStaging テーブルと UACI_RHStaging テーブルに同じ項目名とタイプを作成する必要があります。これらの項目には、セッション・データに同じ名前の名前と値のペアを作成することで、実行時にデータを追加できます。例えば、コンタクトおよびレスポンス履歴テーブルに catalogID 項目が含まれているとします。その場合、UACI_CHStaging テーブルと UACI_RHStaging テーブルの両方に catalogID 項目を追加する必要があります。その後、Interact API では、catalogID という名前と値のペアとしてイベント・パラメーターを定義することで、この項目にデータが追加されます。セッション・データは、プロファイル・テーブル、一時データ、学習、または Interact API で提供できます。

以下の図は、オーディエンス Aud1 および Aud2 のテーブル例を示したものです。この図には、ランタイム・データベース内のすべてのテーブルは含まれていません。



テーブル内の項目はすべて必須です。Campaign のコンタクトおよびレスポンス履歴テーブルと一致するように、`CustomerID` と `UserDefinedFields` を変更することができます。

テスト実行テーブル

テスト実行テーブルは、インタラクティブ・フローチャートのテスト実行にのみ使用されます。インタラクティブ・フローチャートのテスト実行では、セグメント・ロジックをテストする必要があります。テスト実行データベースは Interact インストール済み環境に 1 つ構成するだけで十分です。テスト実行テーブルはスタンドアロン・データベース内にある必要はありません。例えば、Campaign に顧客データ・テーブルを使用できます。

テスト実行テーブルに関連付けられているデータベース・ユーザーには、テスト実行結果テーブルを追加するための作成権限が必要です。

テスト実行データベースには、インタラクティブ・チャネルにマップされたすべてのテーブルを含める必要があります。

これらのテーブルには、インタラクティブ・フローチャートのテスト対象のシナリオを実行するためのデータを含める必要があります。例えば、インタラクティブ・フローチャートに、ボイス・メール・システムでの選択項目に基づいて、担当者を複数のセグメントに分けるためのロジックがある場合は、考えられるすべての選択項目に対して少なくとも 1 つの行を含める必要があります。Web サイトのフォームを使用するインタラクションを作成する場合は、欠落データまたは誤った形式のデータを示す行を含める必要があります。例えば、電子メール・アドレスの値に `name@domaincom` を使用します。

各テスト実行テーブルには、少なくとも適切なオーディエンス・レベルの ID リストと、使用する予定のリアルタイム・データを示す列を含める必要があります。テスト実行ではリアルタイム・データにアクセスできないため、予想されるあらゆるリアルタイム・データのサンプル・データを提供する必要があります。例えば、リアルタイムに収集できるデータ (属性 `lastPageVisited` に保管されている、最後にアクセスされた Web ページの名前、または属性 `shoppingCartItemCount` に保管されている、ショッピング・カートに入っているアイテムの数など) を使用する場合は、同じ名前で列を作成し、その列にサンプル・データを設定します。これにより、本来は動作的または文脈的であるフローチャート・ロジックの分岐のテスト実行が可能になります。

インタラクティブ・フローチャートのテスト実行は、大きなデータ・セットを使用する作業用に最適化されていません。インタラクション・プロセスでテスト実行に使用する行の数を制限できます。ただし、これにより、常に最初の行セットが選択されます。別の行セットが選択されるようにする場合は、テスト実行テーブルの別のビューを使用してください。

実行時のインタラクティブ・フローチャートのスループット・パフォーマンスをテストするには、テスト環境用のプロファイル・テーブルを含む、テスト・ランタイム環境を作成する必要があります。

実際には、テスト用に 3 つのテーブル・セット (インタラクティブ・フローチャートのテスト実行用のテスト実行テーブル、テスト・サーバー・グループ用のテスト・プロファイル・テーブル、および運用プロファイル・テーブル・セット) が必要になるかもしれません。

動的に作成されたテーブルに使用されるデフォルト・データ型のオーバーライド

Interact ランタイム環境では、2 つのシナリオ (フローチャートのテスト実行時、およびまだ存在しないテーブルへの書き込みを行うスナップショット・プロセスの実行時) において、テーブルが動的に作成されます。これらのテーブルを作成する場合、Interact はサポートされている各データベース・タイプのハードコーディングされたデータ型に依存します。

デフォルトのデータ型は、testRunDataSource または prodUserDataSource に uaci_column_types という名前の代替データ型のテーブルを作成することでオーバーライドできます。この追加テーブルにより、Interact はハードコーディングされたデータ型でカバーされないまれなケースに対応できるようになります。

uaci_column_types テーブルが定義されている場合、Interact は、テーブル生成に使用するデータ型として、メタデータを列に使用します。uaci_column_types テーブルが定義されていない場合、またはテーブルを読み取ろうとしたときになんらかの例外が発生した場合は、デフォルトのデータ型が使用されます。

始動時に、ランタイム・システムはまず uaci_column_types テーブルの testRunDataSource を確認します。uaci_column_types テーブルが testDataSource に存在しない場合、または prodUserDataSource が別のデータベース・タイプである場合、Interact はそのテーブルの prodUserDataSource を確認します。

デフォルトのデータ型をオーバーライドするには

動的に作成されたテーブルのデフォルトのデータ型をオーバーライドする場合は、以下のステップに従います。

1. 次のプロパティーを指定して、TestRunDataSource または ProdUserDataSource にテーブルを作成します。

   テーブル名: uaci_column_types

   列名:

   - uaci_float
   - uaci_number
   - uaci_datetime
   - uaci_string

   データベースでサポートされている適切なデータ型を使用して、各列を定義します。
2. ランタイム・サーバーを再始動して、Interact が新しいテーブルを認識できるようにします。

   重要: ランタイム・サーバーは、uaci_column_types テーブルに変更を加えるたびに再始動する必要があります。

動的に作成されたテーブルのデフォルトのデータ型

以下のテーブルには、Interact ランタイム・システムが、浮動小数、数値、日時および文字列の列のサポートされているデータベースにデフォルトで使用するハードコーディングされたデータ型がリストされています。

表 1. 動的に作成されたテーブルのデフォルトのデータ型

データベース	デフォルトのデータ型
DB2®	• float • bigint • timestamp • varchar
Informix®	• float • int8 • DATETIME YEAR TO FRACTION • char2
Oracle	• float • number(19) • timestamp • varchar2
SQL Server	• float • bigint • datetime • nvarchar

プロファイル・データベース

プロファイル・データベースの内容は、インタラクティブ・フローチャートと Interact API を構成するために必要なデータに完全に依存します。Interact は、各データベースに特定のテーブルまたはデータを含めることを要求、あるいは推奨します。

プロファイル・データベースには以下を含める必要があります。

- インタラクティブ・チャネルでマップされたすべてのテーブル。

  これらのテーブルには、インタラクティブ・フローチャートを運用で実行するために必要なすべてのデータを含める必要があります。これらのテーブルをフラット化し、簡素化して、適切に索引付けする必要があります。ディメンション・データへのアクセスにはパフォーマンス・コストが発生するため、できるだけ非正規化スキーマを使用する必要があります。最低でも、オーディエンス・レベル ID 項目のプロファイル・テーブルには索引を付ける必要があります。ディメンション・テーブルから取得された他の項目がある場合は、データベースからのフェッチ時間を短縮させるために、これらの項目に適切に索引を付ける必要があります。プロファイル・テーブルのオーディエンス ID は、Campaign に定義されているオーディエンス ID と一致する必要があります。
- enableScoreOverrideLookup 構成プロパティーを true に設定する場合は、少なくとも 1 つのオーディエンス・レベルのスコア・オーバーライド・テーブルを含める必要があります。このスコア・オーバーライド・テーブルの名前は scoreOverrideTable プロパティーを指定して定義します。

スコア・オーバーライド・テーブルには、個々の顧客とオファーのペアを含めることができます。サンプルのスコア・オーバーライド・テーブル UACI_ScoreOverride は、プロファイル・データベースに対して aci_usertab SQL スクリプトを実行することで作成できます。「オーディエンス ID」列のこのテーブルにも索引を付ける必要があります。

enableScoreOverrideLookup プロパティーを false に設定する場合は、スコア・オーバーライド・テーブルを含める必要はありません。

- enableDefaultOfferLookup 構成プロパティーを true に設定する場合は、グローバル・オファー・テーブル (UACI_DefaultOffers) を含める必要があります。グローバル・オファー・テーブルは、プロファイル・データベースに対して aci_usertab SQL スクリプトを実行することで作成できます。

  グローバル・オファー・テーブルにはオーディエンスとオファーのペアを含めることができます。

- enableOfferSuppressionLookup プロパティーを true に設定する場合は、少なくとも 1 つのオーディエンス・レベルのオファー非表示テーブルを含める必要があります。オファー非表示テーブルの名前は、offerSuppressionTable プロパティーを指定して定義します。

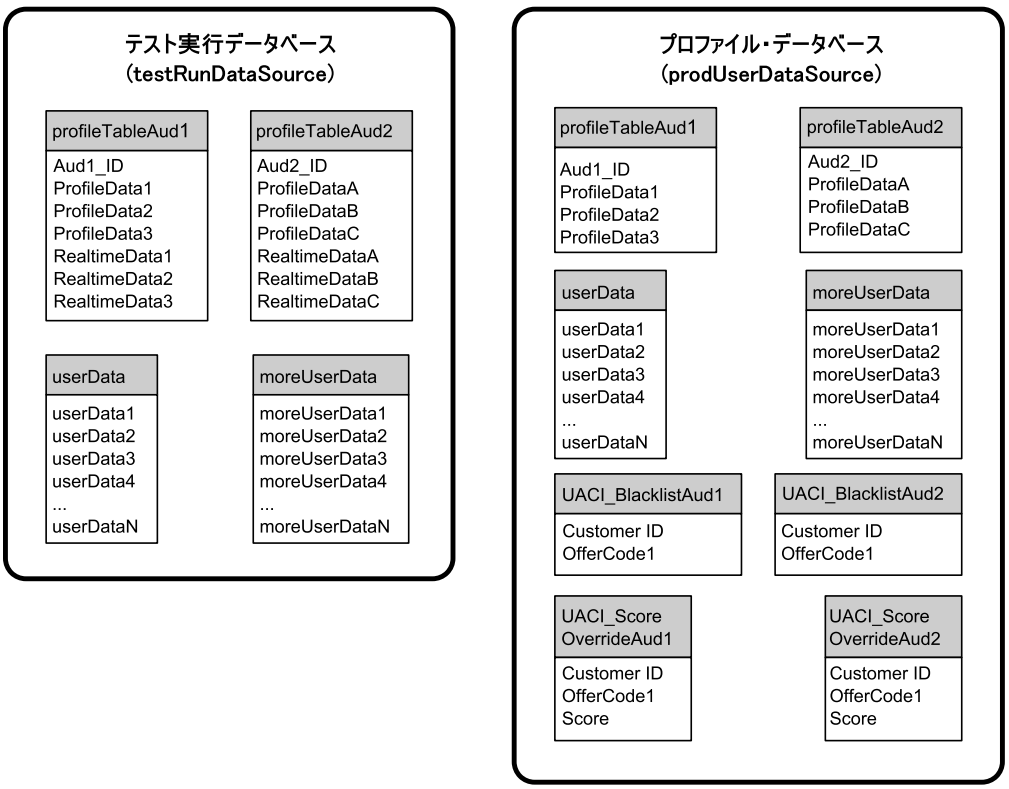
  オファー非表示テーブルにはオーディエンス・メンバーに対して非表示の各オファーの行を含めることができますが、すべてのメンバーに対して入力する必要はありません。サンプルのオファー非表示テーブル UACI_BlackList は、プロファイル・データベースに対して aci_usertab SQL スクリプトを実行することで作成できます。

  enableOfferSuppressionLookup プロパティーを false に設定する場合は、オファー非表示テーブルを含める必要はありません。

これらのテーブルのいずれかに大容量データがある場合は、パフォーマンスが低下する可能性があります。最良の結果を得るために、大容量データがある、実行時に使用されるテーブルのオーディエンス・レベル列に適切な索引を付けてください。

上記の構成プロパティーはすべて「**Interact**」>「**プロファイル**」または「**Interact**」>「**プロファイル**」>「**オーディエンス・レベル**」>「***AudienceLevel***」カテゴリーにあります。aci_usertab SQL スクリプトは、ランタイム環境のインストール・ディレクトリーの ddl ディレクトリーにあります。

以下の図は、Aud1 オーディエンス・レベルと Aud2 オーディエンス・レベルのテスト実行データベースおよびプロファイル・データベースのテーブル例を示したものです。



学習テーブル

Interact の組み込み学習を使用する場合は、学習テーブルを構成する必要があります。これらのテーブルには、組み込み学習機能で使用するすべてのデータが含まれます。

動的学習属性を使用する場合は、UACI_AttributeList テーブルにデータを設定する必要があります。

学習では、中間ステージング・テーブルへの書き込み、およびステージング・テーブルから学習テーブルへの情報の集約を行います。「Interact」>「オファー配信 (offerserving)」>「組み込み学習構成 (Built-in Learning Config)」カテゴリーの insertRawStatsIntervalInMinutes 構成プロパティーと aggregateStatsIntervalInMinutes 構成プロパティーでは、学習テーブルへのデータ設定の頻度を決定します。

insertRawStatsIntervalInMinutes 属性では、顧客とオファーごとの受け入れおよびコンタクト情報が、メモリーからステージング・テーブルの UACI_OfferStatsTX と UACI_OfferAllTx に移動される頻度を決定します。ステージング・テーブルに保管されている情報は集約され、aggregateStatsIntervalInMinutes 構成プロパティーで決定される一定の間隔で UACI_OfferStats テーブルと UACI_OfferStatsAll テーブルに移動されます。

Interact 組み込み学習では、このデータを使用してオファーの最終スコアを計算します。

クロスセッション・レスポンス・トラッキングのコンタクト履歴

クロスセッション・レスポンス機能を有効にする場合は、ランタイム環境からの Campaign コンタクト履歴テーブルへの読み取り専用アクセス権が必要になります。Campaign システム・テーブルを表示するようにランタイム環境を構成することも、Campaign コンタクト履歴テーブルのコピーを作成することもできます。テーブルのコピーを作成する場合は、コピーを最新の状態に保つプロセスを管理する必要があります。コンタクトおよびレスポンス履歴モジュールは、コンタクト履歴テーブルのコピーを更新しません。

クロスセッション・レスポンス・トラッキング機能に必要なテーブルを追加するには、これらのコンタクト履歴テーブルに対して aci_crhtab SQL スクリプトを実行する必要があります。

Interact 機能スクリプトの使用

Interact で使用可能ないくつかのオプション機能では、プロファイル・データベースの特定のテーブルに変更を加える必要があります。設計環境とランタイム環境の両方の Interact インストールに、機能 DDL スクリプトが含まれています。これらのスクリプトによって、テーブルに必要な特定の列が追加されます。

これらの機能を有効にするには、適切なデータベースやテーブルに対して適切なスクリプトを実行します。

dbType は、データベース・タイプです (例えば、Microsoft SQL Server の sqlsvr)。

機能名	機能スクリプト	実行対象	変更
グローバル・オファー、オファー非表示、およびスコア・オーバーライド	ランタイム環境のインストール・ディレクトリー\ddl\aci_usrtab_*dbType*.sql	プロファイル・データベース (userProdDataSource)	DefaultOffers、UACI_BlackList、および UACI_ScoreOverride の各テーブルを作成します。
スコア設定	ランタイム環境のインストール・ディレクトリー\ddl\acifeatures\aci_scoringfeature_*dbType*.sql	プロファイル・データベース (userProdDataSource) のスコア・オーバーライド・テーブル	LikelihoodScore 列および AdjExploreScore 列を追加します。
学習	設計環境のインストール・ディレクトリー\ddl\acifeatures\aci_lrnfeature_*dbType*.sql	コンタクト履歴テーブルを含む Campaign データベース	列 RTSelectionMethod を UA_DtlContactHist テーブルに追加します。

コンタクトとレスポンスの履歴のトラッキングについて

コンタクトとレスポンスの履歴を Campaign のコンタクトとレスポンスの履歴テーブルに記録するように、ランタイム環境を構成することができます。ランタイム・サーバーは、コンタクトとレスポンスの履歴をステージング・テーブルに保管します。コンタクトとレスポンスの履歴モジュールは、このデータを、ステージング・テーブルから Campaign のコンタクトとレスポンスの履歴テーブルにコピーします。

コンタクトとレスポンスの履歴モジュールは、設計環境の「構成」ページで、interactInstalled プロパティーおよび contactAndResponseHistTracking > isEnabled プロパティーを「はい」に設定した場合にのみ機能します。

クロス・セッション・レスポンス・トラッキング・モジュールを使用している場合、コンタクトとレスポンスの履歴モジュールは別のエンティティーになります。

コンタクトとレスポンスのタイプの構成

以下の表が示すように、Interact では 1 つのコンタクト・タイプと 2 つのレスポンス・タイプを記録することができます。これらのプロパティーはすべて、contactAndResponseHistTracking カテゴリーに含まれています。

イベント	コンタクトとレスポンスのタイプ	構成プロパティー
オファー・コンタクトをログに記録	コンタクト	コンタクト済み
オファー承認をログに記録	レスポンス	承認
オファー拒否をログに記録	レスポンス	拒否

また、postEvent メソッドを使用して、カスタムのレスポンス・タイプを追加で記録することも可能です。

Campaign システム・テーブルに含まれている UA_UsrResponseType テーブルの CountsAsResponse 列が正しく構成されていることも、確認する必要があります。これらのレスポンス・タイプはすべて、UA_UsrResponseType テーブル内に存在する必要があります。

UA_UsrResponseType で有効なエントリーにするには、CountsAsResponse を含むそのテーブル内のすべての列の値を定義する必要があります。CountsAsResponse の有効な値は、0、1、または 2 です。0 はレスポンスがないことを示し、1 はレスポンスがあることを示し、2 は拒否を示します。これらのレスポンスは、レポート用に使用されます。

追加のレスポンス・タイプ

Interact では、Interact API の postEvent メソッドを使用して、オファーの「承認」または「拒否」アクションをログに記録するイベントをトリガーできます。また、システムを補完して、postEvent 呼び出しが追加のレスポンス・タイプ (参照、考慮、確定、調達など) を記録できるようにすることができます。これらのレスポンス・タイプはすべて、Campaign システム・テーブルの UA_UsrResponseType テーブル内に存在する必要があります。postEvent メソッドに特定のイベント・パラメーターを使用することで、追加のレスポンス・タイプを記録し、学習に承認を組み込む必要があるかどうかを定義します。

追加のレスポンス・タイプをログに記録するには、以下のイベント・パラメーターを追加する必要があります。

- **UACIRESPONSETYPECODE** — レスポンス・タイプのコードを表すストリング。値は、UA_UsrResponseType テーブルの有効なエントリーでなければなりません。

UA_UsrResponseType で有効なエントリーにするには、CountsAsResponse を含むそのテーブル内のすべての列を定義する必要があります。 CountsAsResponse の有効な値は、0、1、または 2 です。0 はレスポンスがないことを示し、1 はレスポンスがあることを示し、2 は拒否を示します。これらのレスポンスは、レポート用に使用されます。

- **UACILOGTOLEARNING** — 1 または 0 の数値。1 は、Interact が学習用の承認としてイベントをログに記録する必要があることを示します。0 は、Interact が学習用にイベントをログに記録する必要がないことを示します。このパラメーターを使用することで、学習に影響を及ぼさずに異なるレスポンス・タイプをログに記録する、複数の postEvent メソッドを作成することができます。UACILOGTOLEARNING を定義しない場合、Interact はデフォルト値の 0 であるとみなします。

「オファー承認をログに記録」アクションで複数のイベント (ログに記録するレスポンス・タイプごとに 1 つ) を作成するか、あるいは「オファー承認をログに記録」アクションを使用して単一のイベントを作成し、異なる複数のレスポンス・タイプをログに記録するために使用するすべての postEvent 呼び出しに使用することができます。

例えば、レスポンスのタイプごとに、「オファー承認をログに記録」アクションでイベントを作成します。UA_UsrResponseType テーブルの「名前 (コード) (as Name (code))」で、「参照 (EXP)」、「考慮 (CON)」、および「確定 (CMT)」というカスタム・レスポンスを定義します。その後、3 つのイベントを作成し、それらに LogAccept_Explore、LogAccept_Consider、および LogAccept_Commit という名前を付けます。 3 つのイベントはすべて (「オファー承認をログに記録」アクションを持つ) 完全に同じものですが、名前が異なるため、その API を使用して作業を行うユーザーは、それらを区別することができます。

また、「オファー承認をログに記録」アクションで単一のイベントを作成して、すべてのカスタム・レスポンス・タイプに使用することもできます。これには、例えば LogCustomResponse という名前を付けます。

この API を使用して作業を行う場合、それらのイベントに機能的な違いはありませんが、この命名規則によってコードがより明確になる場合があります。また、それぞれのカスタム・レスポンスに別個の名前を付けると、「チャネル・イベント・アクティビティー・サマリー」レポートに表示される情報が、より正確になります。

まず、すべての名前と値のペアをセットアップします。

```
//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);
```

この 1 つ目の例は、個々のイベントを使用する場合を示しています。

```
//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);
```

この 2 つ目の例は、単一のイベントのみを使用する場合を示しています。

```
//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);
```

どちらの例も、完全に同じアクションを実行していますが、片方がもう 1 つの方よりも読みやすくなる場合があります。

ランタイム環境のステージング・テーブルから Campaign の履歴テーブルへのマッピング

以下の表には、ランタイム環境のステージング・テーブルを Campaign の履歴テーブルにマップする方法が示されています。オーディエンス・レベルごとにこれらのテーブルのいずれか 1 つが必要であることを覚えておいてください。示されているテーブル名は、ランタイム・テーブルと Campaign システム・テーブルにおけるデフォルト・オーディエンス用に作成されたサンプル・テーブルです。

表 2. コンタクト履歴

UACI_CHStaging **Interact コンタクト履歴ステージング・テーブルの列名**	**Campaign コンタクト履歴テーブル**	**テーブルの列名**
ContactID	該当なし	該当なし
TreatmentCode	UA_Treatment	TreatmentCode
CampaignID	UA_Treatment	CampaignID
OfferID	UA_Treatment	OfferID
CellID	UA_Treatment	CellID
CustomerID	UA_DtlContactHist	CustomerID
ContactDate	UA_DtlContactHist	ContactDateTime
ExpirationDateTime	UA_Treatment	ExpirationDateTime
EffectiveDateTime	UA_Treatment	EffectiveDateTime
ContactType	UA_DtlContactHist	ContactStatusID
UserDefinedFields	UA_DtlContactHist	UserDefinedFields

ContactID は、UACI_CHOfferAtrib を UACI_CHStaging と結合させるキーです。

表 3. オファー属性

UACI_CHOfferAttrib **Interact コンタクト履歴ステージング・テーブルの列名**	**Campaign コンタクト履歴テーブル**	**テーブルの列名**
ContactID	該当なし	該当なし
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

ContactID は、UACI_CHOfferAtrib を UACI_CHStaging と結合させるキーです。

表 4. レスポンス履歴

UACI_RHStaging **Interact レスポンス履歴ステージング・テーブルの列名**	**Campaign レスポンス履歴テーブル**	**テーブルの列名**
SeqNum	該当なし	該当なし
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime
ResponseType	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

SeqNum は、データを識別するためにコンタクトおよびレスポンス履歴モジュールが使用するキーですが、Campaign レスポンス・テーブルには記録されません。

userDefinedFields 列には選択したすべてのデータを含めることができます。ステージング・テーブルに列を追加すると、コンタクトおよびレスポンス履歴モジュールは、その列を UA_DtlContactHist テーブルまたは UA_ResponseHistory テーブルに同じ名前で書き込みます。例えば、linkFrom という列を UACI_CHStaging テーブルに追加した場合、コンタクトおよびレスポンス履歴モジュールはそのデータを UA_DtlContactHist テーブルの linkFrom 列にコピーします。

重要: Campaign コンタクトおよびレスポンス履歴テーブルに追加列がある場合は、コンタクトおよびレスポンス履歴モジュールを実行する前に、ステージング・テーブルに一致する列を追加する必要があります。

ステージング・テーブルに列を追加する場合は、ランタイム・セッション・データ内の名前と値のペアと同じ名前の列を作成します。例えば、NumberItemsInWishList および NumberItemsInShoppingCart という名前と値のペアを作成したとします。「オファー承認をログに記録」イベントまたは「オファー拒否をログに記録」イベントの発生時に、UACI_RHStaging テーブルに NumberItemsInWishList 列と NumberItemsInShoppingCart 列が存在する場合、ランタイム環境ではこれらの項目にデータが追加されます。ランタイム環境では、「オファー・コンタクトをログに記録」イベントの発生時に UACI_CHStaging テーブルにデータが追加されます。

これらのユーザー定義項目を使用して、オファーの提示に使用されるスコアを含めることができます。その場合、FinalScore という名前の列を、ランタイム・テーブル内の UACI_CHStaging テーブルと Campaign システム・テーブル内の UA_DtlContactHist テーブルの両方に追加します。組み込み学習を使用する場合は、オファーに使用される最終スコアが Interact によって、自動的に FinalScore 列に追加されます。

カスタマイズされた学習モジュールを作成する場合は、ITreatment インターフェースの setActualValueUsed メソッドと ILearning インターフェースの logEvent メソッドを使用できます。

学習を使用しない場合は、Score という名前の列を、ランタイム・テーブルの UACI_CHStaging テーブルと Campaign システム・テーブルの UA_DtlContactHist テーブルの両方に追加します。オファーに使用されるスコアは、Interact によって自動的に Score 列に追加されます。

コンタクトとレスポンスの履歴モジュール用に JMX 監視を構成するには

設計環境の Marketing Platform では、「Campaign」>「監視」カテゴリーで以下の構成プロパティーを編集します。

構成プロパティー	設定
monitorEnabledForInteract	**True**
ポート	JMX サービスのポート番号

構成プロパティー	設定
プロトコル (protocol)	**JMXMP** または **RMI** JMXMP プロトコルを選択した場合でも、コンタクトとレスポンスの履歴モジュールのセキュリティーは有効になっていません。

コンタクトとレスポンスの履歴データを JMX 監視ツールで表示すると、属性は最初はパーティションで編成され、次にオーディエンス・レベルで編成されます。

JMXMP プロトコルを使用するコンタクトとレスポンスの履歴モジュールの監視用のデフォルト・アドレスは、`service:jmx:jmxmp://CampaignServer:port/campaign` です。

RMI プロトコルを使用するコンタクトとレスポンスの履歴モジュールの監視用のデフォルト・アドレスは、`service:jmx:rmi:///jndi/rmi://CampaignServer:port/campaign` です。

クロスセッション・レスポンス・トラッキングについて

訪問者がタッチポイントへの 1 回のアクセスでトランザクションを完了させるとは限りません。顧客が Web サイト上でショッピング・カートにアイテムを追加しても、購入するのは 2 日後になる場合があります。ランタイム・セッションを無期限でアクティブにしておくことはできません。クロスセッション・レスポンス・トラッキングを有効にすれば、1 つのセッションのオファー提示をトラッキングし、別のセッションのレスポンスと照合することができます。

Interact クロスセッション・レスポンス・トラッキングでは、デフォルトで、処理コードまたはオファー・コードで照合できます。また、選択したカスタム・コードを照合するように構成することもできます。クロスセッション・レスポンスでは使用可能なデータで照合します。例えば、Web サイトに、1 週間の割引商品の表示時に生成された販促コードの付いたオファーが含まれているとします。ユーザーがショッピング・カートにアイテムを追加しても、購入は 3 日後になる可能性があります。承認イベントをログに記録するために `postEvent` 呼び出しを使用する場合は、販促コードのみを含めることができます。ランタイムは現行セッションで照合する処理コードやオファー・コードを見つけられないため、使用可能な情報を含む承認イベントをクロスセッション・レスポンス (XSessResponse) ステージング・テーブルに入れます。CrossSessionResponse サービスは定期的に XSessResponse テーブルを読み取り、レコードと使用可能なコンタクト履歴データとの照合を試みます。この CrossSessionResponse サービスは販促コードとコンタクト履歴を照合し、適切なレスポンスをログに記録するために必要なデータをすべて収集します。次に、CrossSessionResponse サービスはレスポンスをレスポンス・ステージング・テーブルに書き込み、学習が有効な場合は、学習テーブルにも書き込みます。その後、コンタクトおよびレスポンス履歴モジュールはレスポンスを Campaign コンタクトおよびレスポンス履歴テーブルに書き込みます。

クロスセッション・レスポンス・トラッキングのデータ・ソース構成

Interact クロスセッション・レスポンス・トラッキングでは、ランタイム環境のセッション・データを Campaign コンタクトおよびレスポンス履歴と照合します。デフォルトでは、クロスセッション・レスポンス・トラッキングで処理コードまたはオファー・コードでの照合が行われます。カスタム代替コードで照合するようにランタイム環境を構成することができます。

- 代替コードで照合する場合は、Interact ランタイム・テーブルの UACI_TrackingType テーブルにその代替コードを定義する必要があります。
- ランタイム環境から Campaign コンタクト履歴テーブルにアクセスできる必要があります。Campaign コンタクト履歴テーブルにアクセスできるようにするには、ランタイム環境をそのように構成するか、ランタイム環境でコンタクト履歴テーブルのコピーを作成します。

  このアクセス権は読み取り専用であり、コンタクトおよびレスポンス履歴ユーティリティーから独立しています。

  テーブルのコピーを作成する場合、コンタクト履歴コピーのデータが正確であることを保証するのはユーザーの責任です。
  purgeOrphanResponseThresholdInMinutes プロパティーを使用して、一致しないレスポンスが CrossSessionResponse サービスによって消去されるまでに保持する時間の長さを設定します。この時間は、コンタクト履歴テーブル・コピー内のデータの更新頻度と一致させることができます。コンタクトおよびレスポンス履歴モジュールを使用する場合は、データが最新の状態になるように ETL 更新を調整する必要があります。

クロスセッション・レスポンス・トラッキング用のコンタクトおよびレスポンス履歴テーブルの構成

コンタクト履歴テーブルのコピーを作成するか、Campaign システム・テーブル内の実際のテーブルを使用するかに関係なく、以下のステップを実行する必要があります。

1. コンタクトおよびレスポンス履歴テーブルを Campaign で適切にマップする必要があります。
2. Campaign システム・テーブル内の UA_DtlContactHist テーブルと UA_ResponseHistory テーブルに対して、Interact 設計環境インストール・ディレクトリーの interactDT/ddl/acifeatures ディレクトリーにある aci_lrnfeature SQL スクリプトを実行する必要があります。

   これにより、RTSelectionMethod 列が UA_DtlContactHist テーブルと UA_ResponseHistory テーブルに追加されます。オーディエンス・レベルごとにこれらのテーブルに対して aci_lrnfeature スクリプトを実行します。必要に応じて、各オーディエンス・レベルの適切なテーブルで使用するようにスクリプトを編集します。
3. コンタクト履歴テーブルをランタイム環境にコピーする場合は、この時点で行います。

4. コンタクトおよびレスポンス履歴データ・ソースに対して、Interact ランタイム環境インストール・ディレクトリーの ddl ディレクトリーにある aci_crhtab SQL スクリプトを実行します。

   このスクリプトにより、UACI_XsessResponse テーブルと UACI_CRHTAB_Ver テーブルが作成されます。
5. 各オーディエンス・レベルの UACI_XsessResponse テーブル・バージョンを作成します。

クロスセッション・レスポンス・トラッキングをサポートするためにランタイム環境からアクセス可能な Campaign コンタクト履歴テーブルのコピーを作成する場合は、以下のガイドラインを使用してください。

- クロスセッション・レスポンス・トラッキングには以下のテーブルへの読み取り専用アクセス権が必要です。
- クロスセッション・レスポンス・トラッキングには Campaign コンタクト履歴の以下のテーブルが必要です。
  - UA_DtlContactHist (オーディエンス・レベルごと)
  - UA_Treatment

  正確なレスポンス・トラッキングを行うには、これらのテーブル内のデータを定期的に更新する必要があります。

クロスセッション・レスポンス・トラッキングのパフォーマンスを向上させるために、コンタクト履歴データをコピーする方法によって、または Campaign コンタクト履歴テーブルの表示を構成することによって、コンタクト履歴データの量を制限できます。例えば、オファーの有効期間を 30 日以下とするビジネスに適用する場合は、表示するコンタクト履歴データを過去 30 日間に限定する必要があります。

コンタクトおよびレスポンス履歴モジュールが実行されるまでは、クロスセッション・レスポンス・トラッキングの結果は表示されません。例えば、デフォルトの processSleepIntervalInMinutes は 60 分です。そのため、Campaign レスポンス履歴にクロスセッション・レスポンスが表示されるまで、少なくとも 1 時間はかかる可能性があります。

UACI_TrackingType テーブル

UACI_TrackingType テーブルはランタイム環境テーブルの一部です。このテーブルでは、クロス・セッション・レスポンス・トラッキングで使用されるトラッキング・コードを定義します。トラッキング・コードは、ランタイム・セッションの現行オファーとコンタクトおよびレスポンス履歴を照合するためにランタイム環境で使用されるメソッドを定義します。

列	タイプ	説明
TrackingCodeType	int	トラッキング・コード・タイプを表す数値。この数値は、セッション・データの情報をコンタクトおよびレスポンス履歴テーブルと照合するために使用される SQL コマンドによって参照されます。

列	タイプ	説明
名前	varchar(64)	トラッキング・コード・タイプの名前。これは、postEvent メソッドで UACI_TrackingCodeType 予約済みパラメーターを使用して、セッション・データに渡されます。
説明	varchar(512)	トラッキング・コード・タイプの簡単な説明。この項目はオプションです。

デフォルトでは、以下の表に示されているように、ランタイム環境には 2 つのトラッキング・コード・タイプが定義されます。代替コードの場合は、固有の TrackingCodeType を定義する必要があります。

TrackingCodeType	名前	説明
1	処理コード	UACI 生成処理コード
2	オファー・コード	UAC キャンペーン・オファー・コード

UACI_XSessResponse

Interact クロスセッション・レスポンス・トラッキングで使用可能なコンタクトおよびレスポンス履歴データ・ソースには、オーディエンス・レベルごとに、以下の表に示されているいずれか 1 つのインスタンスが存在する必要があります。

列	タイプ	説明
SeqNumber	bigint	データ行の ID。CrossSessionResponse サービスは、すべてのレコードを SeqNumber 順に処理します。
ICID	bigint	インタラクティブ・チャネル ID
AudienceID	bigint	このオーディエンス・レベルのオーディエンス ID。この列の名前は、Campaign に定義されているオーディエンス ID と一致する必要があります。サンプル・テーブルには CustomerID という列が含まれています。
TrackingCode	varchar(64)	postEvent メソッドの UACIOfferTrackingCode パラメーターによって渡される値。
TrackingCodeType	int	トラッキング・コードの数値表現。値は、UACI_TrackingType テーブルで有効なエントリーでなければなりません。
OfferID	bigint	Campaign に定義されたオファー ID。
ResponseType	int	このレコードのレスポンス・タイプ。値は、UA_UsrResponseType テーブルの有効なエントリーでなければなりません。
ResponseTypeCode	varchar(64)	このレコードのレスポンス・タイプ・コード。値は、UA_UsrResponseType テーブルの有効なエントリーでなければなりません。
ResponseDate	datetime	レスポンスの日付。

列	タイプ	説明
Mark	bigint	この項目の値はレコードの状態を識別します。 • 1 — 処理中 • 2 — 成功 • NULL — 再試行 • -1 — レコードは purgeOrphanResponseThresholdInMinutes 分より長い間データベース内にあります。 このテーブルのデータベース管理者による保守作業の一環として、この項目で一致しないレコード (つまり、値が -1 のすべてのレコード) を調べることができます。値が 2 のすべてのレコードは、CrossSessionResponse サービスによって自動的に削除されます。
UsrDefinedFields	char(18)	オファー・レスポンスをコンタクトおよびレスポンス履歴と照合する際に含めるカスタム項目。例えば、販促コードで照合する場合は、販促コード用のユーザー定義項目を含めます。

クロスセッション・レスポンス・トラッキングを有効にするには

クロスセッション・レスポンス・トラッキングを最大限に利用するには、コンタクトおよびレスポンス履歴モジュールを構成する必要があります。

クロスセッション・レスポンス・トラッキングを使用するには、Campaign コンタクトおよびレスポンス履歴テーブルを読み取れるようにランタイム環境を構成する必要があります。設計環境内の実際の Campaign コンタクトおよびレスポンス履歴テーブル、またはランタイム環境のデータ・ソース内のテーブル・コピーから読み取り可能です。これは、コンタクトおよびレスポンス履歴モジュールの構成とは独立しています。

処理コードやオファー・コード以外のもので照合する場合は、UACI_TrackingType テーブルにそれを追加する必要があります。

1. ランタイム環境からアクセス可能なコンタクトおよびレスポンス履歴テーブルに XSessResponse テーブルを作成します。
2. ランタイム環境の contactAndResponseHistoryDataSource カテゴリーにプロパティーを定義します。
3. オーディエンス・レベルごとに crossSessionResponseTable プロパティーを定義します。
4. オーディエンス・レベルごとに OverridePerAudience カテゴリーを作成します。

クロスセッション・レスポンス・オファーの照合

デフォルトでは、クロスセッション・レスポンス・トラッキングで処理コードまたはオファー・コードでの照合が行われます。crossSessionResponse サービスは SQL コマンドを使用して、セッション・データの処理コード、オファー・コード、また

はカスタム・コードを Campaign のコンタクトおよびレスポンス履歴テーブルと照合します。これらの SQL コマンドは、カスタマイズしたトラッキング・コード、オファー・コード、またはカスタム・コードと照合するように編集することができます。

処理コードによる照合

処理コードで照合を行う SQL は、このオーディエンス・レベルの XSessResponse テーブルのすべての列と、OfferIDMatch と呼ばれる列を返す必要があります。OfferIDMatch 列の値は、XSessResponse レコードの処理コードに対応した offerId でなければなりません。

以下に、処理コードを照合する、デフォルトで生成される SQL コマンドの例を示します。Interact は、オーディエンス・レベルの適切なテーブル名を使用する SQL を生成します。この SQL は、「Interact」>「サービス」>「crossSessionResponse」>「OverridePerAudience」>*「AudienceLevel*>「TrackingCodes」>「byTreatmentCode」>「SQL」プロパティーが「**システム生成 SQL を使用 (Use System Generated SQL)**」に設定されている場合に使用されます。

```
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
```

値の UACI_XsessResponse、UA_DtlContactHist、CustomerID、および UA_ContactHistory は、Interact の設定によって定義されます。例えば、UACI_XsessResponse は「Interact」>「プロファイル」>「オーディエンス・レベル」> [AudienceLevelName] >「crossSessionResponseTable」構成プロパティーで定義されます。

コンタクトおよびレスポンス履歴テーブルをカスタマイズした場合は、そのテーブルで使用できるようにこの SQL を変更する必要がある可能性があります。 SQL のオーバーライドを「Interact」>「サービス」>「crossSessionResponse」>「OverridePerAudience」>*「(AudienceLevel)」*>「TrackingCodes」>「byTreatmentCode」>「OverrideSQL」プロパティーで定義してください。オーバーライド SQL をいくつか指定した場合は、「SQL」プロパティーを「**オーバーライド SQL (Override SQL)**」に変更する必要もあります。

オファー・コードによる照合

オファー・コードで照合を行う SQL は、このオーディエンス・レベルの XSessResponse テーブルのすべての列と、TreatmentCodeMatch と呼ばれる列を返す必要があります。 TreatmentCodeMatch 列の値は、XSessResponse レコードのオファー ID (およびオファー・コード) を伴う処理コードです。

以下に、オファー・コードを照合する、デフォルトで生成される SQL コマンドの例を示します。Interact は、オーディエンス・レベルの適切なテーブル名を使用する SQL を生成します。この SQL は、「Interact」>「サービス」>「crossSessionResponse」>「OverridePerAudience」>「*AudienceLevel*」>「TrackingCodes」>「byOfferCode」>「SQL」プロパティーが「**システム生成 SQL を使用 (Use System Generated SQL)**」に設定されている場合に使用されます。

```
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
```

値の UACI_XsessResponse、UA_DtlContactHist、CustomerID、および UA_ContactHistory は、Interact の設定によって定義されます。例えば、UACI_XsessResponse は 「Interact」>「プロファイル」>「オーディエンス・レベル」> [AudienceLevelName] >「crossSessionResponseTable」構成プロパティーで定義されます。

コンタクトおよびレスポンス履歴テーブルをカスタマイズした場合は、そのテーブルで使用できるようにこの SQL を変更する必要がある可能性があります。 SQL のオーバーライドを「Interact」>「サービス」
>「crossSessionResponse」>「OverridePerAudience」>*「(AudienceLevel)」*
>「TrackingCodes」>「byOfferCode」>「OverrideSQL」プロパティーで定義してください。オーバーライド SQL をいくつか指定した場合は、「SQL」プロパティーを「**オーバーライド SQL (Override SQL)**」に変更する必要もあります。

代替コードによる照合

選択したいくつかの代替コードで照合するように SQL コマンドを定義できます。例えば、オファー・コードまたは処理コードとは別の販促コードまたは製品コードを使用できます。

この代替コードは、Interact ランタイム環境テーブルの UACI_TrackingType テーブルに定義する必要があります。

SQL またはストアード・プロシージャーは、「Interact」>「サービス」
>「crossSessionResponse」>「OverridePerAudience」> *「(AudienceLevel)」*
>「TrackingCodes」>「byAlternateCode」>「OverrideSQL」プロパティーで指定する必要があります。これにより、このオーディエンス・レベルの XSessResponse テーブル内のすべての列、および TreatmentCodeMatch 列と OfferIDMatch 列が返されます。必要に応じて、OfferIDMatch の代わりに offerCode (offerCode1、offerCode2、 ... offerCode*N* などの形式。ここで、N 部分はオファー・コードを表します) を返すこともできます。TreatmentCodeMatch column 列と OfferIDMatch 列 (またはオファー・コード列) の値は、XSessResponse レコードの TrackingCode に対応している必要があります。

例えば、以下の SQL 疑似コードは、XSessResponse テーブルの AlternateCode 列と一致しています。

```
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>
```

ここで、<*x*> は UACI_TrackingType テーブルに定義されているトラッキング・コードです。

ランタイム環境でのデータベース・ロード・ユーティリティーの使用

デフォルトでは、ランタイム環境で、セッション・データのコンタクトおよびレスポンス履歴データがステージング・テーブルに書き込まれます。ただし、非常にアクティブな運用システムでは、ランタイムがすべてのデータをステージング・テーブルに書き込む前にそのデータをキャッシュに入れるために必要なメモリーが非常に大きいために用意できない場合があります。パフォーマンスを向上させるためにデータベース・ロード・ユーティリティーを使用するようにランタイムを構成することができます。

すべてのコンタクトおよびレスポンス履歴をステージング・テーブルに書き込む前にメモリーで保持する代わりに、データベース・ロード・ユーティリティーを有効にすると、ランタイムはデータをステージング・ファイルに書き込みます。ステージング・ファイルを含むディレクトリーの場所は、`externalLoaderStagingDirectory` プロパティーを使用して定義します。このディレクトリーにはいくつかのサブディレクトリーが含まれます。最初のサブディレクトリーはランタイム・インスタンス・ディレクトリーで、ここには `contactHist` ディレクトリーと `respHist` ディレクトリーがあります。 `contactHist` ディレクトリーと `respHist` ディレクトリーには、ステージング・ファイルを含む、*audienceLevelName.uniqueID.currentState* という形式の一意に名前付けされたサブディレクトリーがあります。

現在の状態	説明
CACHE	ディレクトリー・コンテンツは現在ファイルに書き込まれています。
READY	ディレクトリー・コンテンツの処理準備は完了しています。
RUN	ディレクトリー・コンテンツは現在データベースに書き込まれています。
PROCESSED	ディレクトリー・コンテンツはデータベースに書き込まれました。
ERROR	データベースへのディレクトリー・コンテンツの書き込み中にエラーが発生しました。
ATTN	ディレクトリー・コンテンツに注意する必要があります。つまり、このディレクトリー・コンテンツのデータベースへの書き込みを完了するために、いくつかのステップを手動で実行する必要がある可能性があります。
RERUN	ディレクトリー・コンテンツはデータベースへの書き込み準備が完了しています。問題を修正した後で、ディレクトリーの名前を `ATTN` または `ERROR` から `RERUN` に変更する必要があります。

ランタイム・インスタンス・ディレクトリーは、アプリケーション・サーバーの起動スクリプトに interact.runtime.instance.name JVM プロパティーを定義することで定義できます。例えば、`-Dinteract.runtime.instance.name=instance2` を Web アプリケーション・サーバーの起動スクリプトに追加できます。設定されていない場合、デフォルト名は `DefaultInteractRuntimeInstance` になります。

`samples` ディレクトリーには、独自のデータベース・ロード・ユーティリティー制御ファイルの書き込みに役立つサンプル・ファイルが含まれています。

ランタイム環境でデータベース・ロード・ユーティリティーを使用可能にするには

データベース・ロード・ユーティリティーの任意のコマンドまたはコントロール・ファイルをランタイム環境で使用できるように構成するには、あらかじめそれらを定義しておく必要があります。それらのファイルは、同じサーバー・グループ内のすべてのランタイム・サーバーで同じロケーションに存在している必要があります。

Interact では、Interact ランタイム・サーバーがインストールされた環境の loaderService ディレクトリーに、サンプルのコマンドとコントロール・ファイルが含まれています。

1. ランタイム環境ユーザーが、Marketing Platform で定義されているランタイム・テーブル・データ・ソースのログイン資格情報を持っていることを確認します。

   Marketing Platform のデータ・ソースの名前は、systemTablesDataSource になっている必要があります。
2. 「Interact」>「全般」>「systemTablesDataSource」>「loaderProperties」構成プロパティーを定義します。
3. 「Interact」>「サービス (services)」>「externalLoaderStagingDirectory」プロパティーを定義します。
4. 「Interact」>「サービス (services)」>「responseHist」>「fileCache」構成プロパティーを必要に応じて変更します。
5. 「Interact」>「サービス (services)」>「contactHist」>「fileCache」構成プロパティーを必要に応じて変更します。
6. ランタイム・サーバーを再始動します。

第 4 章 オファーの提供

提示するオファーの選択方法を向上させるために、さまざまな方法で Interact を構成できます。以下のセクションでは、それらのオプション機能について詳しく説明します。

オファーの適格性

Interact の目的は適格なオファーを提示することです。簡単に言うと、Interact では、訪問者、チャネル、およびシチュエーションに基づいて、適格なオファーの中から最適のものが提示されます。

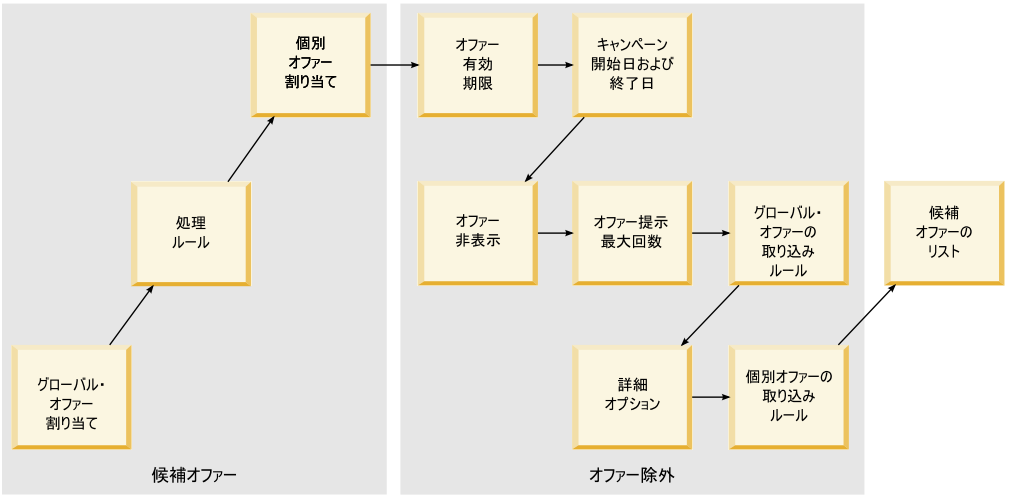
処理ルールは、顧客に対して適格なオファーを Interact で判別するための開始点にすぎません。Interact には、ランタイム環境で提示するオファーの判別方法を向上させるために実装できるいくつかのオプション機能があります。これらの機能によって、オファーが顧客に提示されることが保証されるわけではありません。これらの機能は、オファーが顧客に提示されるものとして的確である可能性に影響を与えるものです。環境に対して最良のソリューションを実装するために、これらの機能を必要に応じて使用できます。

オファーの適格性に影響を与える主な領域が 3 つあり、それらは、候補オファーのリストの生成、マーケティング・スコアの判別、および学習です。

候補オファーのリストの生成

候補オファーのリストの生成には 2 つの主なステージがあります。最初のステージでは、顧客に対して適格である可能性のあるすべてのオファーのリストが生成されます。2 番目のステージでは、顧客に対して適格ではないオファーがすべてフィルターで除去されます。両方のステージには、候補オファー・リストの生成に影響を与えることができる個所がいくつかあります。

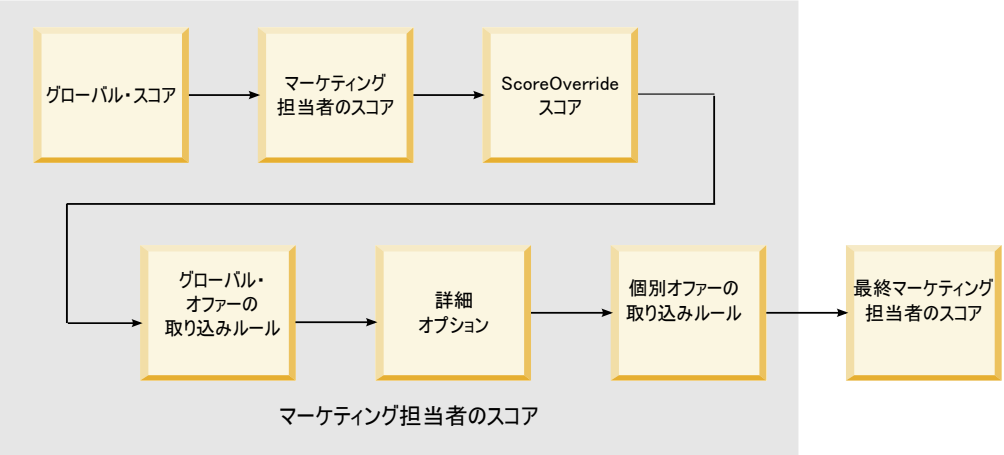
以下の図は、候補オファー・リスト生成のステージを示しています。矢印は優先順位の順序を示します。例えば、オファーが**オファー提示の最大回数**フィルターを通過しても、**グローバル・オファーの取り込みルール**・フィルターを通過しない場合、ランタイム環境ではそのオファーは除外されます。



- **グローバル・オファーの割り当て** — グローバル・オファー・テーブルを使用して、オーディエンス・レベルによってグローバル・オファーを定義できます。
- **処理ルール** — インタラクション方法タブを使用して、インタラクション・ポイントのセグメントによってオファーを定義する基本的な方法です。
- **個別オファーの割り当て** — スコア・オーバーライド・テーブルを使用して、顧客によって固有オファーの割り当てを定義できます。
- **オファーの有効期限** — Campaign でオファーを作成するときに、有効期限を定義できます。オファーの有効期限が切れている場合、ランタイム環境ではそのオファーは除外されます。
- **キャンペーンの開始日および終了日** — Campaign でキャンペーンを作成するときに、キャンペーンの開始日および終了日を定義できます。キャンペーンの開始日が来ていないか、キャンペーンの終了日が過ぎている場合、ランタイム環境ではそのオファーは除外されます。
- **オファー非表示** — オファー非表示テーブルを使用して、特定のオーディエンス・メンバーに対するオファー非表示を定義できます。
- **オファー提示の最大回数** — インタラクティブ・チャネルを定義するときに、顧客にオファーを提示するセッションごとの最大回数を定義します。オファーの提示が既にこの回数に達している場合、ランタイム環境ではそのオファーは除外されます。
- **グローバル・オファーの取り込みルール** — グローバル・オファー・テーブルを使用して、オーディエンス・レベルでオファーをフィルターに掛けるためのブール式を定義できます。結果が FALSE である場合、ランタイム環境ではそのオファーは除外されます。
- **詳細オプション** — 処理ルールの「**次の式が TRUE の場合は、このルールを対象と見なす**」詳細オプションを使用して、セグメント・レベルでオファーをフィルターに掛けることができます。結果が FALSE である場合、ランタイム環境ではそのオファーは除外されます。
- **個別オファーの取り込みルール** — スコア・オーバーライド・テーブルを使用して、顧客レベルでオファーをフィルターに掛けるためのブール式を定義できます。結果が FALSE である場合、ランタイム環境ではそのオファーは除外されます。

マーケティング・スコアの計算

マーケティング・スコアに (計算を使用することにより) 影響を与えたり、マーケティング・スコアをオーバーライドしたりする多くの方法があります。以下の図は、マーケティング・スコアに影響を与えたり、マーケティング・スコアをオーバーライドしたりできる、さまざまなステージを示しています。矢印は優先順位の順序を示します。例えば、処理ルールの詳細オプションでマーケティング・スコアを決定するための式を定義し、スコア・オーバーライド・テーブルの式を定義する場合、スコア・オーバーライド・テーブルの式の方が優先順位は上です。



- **グローバル・スコア** — グローバル・オファー・テーブルを使用して、オーディエンス・レベルごとにスコアを定義できます。
- **マーケティング担当者のスコア** — 処理ルールのスライダーを使用して、セグメントごとにスコアを定義できます。
- **スコア・オーバーライド・スコア** — スコア・オーバーライド・テーブルを使用して、顧客ごとにスコアを定義できます。
- **グローバル・オファーの取り込みルール** — グローバル・オファー・テーブルを使用して、オーディエンス・レベルごとにスコアを計算する式を定義できます。
- **詳細オプション** — 処理ルールの「**次の式をマーケティング・スコアとして使用する**」詳細オプションを使用して、セグメントごとにスコアを計算する式を定義できます。
- **スコア・オーバーライド・オファーの取り込みルール** — スコア・オーバーライド・テーブルを使用して、顧客ごとにスコアを計算する式を定義できます。

学習の影響

Interact 組み込み学習モジュールを使用している場合、学習属性のリストや信頼性レベルなどの標準学習構成に優先して、学習出力に影響を与えることができます。他のコンポーネントの使用中に、学習アルゴリズムのコンポーネントをオーバーライドできます。

デフォルト・オファー・テーブルおよびスコア・オーバーライド・テーブルの `LikelihoodScore` 列および `AdjExploreScore` 列を使用して、学習をオーバーライドできます。これらの列は、`aci_scoringfeature` 機能スクリプトを使用して、デフォ

ルト・オファー・テーブルおよびスコア・オーバーライド・テーブルに追加できます。これらのオーバーライドを適切に使用するには、Interact 組み込み学習を十分に理解していることが必要です。

学習モジュールでは、候補オファーのリストおよび候補オファーごとのマーケティング・スコアが取得され、それらが最終計算に使用されます。オファー・リストは、顧客がオファーを受け入れるという見込み (受け入れの可能性) を計算するために学習属性と共に使用されます。学習アルゴリズムによって、これらの可能性と表示の履歴数が使用されて探索と活用の間のバランスが取られ、オファーの重みが決定されます。最後に、組み込み学習でオファーの重みが取得されて、それが最終マーケティング・スコアと乗算され、最終スコアが返されます。オファーはこの最終スコア順にソートされます。

オファーの非表示について

ランタイム環境でオファーを非表示にする方法が以下のようにいくつかあります。

- インタラクティブ・チャネルの **1 回の訪問時のオファーの最大表示回数**要素。

  インタラクティブ・チャネルを作成または編集するときに、**1 回の訪問時のオファーの最大表示回数**を定義します。
- オファー非表示テーブルの使用。

  プロファイル・データベースにオファー非表示テーブルを作成します。
- 有効期限が切れているオファー。
- 有効期限が切れているキャンペーンのオファー。
- オファーの取り込みルールをパスしないために除外されたオファー (処理ルールの詳細オプション)。
- Interact セッションで既に明示的に受け入れられたか拒否されたオファー。顧客が明示的にオファーを受け入れたか拒否した場合、そのオファーはセッション期間中は非表示になります。

オファー非表示テーブルを有効にするには

非表示にされるオファーのリストを参照するように Interact を構成できます。

1. オーディエンス ID とオファー ID を含む、すべてのオーディエンス用の新規テーブル、`offerSuppressionTable` を作成します。
2. `enableOfferSuppressionLookup` プロパティーを **true** に設定します。
3. `offerSuppressionTable` プロパティーを該当するオーディエンスのオファー非表示テーブルの名前に設定します。

オファー非表示テーブル

オファー非表示テーブルを使用すると、特定のオーディエンス ID に対してオファーを非表示にできます。例えば、オーディエンスが顧客である場合、顧客 John Smith に対してオファーを非表示にできます。このテーブルの少なくとも 1 つのオーディエンス・レベルに対するバージョンが運用プロファイル・データベースに存在する必要があります。サンプルのオファー非表示テーブル `UACI_Blacklist` は、

プロファイル・データベースに対して aci_usertab SQL スクリプトを実行することで作成できます。aci_usertab SQL スクリプトは、ランタイム環境のインストール・ディレクトリー内の ddl ディレクトリーにあります。

各行に AudienceID フィールドおよび OfferCode1 フィールドを定義する必要があります。オーディエンス ID またはオファー・コードが複数の列から構成されている場合は、列を追加できます。これらの列は Campaign で定義された列名と一致する必要があります。例えば、オーディエンス Customer がフィールド HHold_ID と MemberNum によって定義されている場合、オファー非表示テーブルに HHold_ID と MemberNum を追加する必要があります。

名前	説明
AudienceID	(必須) この列の名前は、Campaign のオーディエンス ID を定義している列の名前と一致する必要があります。オーディエンス ID が複数の列から構成されている場合、それらの列をこのテーブルに追加できます。各行には、デフォルトのオファー、例えば customer1 が割り当てられたオーディエンス ID が含まれている必要があります。
OfferCode1	(必須) オーバーライドするオファーに対するオファー・コード。オファー・コードが複数のフィールドから構成されている場合、追加の列、例えば OfferCode2 などを追加できます。

グローバル・オファーと個別の割り当て

インタラクション方法タブで構成された処理ルールに優先して特定のオファーを割り当てるようにランタイム環境を構成できます。オーディエンス・レベルのあらゆるメンバーに対してグローバル・オファーを定義し、特定のオーディエンス・メンバーに対して個別の割り当てを定義できます。例えば、すべての世帯に対して、その他のものが有効でない場合に表示するグローバル・オファーを定義し、特定の Smith 世帯に対して、個別オファーの割り当てを作成できます。

ゾーン、セル、およびオファーの取り込みルールによって、グローバル・オファーと個別の割り当ての両方を制約できます。グローバル・オファーと個別の割り当ての両方は、運用プロファイル・データベースの特定のテーブルにデータを追加することによって構成されます。

グローバル・オファーと個別の割り当てを適切に機能させるには、配置内に、参照されるすべてのセルおよびオファー・コードが存在する必要があります。必要なデータを確実に有効にするには、デフォルトのセル・コードおよび UACI_ICBatchOffers テーブルを構成する必要があります。

デフォルトのセル・コードを定義するには

グローバル・オファーまたは個別オファーの割り当てのために、デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルを使用する場合、IndividualTreatment カテゴリーのオーディエンス・レベルおよびテーブル・タイプごとに DefaultCellCode プロパティーを定義することによって、デフォルトのセル・コードを定義する必要があります。

DefaultCellCode は、デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルで特定の行にセル・コードが定義されていない場合に使用されます。このデフォルトのセル・コードはレポートで使用されます。

DefaultCellCode は Campaign で定義されたセル・コード形式と一致する必要があります。このセル・コードはレポートに表示されるすべてのオファー割り当てに対して使用されます。一意のデフォルト・セル・コードを定義すると、デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルによって割り当てられたオファーを容易に識別できます。

UACI_ICBatchOffers テーブルを定義するには

デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルを使用する場合、すべてのオファー・コードが配置内に存在することを確認する必要があります。デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルで使用するすべてのオファーが処理ルールで使用されることがわかっている場合、それらのオファーは配置内に存在します。ただし、処理ルールで使用されないオファーについては、UACI_ICBatchOffers テーブルで定義する必要があります。

UACI_ICBatchOffers テーブルは Campaign システム・テーブル内に存在します。

UACI_ICBatchOffers テーブルには、デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルで使用されるオファー・コードを追加する必要があります。テーブルの形式は以下のとおりです。

列名	タイプ	説明
ICName	varchar(64)	オファーが関連付けられているインタラクティブ・チャネルの名前。2 つの異なるインタラクティブ・チャネルに同じオファーを使用している場合、インタラクティブ・チャネルごとに行を指定する必要があります。
OfferCode1	varchar(64)	オファー・コードの最初の部分。
OfferCode2	varchar(64)	オファー・コードの 2 番目の部分 (必要な場合)。
OfferCode3	varchar(64)	オファー・コードの 3 番目の部分 (必要な場合)。
OfferCode4	varchar(64)	オファー・コードの 4 番目の部分 (必要な場合)。
OfferCode5	varchar(64)	オファー・コードの 5 番目の部分 (必要な場合)。

グローバル・オファー・テーブルについて

グローバル・オファー・テーブルを使用すると、オーディエンス・レベルで処理を定義できます。例えば、オーディエンスの世帯のすべてのメンバーに対してグローバル・オファーを定義できます。

Interact オファー提供の以下の要素についてグローバル設定を定義できます。

- グローバル・オファーの割り当て
- グローバル・マーケティング担当者のスコア (数値または式による)
- オファーをフィルターに掛けるためのブール式
- 学習の可能性および重み (Interact 組み込み学習を使用している場合)

- グローバル学習のオーバーライド

グローバル・オファー・テーブルを有効にするには

処理ルールで定義されたすべての設定に優先してオーディエンス・レベルに対してグローバル・オファーを割り当てるようにランタイム環境を構成できます。

1. プロファイル・データベースに UACI_DefaultOffers というテーブルを作成します。

   aci_usrtab DDL ファイルを使用して、適切な列を持つ UACI_DefaultOffers テーブルを作成できます。

2. enableDefaultOfferLookup プロパティーを **true** に設定します。

グローバル・オファー・テーブル

グローバル・オファー・テーブルがプロファイル・データベースに存在している必要があります。プロファイル・データベースに対して aci_usertab SQL スクリプトを実行することにより、グローバル・オファー・テーブル、UACI_DefaultOffers を作成できます。aci_usertab SQL スクリプトは、ランタイム環境のインストール・ディレクトリー内の ddl ディレクトリーにあります。

各行に AudienceLevel フィールドおよび OfferCode1 フィールドを定義する必要があります。その他のフィールドは、オファーの割り当てをさらに制約したり、オーディエンス・レベルでの組み込み学習に影響を与えたりするオプションです。

最良のパフォーマンスを得るには、このテーブルのオーディエンス・レベル列でインデックスを作成してください。

名前	タイプ	説明
AudienceLevel	varchar(64)	(必須) デフォルトのオファーを割り当てるオーディエンス・レベルの名前 (例えば、customer や household)。この名前は、Campaign で定義されているオーディエンス・レベルと一致する必要があります。
OfferCode1	varchar(64)	(必須) デフォルトのオファーに対するオファー・コード。オファー・コードが複数のフィールドから構成されている場合、追加の列、例えば OfferCode2 などを追加できます。 グローバル・オファーの割り当てを指定するためにこのオファーを追加している場合は、このオファーを UACI_ICBatchOffers テーブルに追加する必要があります。
Score	float	このオファーの割り当てに対するマーケティング・スコアを定義するための数値。

名前	タイプ	説明
OverrideTypeID	int	1 に設定されている場合、オファーがオファーの候補リストに存在しなければ、このオファーに対してスコア・データを使用するだけでなく、このオファーをリストに追加します。通常、グローバル・オファーの割り当てを指定するには 1 を使用します。 0、*null*、または 1 以外の数に設定されている場合、オファーがオファーの候補リストに存在する場合にのみオファーに対してデータを使用します。多くの場合、処理ルールまたは個別の割り当てによって、この設定はオーバーライドされます。
Predicate	varchar(4000)	処理ルールの詳細オプションに関して、この列に式を入力できます。処理ルールの詳細オプションを書き込むときに使用可能な変数およびマクロと同じものを使用できます。この列の動作は、EnableStateID 列の値によって異なります。 • EnableStateID が 2 の場合、この列は処理ルールの詳細オプションの「**次の式が TRUE の場合は、このルールを対象と見なす**」オプションと同じように機能し、このオファーの割り当てが制約されます。この列にはブール式が含まれる必要があり、このオファーが組み込まれるために TRUE に解決される必要があります。 誤って、数値に解決される式を定義した場合、ゼロ以外の数値は TRUE と見なされ、ゼロは FALSE と見なされます。 • EnableStateID が 3 の場合、この列は処理ルールの詳細オプションの「**次の式をマーケティング・スコアとして使用する**」オプションと同じように機能し、このオファーが制約されます。この列には、数値に解決される式が含まれる必要があります。 • EnableStateID が 1 の場合、Interact ではこの列の値は無視されます。
FinalScore	float	返されるオファーの最終リストを順序付けるために使用する最終スコアをオーバーライドする数値。この列は、組み込み学習モジュールを有効にしている場合に使用されます。この列を使用する独自の学習を実装できます。

名前	タイプ	説明
CellCode	varchar(64)	このデフォルトのオファーを割り当てるインタラクティブ・セグメントに対するセル・コード。セル・コードが複数のフィールドから構成されている場合、別の列を追加できます。 OverrideTypeID が 0 または null の場合、セル・コードを指定する必要があります。セル・コードが含まれない場合、ランタイム環境ではこの行のデータは無視されます。 OverrideTypeID が 1 の場合、この列にセル・コードを指定する必要はありません。セル・コードを指定しない場合、ランタイム環境では、レポート目的でこのオーディエンス・レベルおよびテーブルに対して DefaultCellCode プロパティーで定義されたセル・コードが使用されます。
Zone	varchar(64)	このオファーの割り当てを適用するゾーンの名前。NULL の場合、これはすべてのゾーンに適用されます。
EnableStateID	int	この列の値は Predicate 列の動作を定義します。 • **1** - Predicate 列を使用しません。 • **2** - オファーをフィルターに掛けるブールとして Predicate を使用します。処理ルールの「**次の式が TRUE の場合は、このルールを対象と見なす**」詳細オプションと同じルールに従います。 • **3** - マーケティング担当者のスコアを定義するために Predicate を使用します。処理ルールの「**次の式をマーケティング・スコアとして使用する**」詳細オプションと同じルールに従います。 この列が NULL であるまたは 2 と 3 以外の値である行では、Predicate 列は無視されます。
LikelihoodScore	float	この列は組み込み学習に影響を与えるためにのみ使用されます。aci_scoringfeature DDL を使用するとこの列を追加できます。
AdjExploreScore	float	この列は組み込み学習に影響を与えるためにのみ使用されます。aci_scoringfeature DDL を使用するとこの列を追加できます。

スコア・オーバーライド・テーブルについて

スコア・オーバーライド・テーブルを使用すると、オーディエンス ID または個別のレベルで処理を定義できます。例えば、オーディエンス・レベルが訪問者である場合、特定の訪問者に対してオーバーライドを作成できます。

Interact オファー提供の以下の要素についてオーバーライドを定義できます。

- 個別オファーの割り当て
- 個別のマーケティング担当者のスコア (数値または式による)

- オファーをフィルターに掛けるためのブール式
- 学習の可能性および重み (組み込み学習を使用している場合)
- 個別の学習のオーバーライド

スコア・オーバーライド・テーブルを有効にするには

マーケティング・スコアの代わりにモデリング・アプリケーションから生成されるスコアを使用するように Interact を構成できます。

1. オーバーライドを指定するオーディエンス・レベルごとにスコア・オーバーライド・テーブルを作成します。

   aci_usrtab DDL ファイルを使用して、適切な列を持つサンプルのスコア・オーバーライド・テーブルを作成できます。
2. enableScoreOverrideLookup プロパティーを **true** に設定します。
3. scoreOverrideTable プロパティーに、オーバーライドを指定するオーディエンス・レベルごとのスコア・オーバーライド・テーブルの名前を設定します。

   すべてのオーディエンス・レベルに対してスコア・オーバーライド・テーブルを指定する必要はありません。

スコア・オーバーライド・テーブル

スコア・オーバーライド・テーブルが運用プロファイル・データベースに存在している必要があります。サンプルのスコア・オーバーライド・テーブル UACI_ScoreOverride は、プロファイル・データベースに対して aci_usertab SQL スクリプトを実行することで作成できます。aci_usertab SQL スクリプトは、ランタイム環境のインストール・ディレクトリー内の ddl ディレクトリーにあります。

各行に *AudienceID* フィールド、OfferCode1 フィールド、および Score フィールドを定義する必要があります。その他のフィールドの値は、個別オファーの割り当てをさらに制約したり、組み込み学習のためのスコア・オーバーライド情報を指定したりするオプションです。

名前	タイプ	説明
AudienceID	varchar(64)	(必須) この列の名前は、Campaign のオーディエンス ID を定義している列の名前と一致する必要があります。aci_usertab DDL ファイルによって作成されたサンプル・テーブルでは、この列は CustomerID 列として作成されます。オーディエンス ID が複数の列から構成されている場合、それらの列をこのテーブルに追加できます。各行には、個別オファー、例えば customer1 が割り当てられたオーディエンス ID が含まれている必要があります。最良のパフォーマンスを得るには、この列でインデックスを作成してください。

名前	タイプ	説明
OfferCode1	varchar(64)	(必須) オファーに対するオファー・コード。オファー・コードが複数のフィールドから構成されている場合、追加の列、例えば OfferCode2 などを追加できます。 個別オファーの割り当てを指定するためにこのオファーを追加している場合は、このオファーを UACI_ICBatchOffers テーブルに追加する必要があります。
Score	float	このオファーの割り当てに対するマーケティング・スコアを定義するための数値。
OverrideTypeID	int	0 または *null* (または 1 以外の数) に設定されている場合、オファーがオファーの候補リストに存在する場合にのみオファーに対してデータを使用します。通常、スコア・オーバーライドを指定するには 0 を使用します。セル・コードを指定する必要があります。 1 に設定されている場合、オファーがオファーの候補リストに存在しなければ、このオファーに対してスコア・データを使用するだけでなく、このオファーをリストに追加します。通常、個別オファーの割り当てを指定するには 1 を使用します。
Predicate	varchar(4000)	処理ルールの詳細オプションに関して、この列に式を入力できます。処理ルールの詳細オプションを書き込むときに使用可能な変数およびマクロと同じものを使用できます。この列の動作は、EnableStateID 列の値によって異なります。 • EnableStateID が 2 の場合、この列は処理ルールの詳細オプションの「**次の式が TRUE の場合は、このルールを対象と見なす**」オプションと同じように機能し、このオファーの割り当てが制約されます。この列にはブール式が含まれる必要があり、このオファーが組み込まれるために TRUE に解決される必要があります。 誤って、数値に解決される式を定義した場合、ゼロ以外の数値は TRUE と見なされ、ゼロは FALSE と見なされます。 • EnableStateID が 3 の場合、この列は処理ルールの詳細オプションの「**次の式をマーケティング・スコアとして使用する**」オプションと同じように機能し、このオファーが制約されます。この列には、数値に解決される式が含まれる必要があります。 • EnableStateID が 1 の場合、Interact ではこの列の値は無視されます。

名前	タイプ	説明
FinalScore	float	返されるオファーの最終リストを順序付けるために使用する最終スコアをオーバーライドする数値。この列は、組み込み学習モジュールを有効にしている場合に使用されます。この列を使用する独自の学習を実装できます。
CellCode	varchar(64)	このオファーを割り当てるインタラクティブ・セグメントに対するセル・コード。セル・コードが複数のフィールドから構成されている場合、別の列を追加できます。 OverrideTypeID が 0 または null の場合、セル・コードを指定する必要があります。セル・コードが含まれない場合、ランタイム環境ではこの行のデータは無視されます。 OverrideTypeID が 1 の場合、この列にセル・コードを指定する必要はありません。セル・コードを指定しない場合、ランタイム環境では、レポート目的でこのオーディエンス・レベルおよびテーブルに対して DefaultCellCode プロパティーで定義されたセル・コードが使用されます。
Zone	varchar(64)	このオファーの割り当てを適用するゾーンの名前。NULL の場合、これはすべてのゾーンに適用されます。
EnableStateID	int	この列の値は Predicate 列の動作を定義します。 • **1** - Predicate 列を使用しません。 • **2** - オファーをフィルターに掛けるブールとして Predicate を使用します。処理ルールの「**次の式が TRUE の場合は、このルールを対象と見なす**」詳細オプションと同じルールに従います。 • **3** - マーケティング担当者のスコアを定義するために Predicate を使用します。処理ルールの「**次の式をマーケティング・スコアとして使用する**」詳細オプションと同じルールに従います。 この列が NULL であるまたは 2 と 3 以外の値である行では、Predicate 列は無視されます。
LikelihoodScore	float	この列は組み込み学習に影響を与えるためにのみ使用されます。aci_scoringfeature DDL を使用するとこの列を追加できます。
AdjExploreScore	float	この列は組み込み学習に影響を与えるためにのみ使用されます。aci_scoringfeature DDL を使用するとこの列を追加できます。

Interact 組み込み学習の概要

適切なオファーを適切なセグメントに提示するために可能なことをすべて行う一方で、訪問者の実際の選択から学べるものが常にあります。訪問者の実際の行動は戦略に影響を及ぼします。レスポンス履歴を取得し、それをいくつかのモデリング・ツールによって実行して、インタラクティブ・フローチャートに組み込むことのできるスコアを得ることができます。ただし、これはリアルタイムのデータではありません。

Interact では、訪問者のアクションからリアルタイムで学習するために、2 つのオプションが提供されています。

- 組み込み学習モジュール — ランタイム環境には、ナイーブ・ベイズに基づく学習モジュールがあります。このモジュールでは、選択した顧客属性が監視され、提示すべきオファーの選択を補助するためにそのデータが使用されます。
- 学習 API — ランタイム環境には、独自の学習モジュールを記述するための学習 API もあります。

学習は使用しなくてもかまいません。デフォルトでは、学習は無効にされています。

Interact の学習の理解

Interact 学習モジュールでは、オファーに対する訪問者のレスポンスおよび訪問者属性が監視されます。学習モジュールには、次の 2 つの一般的なモードがあります。

- **探索** — 後の活用モード中に使用される見積もりを最適化するのに十分なレスポンス・データを収集するために、学習モジュールでオファーが提供されます。探索中に提供されるオファーは、必ずしも最適の選択を反映するものではありません。
- **活用** — 探索の段階で十分なデータが収集された後、学習モジュールでは、提示するオファーの選択を補助するために可能性が使用されます。

学習モジュールでは、`confidenceLevel` プロパティーで構成する信頼性レベルと、`percentRandomSelection` プロパティーで構成する、学習モジュールによりランダム・オファーが提示される可能性という、2 つのプロパティーに基づいて、探索と活用が交互に繰り返されます。

`confidenceLevel` を、オファーに対するスコアがアービトレーションで使用される前に、学習モジュールがどの程度確かであるか (または信頼されるべきか) を表す割合に設定します。最初に、学習モジュールに処理するデータがない場合、学習モジュールではマーケティング・スコアが全面的に信頼されます。`minPresentCountThreshold` で定義されている回数だけすべてのオファーが提示された後、学習モジュールは探索モードに入ります。処理するデータが多くない場合、学習モジュールでは、計算された割合が正しいと確信されません。したがって、学習モジュールは探索モードのままです。

学習モジュールでは、各オファーに重みが割り当てられます。重みを計算するため、学習モジュールでは、受け入れデータの履歴と現在のセッション・データに加え、構成された信頼性レベルを入力として取り入れる式が使用されます。式によって、本質的に探索と活用の間のバランスが取られ、適切な重みが返されます。

初期段階中にシステムが良い結果を出すオファーに偏らないように、Interact では percentRandomSelection パーセント分の回数、ランダム・オファーが提示されます。これによって、学習モジュールでは、最も成功しそうなオファー以外のオファーが強制的に提示され、そのようなオファーがより多く提示された場合にオファーの成功の可能性が高くなるかどうかが判別されます。例えば、percentRandomSelection を 5 に構成した場合、5% 分の回数、学習モジュールではランダム・オファーが提示され、そのレスポンス・データが計算に追加されることを意味します。

学習モジュールでは、提示されるオファーが以下の方法で決定されます。

1. 訪問者がオファーを選択する可能性が計算されます。
2. 手順 1 の可能性を使用してオファーの重みが計算され、探索モードか活用モードかどちらにすべきかが決定されます。
3. マーケティング・スコアおよび手順 2 のオファーの重みを使用して、オファーごとに最終スコアが計算されます。
4. 手順 3 で決定されたスコアに基づいてオファーがソートされ、要求された数の上位オファーが返されます。

例えば、学習モジュールでは、訪問者がオファー A を受け入れる可能性は 30%、オファー B を受け入れる可能性は 70% であると判断され、この情報を活用すべきだと判断されるとします。処理ルールでは、オファー A のマーケティング・スコアは 75、オファー B は 55 です。ただし、手順 3 の計算では、オファー A よりオファー B の方が最終スコアが高くなっているため、ランタイム環境ではオファー B が提示されます。

学習は、recencyWeightingFactor プロパティーおよび recencyWeightingPeriod プロパティーにも基づきます。これらのプロパティーを使用すると、古いデータに対してより、最近のデータに対して、より重みを加えることができます。recencyWeightingFactor は、最近のデータが持つ重みの割合です。recencyWeightingPeriod は、最近であると見なされる期間です。例えば、recencyWeightingFactor を .30 に、recencyWeightingPeriod を 24 に構成します。これは、過去 24 時間のデータが、考慮されるすべてのデータの 30% を占めることを意味します。1 週間分のデータがある場合、最初の 6 日間にわたって平均されたすべてのデータが 70% であり、最終日のデータが 30% になります。

すべてのセッションで、以下のデータが学習ステージング・テーブルに書き込まれます。

- オファー・コンタクト
- オファーの受け入れ
- 学習属性

構成可能な間隔で、集約機能によってステージング・テーブルからデータが読み取られ、そのデータが編集されてテーブルに書き込まれます。学習モジュールでは、この集約データが読み取られて計算に使用されます。

学習モジュールを有効にするには

すべてのランタイム・サーバーには、組み込み学習モジュールが備わっています。デフォルトでは、この学習モジュールは無効にされています。構成プロパティーを変更することによって学習を有効にすることができます。

ランタイム環境の Marketing Platform で、「Interact」>「offerserving」カテゴリーの以下の構成プロパティーを編集します。

構成プロパティー	設定
optimizationType	**BuiltInLearning**

学習属性

学習モジュールでは、訪問者属性とオファー受け入れデータを使用して学習されます。監視する訪問者属性を選択できます。これらの訪問者属性には、インタラクティブ・フローチャートで参照するディメンション・テーブルに保管されている属性や、リアルタイムに収集するいくつかのイベント・パラメーターを含め、顧客プロファイル内のものはどれも選択可能です。

監視する属性はいくつでも構成できますが、IBM では、以下のガイドラインに従って、静的学習属性と動的学習属性間で 10 個程度までの学習属性を構成することをお勧めしています。

- 独立した属性を選択してください。

  類似した属性を選択しないでください。例えば、HighValue という属性を作成して、その属性が給料に基づく計算によって定義される場合、HighValue と Salary の両方は選択しないでください。類似の属性は学習アルゴリズムには役立ちません。
- 離散的な値を持つ属性を選択してください。

  属性に値の範囲がある場合、厳密な値を選択する必要があります。例えば、属性として給料を使用する場合、各給料の範囲に特定の値 (範囲 20,000 から 30,000 までは A、30,001 から 40,000 までは B など) を指定する必要があります。
- パフォーマンスの妨げにならないように、追跡する属性の数を制限してください。

  追跡できる属性の数は、パフォーマンス要件と Interact インストール済み環境に応じて異なります。可能な場合は、別のモデリング・ツール (PredictiveInsight など) を使用して、上位 10 個の予測属性を判別してください。予測されない属性を自動的に除去するように学習モジュールを構成できますが、これにはパフォーマンス・コストも掛かります。

監視する属性の数と監視する属性ごとの値の数の両方を定義することによって、パフォーマンスを管理できます。maxAttributeNames プロパティーでは、追跡する訪問者属性の最大数を定義します。maxAttributeValues プロパティーでは、属性ごとに追跡する値の最大数を定義します。その他のすべての値は、otherAttributeValue プロパティーの値によって定義されたカテゴリーに割り当てられます。ただし、学習エンジンは、検出された最初の値のみを追跡します。例えば、訪問者属性の目の

色を追跡するとします。対象とする値を、青色、茶色、および緑色のみとして、maxAttributeValues を 3 に設定します。しかし、最初の 3 人の訪問者の値は、青色、茶色、およびヘーゼルでした。この場合、緑色の目のすべての訪問者は otherAttributeValue に割り当てられます。

学習基準をより詳細に定義できる、動的学習属性を使用することもできます。動的学習属性を使用すると、2 つの属性の組み合わせを単一のエントリーとして学習させることができます。例えば、以下のプロファイル情報を考慮します。

訪問者 ID	カード・タイプ	カード残高
1	ゴールド・カード	$1,000
2	ゴールド・カード	$9,000
3	ブロンズ・カード	$1,000
4	ブロンズ・カード	$9,000

標準学習属性を使用する場合、カード・タイプとカード残高を個別にのみ学習させることができます。訪問者 1 と 2 は、カード・タイプに基づいて同じグループに分類され、訪問者 2 と 4 は、カード残高に基づいて同じグループに分類されます。これは、オファーの受け入れ行動の正確な予測子にならない場合があります。ゴールド・カード保有者は残高がより高額になる傾向があるとすると、訪問者 2 の行動は訪問者 4 とは根本的に異なる可能性があり、そのことは標準学習属性を偏らせます。しかし、動的学習属性を使用すると、これらの訪問者ごとに個別に学習され、予測はより正確になります。

動的学習属性を使用し、訪問者が 1 つの属性に対して 2 つの有効な値を持つ場合、学習モジュールでは検出された最初の値が選択されます。

enablePruning プロパティーを yes に設定している場合、学習モジュールのアルゴリズムによって、予測されない属性が判別され、これらの属性は重みの計算時に考慮の対象から外されます。例えば、髪色を表す属性を追跡しており、学習モジュールによって、訪問者の髪色に基づいてオファーを受け入れるパターンが存在しないと判別された場合、学習モジュールでは、髪色の属性を考慮することは中止されます。属性は、学習集計プロセスの実行 (aggregateStatsIntervalInMinutes プロパティーで定義されている) ごとに再評価されます。動的学習属性も除去されます。

学習属性を定義するには

訪問者の属性を maxAttributeNames の数まで構成できます。

設計環境の Marketing Platform で、「Campaign」>「パーティション」>「パーティション*n*」>「Interact」>「ラーニング」カテゴリーの以下の構成プロパティーを編集します。

(learningAttributes) は、新規学習属性を作成するためのテンプレートです。属性ごとに新規名を入力する必要があります。同じ名前で 2 つのカテゴリーを作成することはできません。

構成プロパティー	設定
attributeName	attributeName は、プロファイル・データの名前と値のペアの名前と一致している必要があります。この名前の大/小文字は区別されません。

動的学習属性を定義するには

動的学習属性を定義するには、学習データ・ソースの UACI_AttributeList テーブルにデータを追加する必要があります。

以下の表のすべての列のタイプは varchar(64) です。

列	説明
AttributeName	動的学習属性の名前。これは、AttributeNameCol で考えられる実際の値でなければなりません。
AttributeNameCol	AttributeName がある完全修飾列名 (プロファイル・テーブルから始まる階層構造)。この列は、標準学習属性である必要はありません。
AttributeValueCol	AttributeName の関連値がある完全修飾列名 (プロファイル・テーブルから始まる階層構造)。

例えば、以下のプロファイル・テーブルとその関連ディメンション・テーブルについて考えます。

表 5. MyProfileTable

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

表 6. MyDimensionTable

KeyField	CardType	CardBalance
Key1	ゴールド・カード	1000
Key2	ゴールド・カード	9000
Key3	ブロンズ・カード	1000
Key4	ブロンズ・カード	9000

以下は、カードの種類と残高をマッチングした UACI_AttributeList テーブル例です。

表 7. UACI_AttributeList

AttributeName	AttributeNameCol	AttributeValueCol
ゴールド・カード	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance

表 7. *UACI_AttributeList (続き)*

AttributeName	AttributeNameCol	AttributeValueCol
ブロンズ・カード	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance

外部学習を有効にするには

学習 Java API を使用して、独自の学習モジュールを作成できます。Marketing Platform の学習ユーティリティーを認識するために、ランタイム環境を構成する必要があります。

ランタイム環境の Marketing Platform で、「Interact」>「offerserving」カテゴリーの以下の構成プロパティーを編集します。 Learning Optimizer API の構成プロパティーは、「Interact」>「offerserving」>「外部学習構成 (External Learning Config)」カテゴリーにあります。

構成プロパティー	設定
optimizationType	**ExternalLearning**
externalLearningClass	外部学習のクラス名
externalLearningClassPath	外部学習用のランタイム・サーバーのクラス・ファイルまたは JAR ファイルへのパス。サーバー・グループを使用していて、すべてのランタイム・サーバーが Marketing Platform の同じインスタンスを参照する場合、すべてのサーバーの同じ場所にクラス・ファイルまたは JAR ファイルのコピーを置く必要があります。

これらの変更を有効にするために、Interact ランタイム・サーバーを再始動する必要があります。

第 5 章 Interact API の理解

Interact サーバーは、さまざまなタッチポイントに対して動的にオファーを提供します。例えば、特定のタイプの問い合わせを行った顧客に対して見込める最も高額での販売または抱き合わせ販売について通知するメッセージをコール・センターの従業員に送るように、ランタイム環境とタッチポイントを構成することができます。また、Web サイトの特定の領域に入った顧客 (訪問者) に合わせて調整されたオファーを提供するように、ランタイム環境とタッチポイントを構成することもできます。

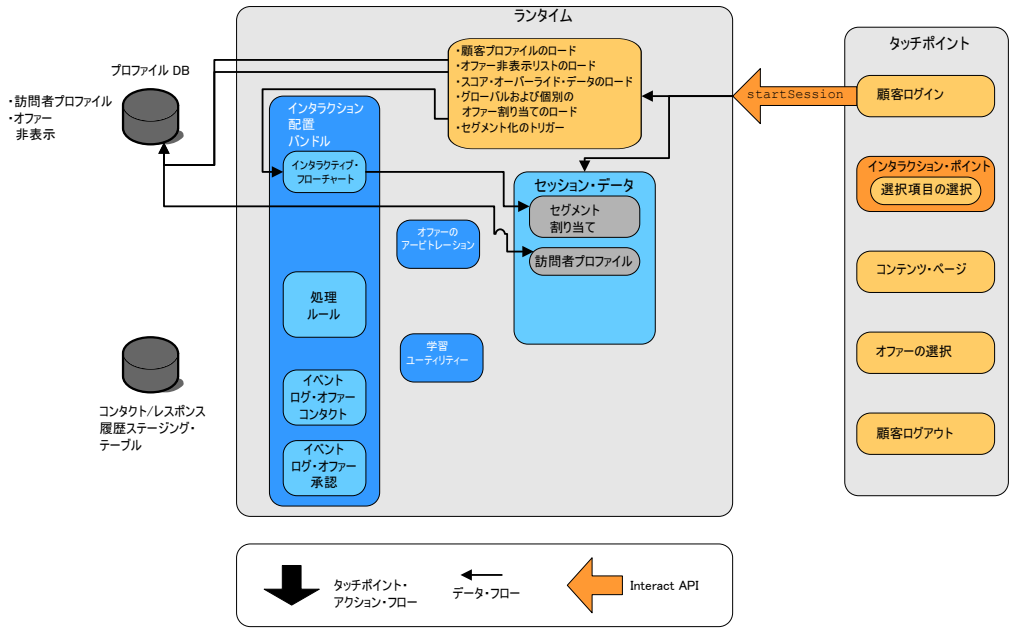
Interact アプリケーション・プログラミング・インターフェース (API) を使用すると、できる限り最適なオファーを協力して提供するように、タッチポイントとランタイム・サーバーを構成することができます。API を使用することで、タッチポイントはランタイム・サーバーからの情報を要求し、訪問者をグループ (セグメント) に割り当てて、そのセグメントに基づいたオファーを表示することができます。また、後で分析してオファーの表示戦略を洗練するために、データをログに記録することもできます。

ご使用の環境に Interact を統合する際の柔軟性をできるだけ高くするために、IBM では Interact API を使用してアクセスできる Web サービスを提供しています。

Interact API データ・フロー

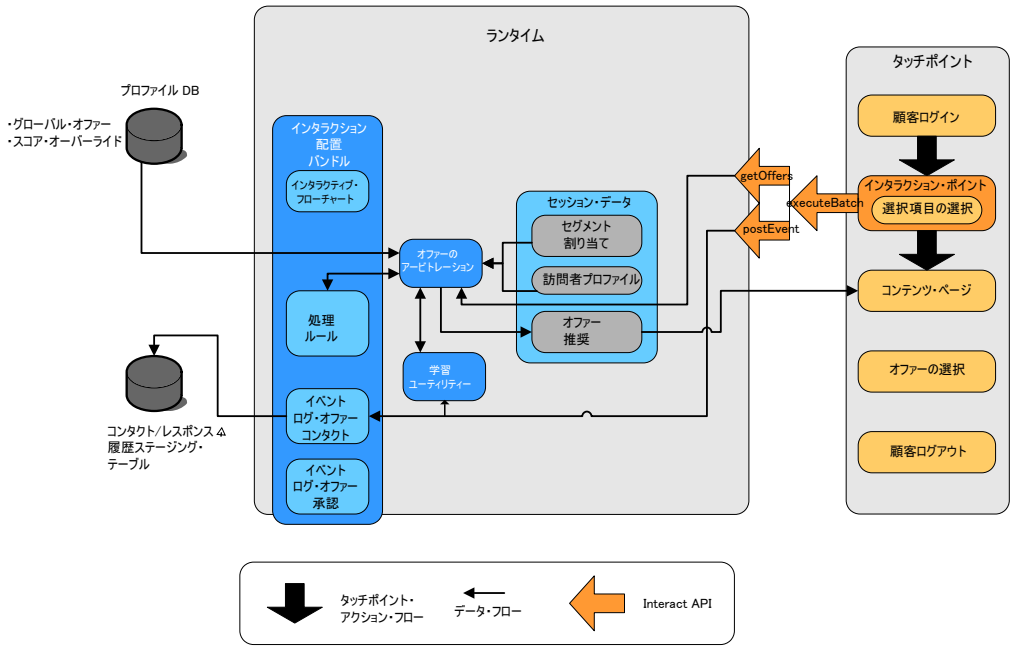
次の図は、Interact API の単純な実装を示しています。訪問者は Web サイトにログインすると、オファーが表示されるページに移動します。訪問者はオファーを選択して、ログアウトします。対話は単純ですが、タッチポイントとランタイム・サーバーの両方でいくつかのイベントが発生します。

訪問者がログインすると、`startSession` がトリガーされます。



この例では、startSession メソッドは 4 つのことを行います。最初に、新しいランタイム・セッションを作成します。2 番目に、顧客のプロファイル・データをセッションにロードするように求める要求を送信します。3 番目に、そのプロファイル・データを使用してインタラクティブ・フローチャートを開始し、その顧客をセグメントに配置するように求める要求を送信します。このフローチャートの実行は非同期です。4 番目に、ランタイムは、オファー非表示、およびグローバル・オファーと個々のオファーの処理に関する情報を、そのセッションにロードします。セッション・データは、そのセッションが存続する間、メモリー内に保持されます。

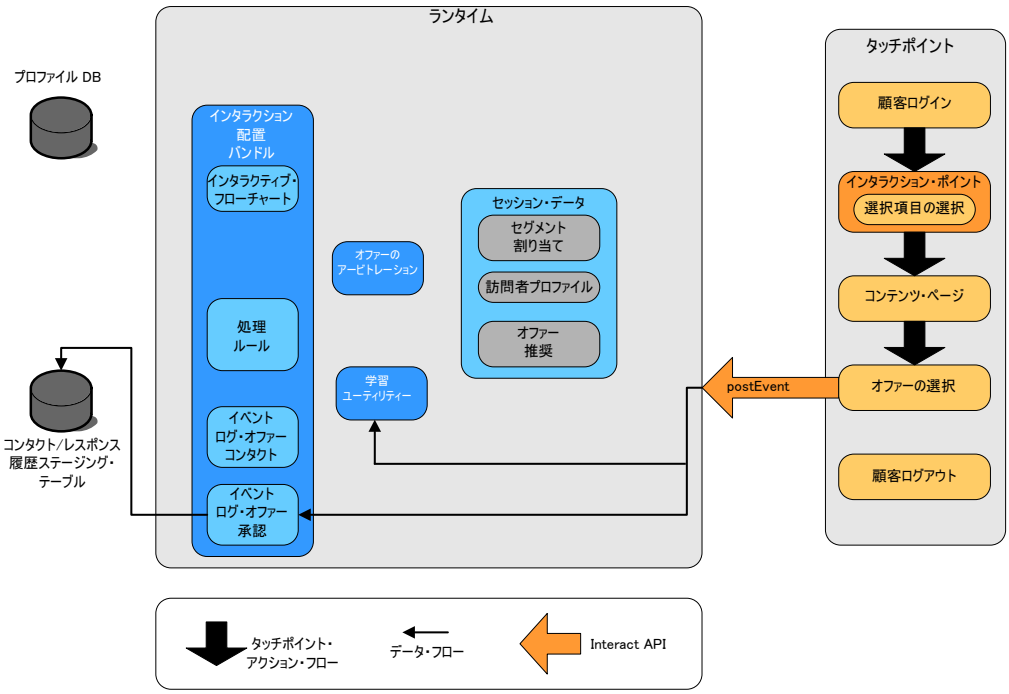
訪問者は、事前定義されているインタラクション・ポイントに到達するまで、サイトを移動します。将来的には、2 番目のインタラクション・ポイント (選択項目の選択) が、オファーのセットを表すリンクを訪問者がクリックする場所になります。executeBatch メソッドをトリガーするリンクは、タッチポイント・マネージャーによって構成されています。



executeBatch メソッドを使用することで、複数のメソッドを単一の呼び出しでランタイム・サーバーに呼び出すことができます。この特定の executeBatch は、他の 2 つのメソッド (getOffers および postEvent) を呼び出します。getOffers メソッドは、オファーのリストを要求します。ランタイムは、セグメント・データ、オファー非表示リスト、処理ルール、および学習モジュールを使用して、オファーのセットを提案します。ランタイムによって返されたオファーのセットは、コンテンツ・ページに表示されます。

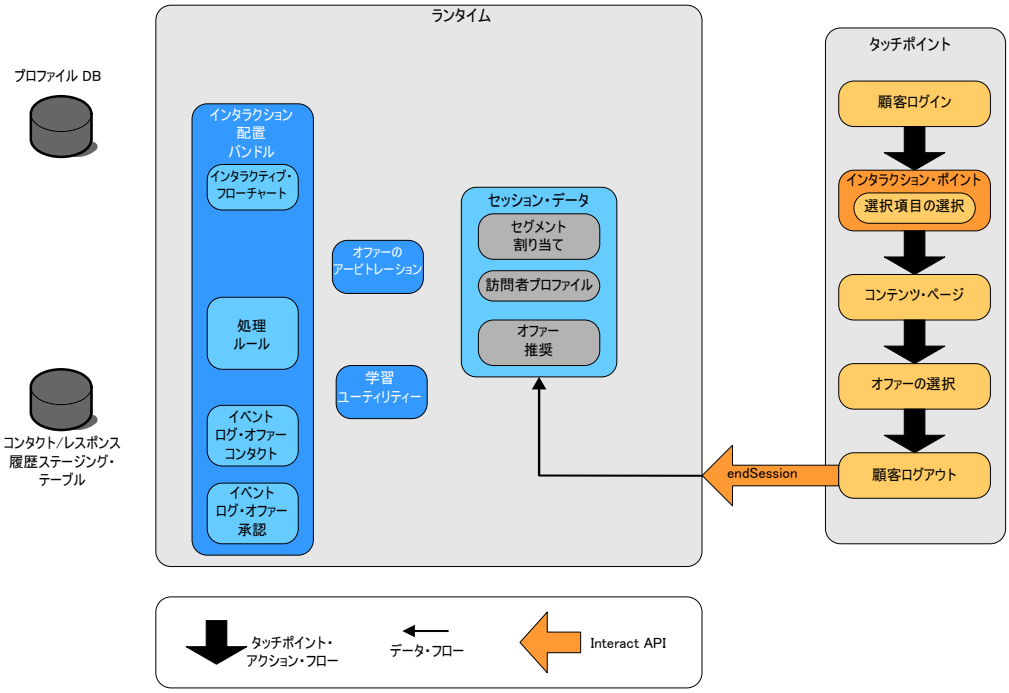
postEvent メソッドは、設計環境で定義されたイベントの 1 つをトリガーします。この特定の事例では、イベントは、表示されたオファーのログをコンタクト履歴に記録するように求める要求を送信します。

訪問者は、オファーの 1 つを選択します (オファーの選択)。



オファーの選択に関連付けられるボタンは、別の postEvent メソッドを送信するように構成されます。このイベントは、オファー承認のログをレスポンス履歴に記録するように求める要求を送信します。

訪問者は、オファーを選択したら、その Web サイトでの作業を終了し、ログアウトします。ログアウト・コマンドは、endSession メソッドにリンクされています。



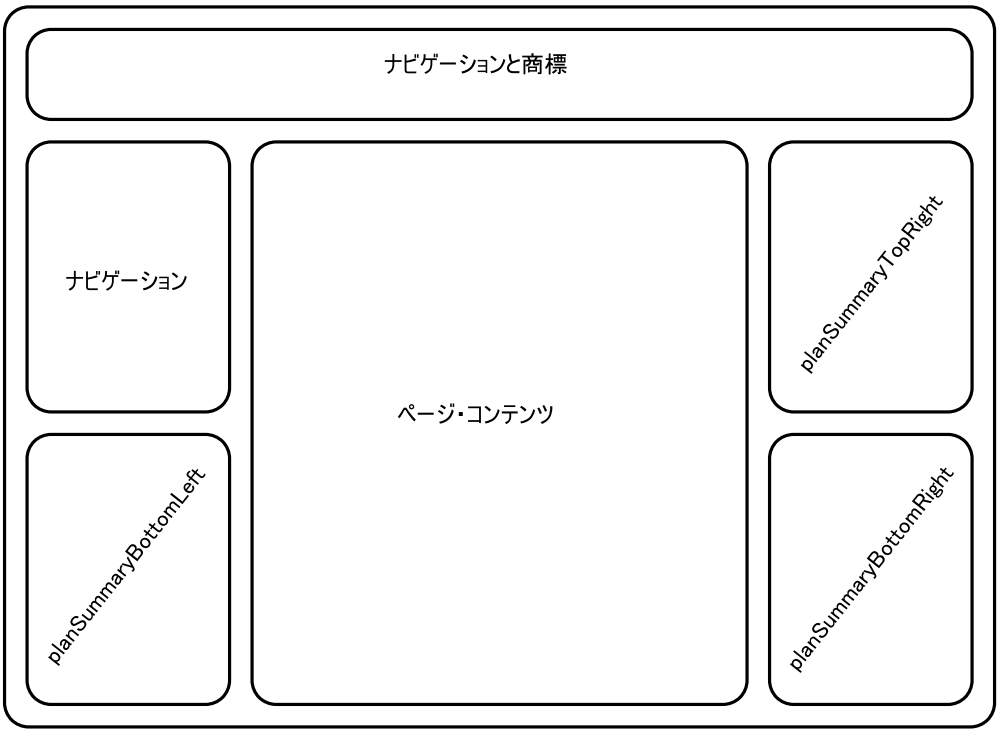
endSession メソッドはセッションをクローズします。訪問者がログアウトするのを忘れた場合に、確実にすべてのセッションが最終的には終了されるようにするために、構成可能なセッション・タイムアウトがあります。セッションに渡された任意

のデータ (例えば、startSession メソッドまたは setAudience メソッドのパラメーターに含まれる情報など) を保持する場合は、インタラクティブ・フローチャートを作成した人物と協力します。インタラクティブ・フローチャートを作成した人物は、セッションが終了してそのデータが失われる前に、スナップショット・プロセスを使用してそのデータをデータベースに書き込むことができます。スナップショット・プロセスに含まれるインタラクティブ・フローチャートは、postEvent メソッドを使用して呼び出すことができます。

この例は、API がタッチポイントとランタイム環境の間で行う基本的な動作を示した、非常に単純な例です (訪問者が、ログイン、オファーが表示されるページへの移動、オファーの選択、およびログアウトの 4 つのアクションのみを行う、単純な対話です)。統合の設計は、必要に応じて (パフォーマンス要件の範囲内で) 複雑にすることができます。

単純なインタラクション計画の例

携帯電話会社の Web サイトの対話を設計しているとします。次の図は、携帯電話プランのサマリー・ページのレイアウトを示しています。



携帯電話プランのサマリー・ページの要件を満たすために、以下の項目を定義します。

アップグレードに関するオファーの専用ゾーンに表示される単一のオファー

- ページ上でアップグレード・オファーを表示する領域を定義する必要があります。また、表示するオファーを Interact が選出したら、その情報をログに記録する必要があります。

  インタラクション・ポイント: ip_planSummaryBottomRight

イベント: evt_logOffer

電話のアップグレード用の 2 つのオファー

- ページ上で電話のアップグレードを表示する各領域を定義する必要があります。

インタラクション・ポイント: ip_planSummaryTopRight

インタラクション・ポイント: ip_planSummaryBottomLeft

分析用に、どのオファーが承認され、どのオファーが拒否されたかを、ログに記録する必要があります。

イベント: evt_offerAccept

イベント: evt_offerReject

また、オファーのコンタクト、承認、または拒否をログに記録するときは必ず処理コードを渡す必要もあります。次の例のように、必要に応じて NameValuePair を作成し、処理コードを格納します。

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

これで、タッチポイントとの統合のコーディングを開始すると同時に、設計環境のユーザーに、インタラクション・ポイントおよびイベントの作成を要求することができます。

オファーを表示するインタラクション・ポイントごとに、まず最初にオファーを取得してから、そのオファーの表示に必要な情報を抽出する必要があります。例えば、Web ページの右下の領域のオファーを要求します (planSummaryBottomRight)。

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

これは、OfferList レスポンスを含むレスポンス・オブジェクトを返します。ただし、Web ページでは OfferList オブジェクトは使用できません。オファー属性の 1 つであることがわかっている、オファーのイメージ・ファイル (offerImg) が必要です。必要なオファー属性は、OfferList から抽出する必要があります。

```
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()+ " >   */
  }
 }
}
```

これでオファーが表示されるので、そのログをコンタクトとして記録します。

```
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)
```

これらの各メソッドを呼び出す代わりに、Web ページの planSummaryBottomLeft 部分に対して、次の例のように executeBatch メソッドを使用することができます。

```
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);
```

この例では、UACIOfferTrackingCode を指定しない場合、Interact ランタイム・サーバーが最後に推奨された処理リストを自動的にコンタクトとしてログに記録するため、UACIOfferTrackingCode を定義する必要はありません。

また、電話のアップグレードのオファーを表示するページの 2 番目の領域に表示するイメージを 30 秒ごとに変更するように書き込んだとします。3 つのイメージを交替で表示することを決めたため、以下を使用してオファーのセットをリトリーブし、キャッシュに入れて、イメージを交替させるためのコード内で使用する必要があります。

```
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
  }
 }
}
```

オファーごとに、そのイメージが表示された後で 1 回のみローカル・キャッシュからフェッチし、コンタクトにログを記録するクライアント・コードを作成する必要があります。コンタクトのログを記録するには、以前と同様に UACITrackingCode パラメーターをポストする必要があります。オファーごとに異なるトラッキング・コードがあります。

```
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);
  }
 }
}
```

オファーごとに、そのオファーがクリックされたら、承認されたオファーと拒否されたオファーをログに記録する必要があります。(このシナリオでは、明示的に選択されないオファーは拒否されたものと見なされます。) 次の例は、ip_planSummaryTopRight オファーが選択される場合です。

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

実際には、これら 3 つの postEvent 呼び出しを、executeBatch メソッドとともに送信することをお勧めします。

これは基本的な例であり、統合の最も良い書き方を示したものではありません。例えば、これらの例のいずれにも、レスポンス・クラスを使用したエラー・チェックは含まれていません。

Interact API 統合の設計

タッチポイントと Interact API の統合を構築するには、実装を開始する前にいくつかの設計を行う必要があります。マーケティング・チームと協力して、ランタイム環境がタッチポイントのどこでオファーを提供するか (インタラクション・ポイントの定義)、および他のどの種類のトラッキング機能または対話機能を使用するか (イベントの定義) を決める必要があります。設計段階では、これらは概要にすぎな

い可能性があります。例えば、ある通信会社の Web サイトでは、顧客のプランのサマリー・ページに、プランのアップグレードに関するオファーを 1 つと電話のアップグレード用のオファーを 2 つ表示する必要があります。

自分が所属する会社が顧客との対話を行う場所と対話方法を決定したら、Interact を使用して詳細を定義する必要があります。フローチャートの作成者は、再セグメント・イベントの発生時に使用されるインタラクティブ・フローチャートを設計する必要があります。インタラクション・ポイントおよびイベントの数と名前を決める必要があります。また、適切なセグメント、イベント通知、およびオファーのリトリーブの際に渡される必要があるデータも決める必要があります。設計環境のユーザーが、インタラクティブ・チャネル用のインタラクティブ・ポイントおよびイベントを定義します。その後、ランタイム環境でご使用のタッチポイントとの統合をコーディングする際に、それらの名前を使用します。また、オファーのコンタクトとレスポンスをどの時点でログに記録する必要があるかを定義するために必要なメトリック情報も、定義する必要があります。

考慮事項

統合を書き込む際には、以下のヒントについて考慮してください。

- タッチポイントの設計時に、いくつかのデフォルトの充てん文字コンテンツ (通常は、害のない商標メッセージや空のコンテンツ) を、オファーが表示されるインタラクション・ポイントごとに作成します。これは、現行の状態で現行の訪問者に提供するのに適したオファーがない場合に使用されます。このデフォルトの充てん文字コンテンツは、そのインタラクション・ポイントのデフォルトのストリングとして割り当てる必要があります。
- 予期されていない何らかの理由によってタッチポイントがランタイム・サーバー・グループに到達できなかった場合のために、タッチポイントの設計時に、コンテンツを表すいくつかのメソッドを組み込みます。
- 訪問者を再セグメントするイベント (`postEvent` および `setAudience` を含む) のトリガー時には、フローチャートの実行に多少の時間がかかることを覚えておいてください。`getOffers` メソッドは、セグメント化が終了するまで待ってから実行されます。再セグメントの頻度が高すぎると、`getOffers` 呼び出しのレスポンスのパフォーマンスが低下する場合があります。
- 「オファーの拒否」が何を意味するかを決める必要があります。いくつかのレポート (「チャネル・オファーのパフォーマンスのサマリー (Channel Offer Performance Summary)」レポートなど) は、オファーが拒否された回数を表します。これは、オファー拒否をログに記録アクションが `postEvent` によってトリガーされた回数です。オファー拒否をログに記録の対象を、実際の拒否 (「いいえ、結構です」というラベルが付いたリンクをクリックするなど) にするか、無視されるオファー (どれも選択されない 3 つの異なる広告バナーを表示するページなど) にするかを決める必要があります。
- 有効にすることで Interact のオファー選択を拡張できるオプション機能がいくつかあります。これには、学習、オファー非表示、個々のオファーの割り当て、およびその他のオファー・サービス提供の要素が含まれます。これらのオプション機能のうちのいくつを使用してインタラクションを拡張するかを決める必要があります (もしあれば)。

第 6 章 IBM Unica Interact API の管理

startSession メソッドの使用時には、必ず、ランタイム・サーバー上に Interact ランタイム・セッションを作成します。構成プロパティーを使用して、ランタイム・サーバー上のセッションを管理することができます。タッチポイントと Interact の統合の実装時に、これらの設定の構成が必要になる場合があります。

これらの構成プロパティーは、sessionManagement カテゴリーに含まれます。

ロケールと Interact API

Interact は、英語以外のタッチポイントに使用することができます。タッチポイントおよび API 内のストリングはすべて、そのランタイム環境のユーザー用に定義されているロケールを使用します。

ロケールは、サーバー・グループごとに 1 つのみ選択可能です。

例えば、ランタイム環境で、ユーザー・ロケールが英語に設定されている asm_admin_en とユーザー・ロケールがフランス語に設定されている asm_admin_fr の 2つのユーザーを作成します。ご使用のタッチポイントがフランス語を話すユーザー用に設計されている場合には、そのランタイム環境の asmUserForDefaultLocale プロパティーを asm_admin_fr として定義します。

JMX 監視について

Interact は、任意の JMX 監視アプリケーションからアクセスできる Java Management Extensions (JMX) 監視サービスを提供しています。この JMX 監視を使用することで、ランタイム・サーバーを監視および管理できます。JMX 属性は、ランタイム・サーバーに関する多数の詳細情報を提供します。例えば、JMX 属性 ErrorCount は、前回のリセットまたはシステムの始動以降にログに記録されたエラー・メッセージの数を示します。この情報を使用して、そのシステムでエラーが発生している頻度を知ることができます。誰かがトランザクションを完了した場合に終了セッションのみを呼び出すように Web サイトをコーディングした場合は、startSessionCount と endSessionCount を比較して、未完了のトランザクションの数を知ることもできます。

Interact は、JSR 160 で定義されているように、RMI プロトコルと JMXMP プロトコルをサポートしています。 JSR160 準拠の任意の JMX クライアントを使用して、JMX 監視サービスに接続することができます。

インタラクティブ・フローチャートは、JMX 監視でのみ監視できます。インタラクティブ・フローチャートに関する情報は、Campaign 監視では表示されません。

注: IBM WebSphere® をノード・マネージャーとともに使用している場合は、JMX 監視を有効にするように Generic JVM Argument を定義する必要があります。

RMI プロトコルを使用した JMX 監視を使用するように Interact を構成するには

ランタイム環境の Marketing Platform では、「Interact」>「監視」カテゴリーで以下の構成プロパティーを編集します。

構成プロパティー	設定
protocol	**RMI**
port	JMX サービスのポート番号
enableSecurity	**False** RMI プロトコルを使用する Interact 実装環境では、セキュリティーはサポートされていません。

RMI プロトコルを使用する監視用のデフォルト・アドレスは、service:jmx:rmi:///jndi/rmi://*RuntimeServer*:*port*/interact です。

JMXMP プロトコルを使用した JMX 監視を使用するように Interact を構成するには

JMXMP プロトコルは、クラスパス内に InteractJMX.jar と jmxremote_optional.jar の 2 つの追加ライブラリーがこの順序で含まれていることを必要とします。これらのファイルはどちらも、インストール済みのランタイム環境の lib ディレクトリーにあります。

注: セキュリティーを有効にする場合、ユーザー名とパスワードは、ランタイム環境の Marketing Platform のユーザーと一致している必要があります。空のパスワードは使用できません。

ランタイム環境の Marketing Platform では、「Interact」>「監視」カテゴリーで以下の構成プロパティーを編集します。

構成プロパティー	設定
protocol	**JMXMP**
port	JMX サービスのポート番号
enableSecurity	セキュリティーを無効にする場合は 「**False**」。セキュリティーを有効にする場合は「**True**」。

JMXMP プロトコルの場合、監視用のデフォルト・アドレスは、service:jmx:jmxmp://*RuntimeServer*:*port* です。

jconsole スクリプトの使用

JMX 監視アプリケーションを別に所持していない場合は、JVM とともにインストールされている jconsole を使用することができます。jconsole を開始するには、Interact/tools ディレクトリーにある開始スクリプトを使用します。

1. テキスト・エディターで Interact¥tools¥jconsole.bat (Windows) または Interact/tools/jconsole.sh (Unix) を開きます。
2. INTERACT_LIB を *InteractInstallationDirectory*/lib ディレクトリーへの絶対パスに設定します。
3. HOST を、監視するランタイム・サーバーのホスト名に設定します。
4. PORT を、「Interact」>「監視」>「ポート」 プロパティーで JMX が listen するポートとして構成したポートに設定します。
5. RMI プロトコルを使用して監視する場合は、JMXMP 接続の前にコメントを追加し、RMI 接続の前にそのコメントを削除します。

   デフォルトでは、このスクリプトは JMXMP プロトコルを使用して監視します。

   例えば、jconsole.bat のデフォルト設定を参照してください。

   JMXMP 接続

```
%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 接続

```
%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
```

JMX 属性

以下の表では、JMX 監視で使用可能な属性について説明します。

JMX 監視によって提供されるデータはすべて、前回のリセット以降またはシステムの始動以降のデータです。例えばカウントは、前回のリセットまたはシステムの始動以降の項目数であり、インストールした時点からの項目数ではありません。

表 8. コンタクトとレスポンスの履歴の ETL 監視

属性	説明
AvgCHExecutionTime	コンタクトとレスポンスの履歴モジュールがコンタクトの履歴テーブルへの書き込みに要した平均ミリ秒数。この平均の計算に入れられるのは、成功した操作、および最低 1 つのレコードがコンタクトの履歴テーブルに書き込まれた操作のみです。
AvgETLExecutionTime	コンタクトとレスポンスの履歴モジュールがランタイム環境からのデータの読み取りに要した平均ミリ秒数。この平均には、成功した操作と失敗した操作の時間が含まれます。

表 8. コンタクトとレスポンスの履歴の ETL 監視 (続き)

属性	説明
AvgRHExecutionTime	コンタクトとレスポンスの履歴モジュールがレスポンスの履歴テーブルへの書き込みに要した平均ミリ秒数。この平均の計算に入れられるのは、成功した操作、および最低 1 つのレコードがレスポンスの履歴テーブルに書き込まれた操作のみです。
ErrorCount	前回のリセットまたはシステムの始動以降にログに記録されたエラー・メッセージの数 (もしあれば)。
HighWaterMarkCHExecutionTime	コンタクトとレスポンスの履歴モジュールがコンタクトの履歴テーブルへの書き込みに要した最大ミリ秒数。この値の計算に入れられるのは、成功した操作、および最低 1 つのレコードがコンタクトの履歴テーブルに書き込まれた操作のみです。
HighWaterMarkETLExecutionTime	コンタクトとレスポンスの履歴モジュールがランタイム環境からのデータの読み取りに要した最大ミリ秒数。この計算には、成功した操作と失敗した操作の両方が含まれます。
HighWaterMarkRHExecutionTime	コンタクトとレスポンスの履歴モジュールがレスポンスの履歴テーブルへの書き込みに要した最大ミリ秒数。この値の計算に入れられるのは、成功した操作、および最低 1 つのレコードがレスポンスの履歴テーブルに書き込まれた操作のみです。
LastExecutionDuration	コンタクトとレスポンスの履歴モジュールが最後のコピーの実行に要したミリ秒数。
NumberOfExecutions	初期化以降にコンタクトとレスポンスの履歴モジュールが実行された回数。
LastExecutionStart	最後に実行されたコンタクトとレスポンスの履歴モジュールの開始時刻。
LastExecutionSuccessful	true の場合、最後に実行されたコンタクトとレスポンスの履歴モジュールは成功しました。false の場合は、エラーが発生しました。
NumberOfContactHistoryRecordsMarked	コンタクトとレスポンスの履歴モジュールの現在の実行中に移動される、`UACI_CHStaging` テーブル内にあるコンタクトの履歴レコードの数。この値は、コンタクトとレスポンスの履歴モジュールが現在実行中の場合にのみ、ゼロより大きな値になります。

表 8. コンタクトとレスポンスの履歴の ETL 監視 (続き)

属性	説明
NumberOfResponseHistoryRecordsMarked	コンタクトとレスポンスの履歴モジュールの現在の実行中に移動される、UACI_RHStaging テーブル内にあるレスポンスの履歴レコードの数。この値は、コンタクトとレスポンスの履歴モジュールが現在実行中の場合にのみ、ゼロより大きな値になります。

コンタクトとレスポンスの履歴の ETL 監視属性は、設計環境に組み込まれています。以下の属性はすべて、ランタイム環境に組み込まれています。

表 9. 例外

属性	説明
errorCount	前回のリセットまたはシステムの始動以降にログに記録されたエラー・メッセージの数。
warningCount	前回のリセットまたはシステムの始動以降にログに記録された警告メッセージの数。

表 10. フローチャート・エンジンの統計

属性	説明
activeProcessBoxThreads	現在実行中のフローチャート処理スレッド (すべての実行で共有) のアクティブ数。
activeSchedulerThreads	現在実行中のフローチャート・スケジューラー・スレッドのアクティブ数。
avgExecutionTimeMillis	フローチャートの平均実行時間 (ミリ秒単位)。
CurrentJobsInProcessBoxQueue	フローチャート処理スレッドによる実行を待機しているジョブの数。
CurrentJobsInSchedulerQueue	フローチャート・スケジューラー・スレッドによる実行を待機しているジョブの数。
maximumProcessBoxThreads	実行可能なフローチャート処理スレッド (すべての実行で共有) の最大数。
maximumSchedulerThreads	実行可能なフローチャート・スケジューラー・スレッド (実行ごとに 1 つのスレッド) の最大数。
numExecutionsCompleted	実行が完了したフローチャートの総数。
numExecutionsStarted	実行を開始したフローチャートの総数。

表 11. インタラクティブ・チャネルごとに固有のフローチャート

属性	説明
AvgExecutionTimeMillis	このインタラクティブ・チャネル内のフローチャートの平均実行時間 (ミリ秒単位)。
HighWaterMarkForExecutionTime	このインタラクティブ・チャネル内のフローチャートの最大実行時間 (ミリ秒単位)。
LastCompletedExecutionTimeMillis	このインタラクティブ・チャネル内で最後に完了したフローチャートの実行時間 (ミリ秒単位)。
NumExecutionsCompleted	このインタラクティブ・チャネル内で実行が完了したフローチャートの総数。
NumExecutionsStarted	このインタラクティブ・チャネル内で実行が開始されたフローチャートの総数。

表 12. ロケール

属性	説明
ロケール	JMX クライアントのロケール設定。

表 13. ロガー構成

属性	説明
カテゴリー	そのログ・レベルの操作が可能なログ・カテゴリーを変更します。

表 14. サービス・スレッド・プールの統計

属性	説明
activeContactHistThreads	コンタクト履歴とレスポンス履歴用のタスクをアクティブに実行しているスレッドのおおよその数。
activeFlushCacheToDBThreads	データ・ストアにキャッシュされた統計をフラッシュするタスクをアクティブに実行しているスレッドのおおよその数。
activeOtherStatsThreads	対象となる STAT、イベント・アクティビティー、およびデフォルトの STAT のタスクをアクティブに実行しているスレッドのおおよその数。
CurrentHighWaterMarkInContactHistQueue	コンタクトとレスポンスの履歴データを収集するサービスによってログに記録させるためのキューに入る項目の最大数。

表 14. サービス・スレッド・プールの統計 (続き)

属性	説明
CurrentHighWaterMark InFlushCachetoDBQueue	キャッシュ内のデータをデータベース・テーブルに書き込むサービスによってログに記録させるためのキューに入る項目の最大数。
CurrentHighWaterMarkInOtherStatsQueue	オファーの資格統計、デフォルト・ストリングの使用統計、イベント・アクティビティー統計、およびカスタム・ログを収集してテーブル・データにするサービスによってログに記録させるためのキューに入る項目の最大数。
currentMsgsInContactHistQueue	コンタクト履歴およびレスポンス履歴に使用されるスレッド・プール用のキューに含まれるジョブ数。
currentMsgsInFlushCacheToDBQueue	キャッシュに入れられたデータ・ストアに送られる統計のフラッシュに使用されるスレッド・プール用のキューに含まれるジョブ数。
currentMsgsInOtherStatsQueue	対象となる STAT、イベント・アクティビティー、およびデフォルトの STAT に使用されるスレッド・プール用のキューに含まれるジョブ数。
maximumContactHistThreads	コンタクト履歴およびレスポンス履歴に使用されるプールに同時に含まれたことがあるスレッドの最大数。
maximumFlushCacheToDBThreads	キャッシュに入れられたデータ・ストアに送られる統計のフラッシュに使用されるプールに同時に含まれたことがあるスレッドの最大数。
maximumOtherStatsThreads	対象となる STAT、イベント・アクティビティー、およびデフォルトの STAT に使用されるプールに同時に含まれたことがあるスレッドの最大数。

サービスの統計は、各サービスの属性のセットで構成されます。

- ContactHistoryMemoryCacheStatistics — コンタクト履歴ステージング・テーブル用のデータを収集するサービス。
- CustomLoggerStatistics — テーブルに書き込むカスタム・データを収集するサービス (`UACICustomLoggerTableName` イベント・パラメーターを使用するイベント)
- デフォルトの統計 — 対話点のデフォルト・ストリングが使用された回数に関する統計を収集するサービス。
- 資格統計 — 対象となるオファーの統計を書き込むサービス。
- イベント・アクティビティーの統計 — イベントの統計 (`getOffer` や `startSession` などのシステム・イベントと `postEvent` によってトリガーされるユーザー・イベントの両方) を収集するサービス。

- レスポンス履歴のメモリー・キャッシュ統計 — レスポンス履歴ステージング・テーブルに書き込むサービス。
- クロス・セッション・レスポンス統計 — クロス・セッション・レスポンス・トラッキングのデータを収集するサービス。

表 15. サービス統計

属性	説明
件数	処理されたメッセージの数。
ExecTimeInsideMutex	このサービスのメッセージの処理に費やされた時間 (ミリ秒単位)。ただし、他のスレッドを待機していた時間は除外されます。ExecTimeInsidMutex と ExecTimeMillis の間に大きな差がある場合、そのサービスのスレッド・プール・サイズの変更が必要になる可能性があります。
ExecTimeMillis	このサービスのメッセージの処理に費やされた時間 (ミリ秒単位)。他のスレッドを待機していた時間も含まれます。
ExecTimeOfDBInsertOnly	バッチ挿入部分のみの処理に費やされた時間 (ミリ秒単位)。
HighWaterMark	このサービスについて処理されたメッセージの最大数。
NumberOfDBInserts	実行されたバッチ挿入の総数。
TotalRowsInserted	データベースに挿入された行の総数。

表 16. サービス統計 - データベース・ロード・ユーティリティー

属性	説明
ExecTimeOfWriteToCache	ファイル・キャッシュへの書き込みに費やされた時間 (ミリ秒単位)。必要に応じて、ファイルへの書き込みや、データベースからのプライマリー・キーの取得も含まれます。
ExecTimeOfLoaderDBAccessOnly	データベース・ローダーの実行部分のみの処理に費やされた時間 (ミリ秒単位)。
ExecTimeOfLoaderThreads	データベース・ローダーのスレッドによって費やされた時間 (ミリ秒単位)。
ExecTimeOfFlushCacheFiles	キャッシュのフラッシュと新規の再作成に費やされた時間 (ミリ秒単位)。
ExecTimeOfRetrievePKDBAccess	プライマリー・キーを取得するためのデータベース・アクセスに費やされた時間 (ミリ秒単位)。
NumberOfDBLoaderRuns	データベース・ローダーの実行総数。
NumberOfLoaderStagingDirCreated	作成されたステージング・ディレクトリーの総数。

表 16. サービス統計 - データベース・ロード・ユーティリティー (続き)

属性	説明
NumberOfLoaderStagingDirRemoved	削除されたステージング・ディレクトリーの総数。
NumberOfLoaderStagingDirMovedToAttention	重要に指定変更されたステージング・ディレクトリーの総数。
NumberOfLoaderStagingDirMovedToError	エラーに指定変更されたステージング・ディレクトリーの総数。
NumberOfLoaderStagingDirRecovered	リカバリーされたステージング・ディレクトリーの総数 (起動時やバックグラウンド・スレッドによる再実行も含まれます)。
NumberOfTimesRetrievePKFromDB	データベースからプライマリー・キーを取得した合計回数。
NumberOfLoaderThreadsRuns	データベース・ローダー・スレッドの実行総数。
NumberOfFlushCacheFiles	ファイル・キャッシュをフラッシュした合計回数。

表 17. API 統計

属性	説明
endSessionCount	前回のリセットまたはシステムの始動以降の `endSession` API 呼び出しの数。
endSessionDuration	前回の `endSession` API 呼び出しの経過時間。
executeBatchCount	前回のリセットまたはシステムの始動以降の `executeBatch` API 呼び出しの数。
executeBatchDuration	前回の `executeBatch` API 呼び出しの経過時間。
getOffersCount	前回のリセットまたはシステムの始動以降の `getOffers` API 呼び出しの数。
getOffersDuration	前回の `getOffer` API 呼び出しの経過時間。
getProfileCount	前回のリセットまたはシステムの始動以降の `getProfile` API 呼び出しの数。
getProfileDuration	前回の `getProfileDuration` API 呼び出しの経過時間。
getVersionCount	前回のリセットまたはシステムの始動以降の `getVersion` API 呼び出しの数。
getVersionDuration	前回の `getVersion` API 呼び出しの経過時間。

表 17. API 統計 (続き)

属性	説明
loadOfferSuppressionDuration	前回の loadOfferSuppression API 呼び出しの経過時間。
LoadOffersBySQLCount	前回のリセットまたはシステムの始動以降の LoadOffersBySQL API 呼び出しの数。
LoadOffersBySQLDuration	前回の LoadOffersBySQL API 呼び出しの経過時間。
loadProfileDuration	前回の loadProfile API 呼び出しの経過時間。
loadScoreOverrideDuration	前回の loadScoreOverride API 呼び出しの経過時間。
postEventCount	前回のリセットまたはシステムの始動以降の postEvent API 呼び出しの数。
postEventDuration	前回の postEvent API 呼び出しの経過時間。
runSegmentationDuration	前回の runSegmentation API 呼び出しの経過時間。
setAudienceCount	前回のリセットまたはシステムの始動以降の setAudience API 呼び出しの数。
setAudienceDuration	前回の setAudience API 呼び出しの経過時間。
setDebugCount	前回のリセットまたはシステムの始動以降の setDebug API 呼び出しの数。
setDebugDuration	前回の setDebug API 呼び出しの経過時間。
startSessionCount	前回のリセットまたはシステムの始動以降の startSession API 呼び出しの数。
startSessionDuration	前回の startSession API 呼び出しの経過時間。

表 18. Learning Optimizer の統計

属性	説明
LearningOptimizerAcceptCalls	学習モジュールに渡された承認イベントの数。
LearningOptimizerAcceptTrackingDuration	学習モジュール内の承認イベントのロギングに費やされた累計時間 (ミリ秒単位)。
LearningOptimizerContactCalls	学習モジュールに渡されたコンタクト・イベントの数。
LearningOptimizerContactTrackingDuration	学習モジュール内のコンタクト・イベントのロギングに費やされた累計時間 (ミリ秒単位)。

表 18. Learning Optimizer の統計 (続き)

属性	説明
LearningOptimizerLogOtherCalls	学習モジュールに渡された、コンタクトでもなく承認でもないイベントの数。
LearningOptimizerLogOtherTrackingDuration	学習モジュール内のその他の (コンタクトでもなく承認でもない) イベントのロギングに費やされた所要時間 (ミリ秒単位)。
LearningOptimizerNonRandomCalls	構成済みの学習の実装が適用された回数。
LearningOptimizerRandomCalls	構成済みの学習の実装がバイパスされ、ランダムな選択が適用された回数。
LearningOptimizerRecommendCalls	学習モジュールに渡された推奨要求の数。
LearningOptimizerRecommendDuration	推奨ロジックの学習に費やされた累計時間 (ミリ秒単位)。

表 19. デフォルトのオファーの統計

属性	説明
LoadDefaultOffersDuration	デフォルトのオファーのロードにかかっている経過時間。
DefaultOffersCalls	デフォルトのオファーがロードされた回数。

JMX 操作

次の表では、JMX 監視で使用可能な操作について説明しています。

グループ	属性	説明
ロガー構成	activateDebug	`Interact/conf/interact_log4j.properties` で定義されているログ・ファイルのログ・レベルをデバッグに設定します。
ロガー構成	activateError	`Interact/conf/interact_log4j.properties` で定義されているログ・ファイルのログ・レベルをエラーに設定します。
ロガー構成	activateFatal	`Interact/conf/interact_log4j.properties` で定義されているログ・ファイルのログ・レベルを重大に設定します。
ロガー構成	activateInfo	`Interact/conf/interact_log4j.properties` で定義されているログ・ファイルのログ・レベルを情報に設定します。

グループ	属性	説明
ロガー構成	activateTrace	Interact/conf/interact_log4j.properties で定義されているログ・ファイルのログ・レベルをトレースに設定します。
ロガー構成	activateWarn	Interact/conf/interact_log4j.properties で定義されているログ・ファイルのログ・レベルを警告に設定します。
ロケール	changeLocale	JMX クライアントのロケールを変更します。Interact でサポートされているロケールは、de、en、es、および fr です。
ContactResponseHistory ETLMonitor	リセット	すべてのカウンターをリセットします。
デフォルトのオファーの統計	updatePollPeriod	defaultOfferUpdatePollPeriod を更新します。この値 (秒単位) は、キャッシュ内のデフォルトのオファーを更新する前に、システムが待機する時間の長さを指定します。-1 に設定した場合、システムが始動時に行うのは、デフォルト・オファーの数の読み取りのみです。

第 7 章 IBM Unica Interact API のクラスとメソッド

以下のセクションでは、Interact API を使用して作業を行う前に知っておく必要がある要件などの詳細をリストしています。

注: このセクションでは、ユーザーがご使用のタッチポイント、Java プログラミング言語、および Java ベースの API を使用した作業に精通していることを前提としています。

Interact API には、HTTP で Java 直列化を使用する Java クライアント・アダプターがあります。さらに、Interact は、SOAP クライアントをサポートするために、WSDL を提供しています。WSDL は Java クライアント・アダプターと同じ機能のセットを公開しているため、以下のセクションはこれにも適用されます (例は除きます)。

Interact API クラス

Interact API は、`InteractAPI` クラスに基づきます。サポートしているインターフェースは 6 つあります。

- `AdvisoryMessage`
- `BatchResponse`
- `NameValuePair`
- `Offer`
- `OfferList`
- `Response`

これらのインターフェースは 3 つの具象クラスをサポートしています。以下の 2 つの具象クラスをインスタンス化して、Interact API メソッドに引数として渡す必要があります。

- NameValuePairImpl
- CommandImpl

`AdvisoryMessageCode` という 3 つ目の具象クラスは、サーバーから返されたメッセージ・コードがある場合に、その識別に使用される定数を提供するために使用できます。

このセクションの残りの部分では、Interact API を構成するメソッドについて説明します。

HTTP における Java 直列化の前提条件

1. Java 直列化アダプターを使用した作業を開始する前に、ご使用の `CLASSPATH` に次のファイルを追加しておく必要があります。

   ```
   Interact_Runtime_Environment_Installation_Directory/lib/interact_client.jar
   ```

2. クライアントとサーバーの間でやり取りされるオブジェクトは、すべて com.unicacorp.interact.api パッケージ内にあります。詳しくは、Interact API JavaDoc を参照してください。
3. InteractAPI クラスのインスタンスを取得するには、Interact ランタイム・サーバーの URL を持つ静的メソッド getInstance を呼び出します。

SOAP の前提条件

重要: パフォーマンス・テストによって、Java 直列化アダプターは、生成される SOAP クライアントよりもかなり高いレートで実行されることが示されています。最良のパフォーマンスを得るためには、可能な限り Java 直列化アダプターを使用してください。

SOAP を使用してランタイム・サーバーにアクセスするには、以下を行う必要があります。

1. 任意の SOAP ツールキットを使用して Interact API WSDL を変換します。

   Interact API WSDL は、Interact のインストール時に Interact/conf ディレクトリーにインストールされています。

   WSDL のテキストは、本書の最後にあります。
2. ランタイム・サーバーをインストールし、構成します。

   統合のテストを完全に行うには、ランタイム・サーバーが稼働している必要があります。

SOAP のバージョン

Interact は、Interact ランタイム・サーバー上の SOAP インフラストラクチャーとして、axis2 1.3 を使用します。SOAP axis2 1.3 のどのバージョンをサポートするかについて詳しくは、次の Web サイトを参照してください。

Apache Axis2

Interact は、axis2、XFire、JAX-WS-Ri、DotNet、SOAPUI、および IBM RAD SOAP の各クライアントでテスト済みです。

API JavaDoc

本書に加えて、Interact API の JavaDoc が、ランタイム・サーバーとともにインストールされます。JavaDoc は、参照用として Interact/docs/apiJavaDoc ディレクトリーにインストールされます。

API の例について

本書に含まれている例はすべて、HTTP アダプターを介した Java 直列化を使用して作成されています。 SOAP を使用している場合、WSDL からのクラス生成は SOAP ツールキットや選択したオプションによって異なる場合があるため、これらの例がご使用の環境で完全に同じ動作をしない可能性があります。

セッション・データを使用した作業

startSession メソッドを使用してセッションを開始すると、セッション・データがメモリーにロードされます。そのセッション全体を通して、そのセッション・データ (静的プロファイル・データのスーパーセット) の読み取りと書き込みを行うことができます。セッションには以下のデータが含まれます。

- 静的プロファイル・データ
- セグメントの割り当て
- リアルタイム・データ
- オファー推奨

セッション・データはすべて、endSession メソッドを呼び出すか、sessionTimeout の時間が経過するまで使用可能です。セッションが終了すると、コンタクトまたはレスポンスの履歴やその他のデータベース・テーブルに明示的に保存されていないデータは、すべて失われます。

データは、名前と値のペアのセットとして保管されます。データがデータベース・テーブルから読み取られる場合、名前はそのテーブルの列です。

これらの名前と値のペアは、Interact API を使用した作業を行う中で作成できます。すべての名前と値のペアをグローバル域で宣言する必要はありません。新しいイベント・パラメーターを名前と値のペアとして設定すると、ランタイム環境ではその名前と値のペアがセッション・データに追加されます。例えば、postEvent メソッドでイベント・パラメーターを使用すると、プロファイル・データでそのイベント・パラメーターを使用できなかった場合でも、ランタイム環境ではそのイベント・パラメーターがセッション・データに追加されます。このデータは、セッション・データ内にのみ存在します。

セッション・データはいつでも上書きすることができます。例えば、顧客プロファイルの一部に creditScore が含まれる場合には、カスタム・タイプ NameValuePair を使用してイベント・パラメーターに渡すことができます。NameValuePair クラスでは、setName メソッドおよび setValueAsNumeric メソッドを使用して、値を変更することができます。名前は一致している必要があります。セッション・データ内の名前は、大文字小文字を区別されません。つまり、名前 creditscore または CrEdItScOrE は、どちらも creditScore を上書きします。

そのセッション・データに最後に書き込まれたデータのみが保持されます。例えば、startSession は、lastOffer の値のプロファイル・データをロードします。postEvent メソッドは lastOffer を上書きします。その後、2 番目の postEvent メソッドが lastOffer を上書きします。ランタイム環境では、2 番目の postEvent メソッドによってセッション・データ内に書き込まれたデータのみが保持されます。

セッションが終了すると、データは失われます。ただし、インタラクティブ・フローチャート内でスナップショット・プロセスを使用してデータベース・テーブルにデータを書き込むなど、特別に考慮すべきことを行った場合は除きます。スナップショット・プロセスの使用を計画している場合、名前はご使用のデータベースの制

限を満たしている必要がありますので注意してください。例えば、列の名前に 256 文字までしか使用できない場合には、その名前と値のペアの名前も 256 文字を超えないようにする必要があります。

InteractAPI クラスについて

InteractAPI クラスには、タッチポイントとランタイム・サーバーの統合に使用するメソッドが含まれています。Interact API 内の他のすべてのクラスおよびメソッドは、このクラス内のメソッドをサポートしています。

Interact ランタイムのインストール済み環境の lib ディレクトリーにある interact_client.jar に対して、実装をコンパイルする必要があります。

endSession

endSession(String *sessionID*)

endSession メソッドは、ランタイム・セッション終了のマークを付けます。ランタイム・サーバーは、このメソッドを受信すると、履歴へのログの記録やメモリーのクリアなどを行います。

- **sessionID** — セッションを識別する一意の文字列。

endSession メソッドが呼び出されない場合、ランタイム・セッションはタイムアウトになります。タイムアウト期間は、sessionTimeout プロパティーを使用して構成可能です。

戻り値

ランタイム・サーバーは、以下の属性が設定された Response オブジェクトを使用して endSession メソッドに応答します。

- SessionID
- ApiVersion
- StatusCode
- AdvisoryMessages

例

以下の例は、endSession メソッドと、応答を解析する方法を示します。 sessionId は、このセッションを開始した startSession 呼び出しで使用されるセッションを識別する同じ文字列です。

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

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

executeBatch

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

executeBatch メソッドを使用して、ランタイム・サーバーへの 1 つの要求で、複数のメソッドを実行できます。

- **sessionID** — セッション ID を識別する文字列。このセッション ID は、このメソッド呼び出しによって実行されるすべてのコマンドに使用されます。
- **commandImpl[]** — CommandImpl オブジェクトの配列 (実行するコマンドごとに 1 つずつ)。

このメソッドの呼び出しの結果は、Command 配列内の各メソッドを明示的に呼び出す場合と同じです。このメソッドは、ランタイム・サーバーへの実際の要求の数を最小限に抑えます。ランタイム・サーバーは、各メソッドを連続して実行します。各呼び出しに対するエラーや警告は、そのメソッド呼び出しに対応するレスポンス・オブジェクトで取得されます。エラーが発生した場合、executeBatch はバッチの残りの呼び出しを続行します。メソッドの実行結果がエラーになった場合、BatchResponse オブジェクトの最上位のステータスがそのエラーを示します。エラーがない場合、警告が出ている可能性があれば、最上位のステータスがそれを示します。警告がない場合、最上位のステータスがバッチ実行の成功を示します。

戻り値

ランタイム・サーバーは、BatchResponse オブジェクトを使用して、executeBatch に応答します。

例

以下の例は、1 つの executeBatch 呼び出しで getOffer と postEvent のすべてのメソッドを呼び出す方法と、応答の処理に関する推奨方法を示します。

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

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

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

/** Build command array */
Command[] commands =
{
```

```
  getOffersCommand,
  postEventCommand,
};

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

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

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

getInstance

```
getInstance(String URL)
```

getInstance メソッドは、指定されたランタイム・サーバーと通信する Interact API のインスタンスを作成します。

重要: Interact API を使用して作成するすべてのアプリケーションで、URL パラメーターによって指定されたランタイム・サーバーにマップする InteractAPI オブジェクトをインスタンス化するには、getInstance を呼び出す必要があります。

サーバー・グループに対して、負荷分散装置を使用している場合は、負荷分散装置で構成するホスト名とポートを使用します。負荷分散装置を使用しない場合は、使用可能なランタイム・サーバー間を循環させるためのロジックを組み込む必要があります。

このメソッドは、HTTP アダプターを介す Java シリアライゼーションにのみ適用されます。 SOAP WSDL に定義された対応するメソッドはありません。各 SOAP クライアントの実装には、エンドポイント URL を確立する独自の方法があります。

- **URL** — ランタイム・インスタンスの URL を識別する文字列。例: http://localhost:7001/Interact/servlet/InteractJSService

戻り値

ランタイム・サーバーは InteractAPI を返します。

例

以下の例は、タッチポイントと同じマシン上で実行されるランタイム・サーバー・インスタンスを指す InteractAPI オブジェクトをインスタンス化する方法を示します。

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

getOffers

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

getOffers メソッドを使用して、ランタイム・サーバーからのオファーを要求できます。

- **sessionID** — 現在のセッションを識別する文字列。
- **interactionPoint** — このメソッドが参照するインタラクション・ポイントの名前を識別する文字列。

  注: この名前は、インタラクティブ・チャネルで定義されているインタラクション・ポイントの名前と正確に一致する必要があります。
- **numberOfOffers** — 要求されるオファーの数を識別する整数。

getOffers メソッドは、segmentationMaxWaitTimeInMS プロパティーに定義された時間 (ミリ秒単位) 待機し、すべての再セグメント化が完了してから実行されます。したがって、getOffers 呼び出しの直前に、再セグメント化または setAudience メソッドをトリガーする postEvent メソッドを呼び出す場合は、遅延が生じる可能性があります。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して getOffers に応答します。

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

例

この例は、Overview Page Banner 1 インタラクション・ポイントに対する単一オファーの要求と、その応答を処理する方法を示します。

sessionId は、このセッションを開始した startSession 呼び出しで使用されるランタイム・セッションを識別する同じ文字列です。

```
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());
```

getOffersForMultipleInteractionPoints

`getOffersForMultipleInteractionPoints(String sessionID, String requestStr)`

`getOffersForMultipleInteractionPoints` メソッドを使用して、重複が解消されている複数の IP に対する、ランタイム・サーバーからのオファーを要求できます。

- **sessionID** — 現在のセッションを識別する文字列。
- **requestStr** — `GetOfferRequest` オブジェクトの配列を指定する文字列。

  `GetOfferRequest` オブジェクトはそれぞれ以下を指定します。

  - **ipName** — オファーを要求しているオブジェクトのインタラクション・ポイント (IP) 名
  - **numberRequested** — 指定された IP に必要な一意のオファーの数
  - **offerAttributes** — `OfferAttributeRequirements` のインスタンスを使用する、配信されるオファーの属性についての要件。
  - **duplicationPolicy** — 配信されるオファーの重複ポリシー ID

    単一のメソッド呼び出しにおいて、重複するオファーが異なるインタラクション・ポイントで返されるかどうかは、重複ポリシーによって決まります。 (個々のインタラクション・ポイント内 で重複するオファーが返されることはありません)。現在は、2 つの重複ポリシーがサポートされています。

    - NO_DUPLICATION (ID 値 = 1)。この `GetOfferRequest` インスタンスには、先行する `GetOfferRequest` インスタンスに含まれているオファーは含みません (つまり、Interact により、重複解除が適用されます)。

- ALLOW_DUPLICATION (ID 値 = 2)。この GetOfferRequest インスタンスで指定されている要件を満たすオファーがあれば含めます。先行する GetOfferRequest インスタンスに含まれているオファーは調整されません。

配列パラメーターの要求の順番は、オファーの配信時の優先順位でもあります。

例えば、要求の IP が IP1、IP2 の順で、重複するオファーは許可されず (重複ポリシー ID = 1)、それぞれが 2 つずつオファーを要求しているとします。Interact が IP1 のオファー A、B、C と、IP2 のオファー A、D を検出した場合、その応答には IP1 のオファー A、B と、IP2 のオファー D のみが含まれます。

また、重複ポリシー ID が 1 の場合、優先順位がより高い IP を介して配信されているオファーは、この IP を介して配信されません。

getOffersForMultipleInteractionPoints メソッドは、segmentationMaxWaitTimeInMS プロパティーに定義された時間 (ミリ秒単位) 待機し、すべての再セグメント化が完了してから実行されます。したがって、getOffers 呼び出しの直前に、再セグメント化または setAudience メソッドをトリガーする postEvent メソッドを呼び出す場合は、遅延が生じる可能性があります。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して getOffersForMultipleInteractionPoints に応答します。

- AdvisoryMessages
- ApiVersion
- OfferList の配列
- SessionID
- StatusCode

例

```
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());
}
```

requestStr の構文は以下のようになることに注意してください。

```
requests_for_IP[<requests_for_IP]
```

ここで

```
<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
```

上記の例の requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))}) は、IP1 というインタラクション・ポイントに対して、多くても 5 つの明確に異なるオファー (この同じメソッド呼び出しの間に他のインタラクション・ポイントに対して返すこともできないオファー) を配信することを意味します。この 5 つのオファーはすべて、attr1 という数値属性を持ち、その値は 1 である必要があり、さらに attr2 という文字列属性を持ち、その値は *value2* である必要があります。その 5 つのうち、最大 3 つが attr3 という文字列属性を持ち、その値は *value3* である必要があり、さらに最大 3 つが attr4 という数値属性を待ち、その値は *4* である必要があります。

使用できる属性タイプは、数値、文字列、および日時です。日時属性値の形式は MM/dd/yyyy HH:mm:ss である必要があります。返されるオファーを取得するには、メソッド Response.getAllOfferLists() を使用します。構文の理解を助けるため、setGetOfferRequests の例では、Java オブジェクトを使用しながら GetOfferRequests の同じインスタンスを作成します。この方法が推奨されます。

getProfile

```
getProfile(String sessionID)
```

getProfile メソッドを使用して、タッチポイントを訪れる訪問者に関するプロファイルと一時的な情報を取得できます。

- **sessionID** — セッション ID を識別する文字列。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して getProfile に応答します。

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

例

以下に、getProfile の使用例と、応答の処理方法を示します。

sessionId は、このセッションを開始した startSession 呼び出しで使用されるセッションを識別する同じ文字列です。

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

getVersion

```
getVersion()
```

getVersion メソッドは、Interact ランタイム・サーバーの現在の実装のバージョンを返します。

ベスト・プラクティスは、Interact API を使用してタッチポイントを初期化するときに、この方法を使用することです。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して getVersion に応答します。

- AdvisoryMessages
- ApiVersion
- StatusCode

例

この例では、getVersion を呼び出し、結果を処理する簡単な方法を示します。

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

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

postEvent

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

postEvent メソッドを使用して、インタラクティブ・チャネルで定義されているイベントがあれば実行できます。

- **sessionID** — セッション ID を識別する文字列。
- **eventName** — イベントの名前を識別する文字列。

  注: イベントの名前は、インタラクティブ・チャネルで定義されているイベントの名前と一致する必要があります。この名前の大/小文字は区別されません。
- **eventParameters** — イベントを使用して渡す必要のあるパラメーターを識別する NameValuePairImpl オブジェクト。これらの値はセッション・データに格納されます。

  このイベントが再セグメント化をトリガーする場合、インタラクティブ・フローチャートで要求されるすべてのデータをセッション・データで使用できるようにする必要があります。これらの値に、前のアクション (例えば、startSession や setAudience、あるいはプロファイル・テーブルのロードなど) によってデータが設定されていないものがある場合、不足しているそれぞれの値のための eventParameter を含める必要があります。例えば、すべてのプロファイル・テーブルをメモリーにロードするように構成した場合は、インタラクティブ・フローチャートに必要な一時データの NameValuePair を含める必要があります。

  2 つ以上のオーディエンス・レベルを使用していれば、ほとんどの場合、オーディエンス・レベルごとに異なる eventParameters のセットを持ちます。オーディエンス・レベルに正しいパラメーターのセットを確実に選択するために何らかのロジックを含める必要があります。

重要: このイベントがレスポンス履歴への記録を行う場合は、オファーの処理コードを渡す必要があります。 NameValuePair の名前を "UACIOfferTrackingCode" として定義する必要があります。

イベントごとに処理コードを 1 つのみ渡すことができます。オファー・コンタクトの処理コードを渡さない場合、Interact は、オファーの最後の推奨リストにあるすべてのオファーについて、オファー・コンタクトを記録します。応答の処理コードを渡さない場合、Interact はエラーを返します。

- postEvent で使用されるその他の予約パラメーターとその他のメソッドがいくつかあり、これらについては、このセクションの後半で説明します。

再セグメント化や、コンタクトまたはレスポンスの履歴への書き込みの要求は、応答を待機しません。

UACIExecuteFlowchartByName パラメーターによって指定されない限り、再セグメント化では、現在のオーディエンス・レベルのこのインタラクティブ・チャネルに関連付けられているすべてのインタラクティブ・フローチャートを実行します。getOffers メソッドは、実行する前に、再セグメント化が完了するまで待機します。したがって、getOffers 呼び出しの直前に再セグメント化をトリガーする postEvent メソッドを呼び出す場合、遅延が生じる可能性があります。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して postEvent に応答します。

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

例

以下の postEvent の例では、再セグメント化をトリガーするイベントの新規パラメーターの送信と、その応答の処理方法を示します。

sessionId は、このセッションを開始した startSession 呼び出しで使用されるセッションを識別する同じ文字列です。

```
String eventName = "SearchExecution";

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

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

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

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

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

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

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

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

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

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

setAudience

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

setAudience メソッドを使用して、訪問者のオーディエンス ID とレベルを設定できます。

- **sessionID** — セッション ID を識別する文字列。
- **audienceID** — オーディエンス ID を定義する NameValuePairImpl オブジェクトの配列。
- **audienceLevel** — オーディエンス・レベルを定義する文字列。
- **parameters** — setAudience を使用して渡す必要のあるパラメーターを識別する NameValuePairImpl オブジェクト。これらの値はセッション・データに格納され、セグメント化に使用できます。

プロファイルのすべての列に値が必要です。これは、インタラクティブ・チャネルおよびリアルタイム・データ用に定義されたすべてのテーブルのすべての列のスーパーセットです。 startSession または postEvent を使用してすべてのセッション・データを既に追加済みの場合は、新しいパラメーターを送信する必要はありません。

setAudience メソッドは、再セグメント化をトリガーします。 getOffers メソッドは、実行する前に、再セグメント化が完了するまで待機します。したがって、getOffers 呼び出しの直前に setAudience メソッドを呼び出す場合は、遅延が生じる可能性があります。

setAudience メソッドは、オーディエンス ID のプロファイル・データもロードします。 setAudience メソッドを使用して、startSession メソッドでロードされる同じプロファイル・データを強制的に再ロードすることができます。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して setAudience に応答します。

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

例

この例の場合、オーディエンス・レベルは同じままですが、匿名ユーザーがログインし、既知のユーザーになる場合のように、ID が変わります。

sessionId および audienceLevel は、このセッションを開始した startSession 呼び出しで使用するセッションおよびオーディエンス・レベルを識別する同じ文字列です。

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

NameValuePair[] newAudienceId = { custId2 };

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

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

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

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

setDebug

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

setDebug メソッドを使用して、セッションのすべてのコード・パスのロギング冗長レベルを設定できます。

- **sessionID** — セッション ID を識別する文字列。
- **debug** — デバッグ情報を有効または無効にするブール。有効な値は true または false です。 true の場合、Interact はランタイム・サーバー・ログにデバッグ情報を記録します。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して setDebug に応答します。

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

例

以下の例は、セッションのデバッグ・レベルの変更を示します。

sessionId は、このセッションを開始した startSession 呼び出しで使用されるセッションを識別する同じ文字列です。

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

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

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

startSession

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

startSession メソッドは、ランタイム・セッションを作成および定義します。startSession は、最大 5 つまで以下のアクションをトリガーできます。

- ランタイム・セッションを作成します。
- 現在のオーディエンス・レベルの訪問者のプロファイル・データを、インタラクティブ・チャネル用に定義されたテーブル・マッピングでロード用にマーク付けされたディメンション・テーブルを含め、ランタイム・セッションにロードします。
- セグメント化をトリガーし、現在のオーディエンス・レベルのすべてのインタラクティブ・フローチャートを実行します。
- enableOfferSuppressionLookup プロパティーが true に設定されている場合、オファー非表示データをセッションにロードします。
- enableScoreOverrideLookup プロパティーが true に設定されている場合、スコア・オーバーライド・データをセッションにロードします。

startSession メソッドには以下のパラメーターが必要です。

- **sessionID** — セッション ID を識別する文字列。セッション ID を定義する必要があります。例えば、カスタマー ID およびタイムスタンプの組み合わせを使用できます。

  ランタイム・セッションの作成元を定義するには、セッション ID を指定する必要があります。この値は、クライアントによって管理されます。同じセッション ID のすべてのメソッド呼び出しは、クライアントによって同期される必要があります。同じセッション ID で同時に API を呼び出した場合の動作は定義されていません。

- **relyOnExistingSession** — このセッションで新規または既存のセッションを使用するかどうかを定義するブール。有効な値は true または false です。 true の場合、startSession メソッドを使用して既存のセッション ID を指定する必要があります。 false の場合、新規セッション ID を指定する必要があります。

  relyOnExistingSession を true に設定し、セッションが存在する場合、ランタイム環境では、既存のセッション・データを使用します。データの再ロードやセグメント化のトリガーは行われません。セッションが存在しない場合、ランタイム環境では、データのロードやセグメント化のトリガーなどの新規セッションが作成されます。タッチポイントにランタイム・セッションよりも長いセッションがある場合は、relyOnExistingSession を true に設定し、それをすべての startSession 呼び出しで使用すると便利です。例えば、Web サイト・セッションは 2 時間有効ですが、ランタイム・セッションは 20 分しか有効ではありません。

同じセッション ID で startSession を 2 回呼び出す場合、relyOnExistingSession が false であれば、最初の startSession 呼び出しのすべてのセッション・データは失われます。

- **debug** — デバッグ情報を有効または無効にするブール。有効な値は true または false です。 true の場合、Interact はランタイム・サーバー・ログにデバッグ情報を記録します。各セッションに対して個々にデバッグ・フラグが設定されます。このため、個々のセッションのデバッグ・データをトレースできます。
- **interactiveChannel** — このセッションが参照するインタラクティブ・チャネルの名前を定義する文字列。この名前は、Campaign で定義されているインタラクティブ・チャネルの名前と正確に一致する必要があります。
- **audienceID** — NameValuePairImpl オブジェクトの配列。その名前は、オーディエンス ID を含むテーブルの物理的な列名と一致する必要があります。
- **audienceLevel** — オーディエンス・レベルを定義する文字列。
- **parameters** — startSession で渡す必要のあるパラメーターを識別する NameValuePairImpl オブジェクト。これらの値はセッション・データに格納され、セグメント化に使用できます。

  同じオーディエンス・レベルのインタラクティブ・フローチャートが複数ある場合は、すべてのテーブルのすべての列のスーパーセットを含める必要があります。プロファイル・テーブルをロードするようにランタイムを構成する場合、プロファイル・テーブルに必要なすべての列が含まれているときは、プロファイル・テーブルのデータを上書きする必要がある場合を除いて、パラメーターを渡す必要はありません。プロファイル・テーブルに必要な列のサブセットが含まれている場合は、不足している列をパラメーターとして含める必要があります。

audienceID または audienceLevel が無効で、relyOnExistingSession が false の場合、startSession の呼び出しに失敗する可能性があります。interactiveChannel が無効な場合、relyOnExistingSession が true であるか、false であるかにかかわらず、startSession は失敗します。

relyOnExistingSession が true で、同じ sessionID を使用して 2 回目の startSession 呼び出しを行っても、最初のセッションが期限切れになっている場合、Interact は新規セッションを作成します。

relyOnExistingSession が true で、2 回目の startSession 呼び出しで使用する sessionID は同じでも audienceID または audienceLevel が異なる場合、ランタイム・サーバーは既存のセッションのオーディエンスを変更します。

relyOnExistingSession が true で、2 回目の startSession 呼び出しで使用する sessionID は同じでも interactiveChannel が異なる場合、ランタイム・サーバーは新規セッションを作成します。

戻り値

ランタイム・サーバーは、以下の属性が設定されたレスポンス・オブジェクトを使用して startSession に応答します。

- AdvisoryMessages (StatusCode が 0 以外の場合)
- ApiVersion

- SessionID
- StatusCode

例

以下の例は、startSession を呼び出す 1 つの方法を示します。

```
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 は、startSession によって返されるレスポンス・オブジェクトを処理するメソッドです。

```
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());
}
```

予約パラメーター

Interact API で使用される予約パラメーターがいくつかあります。ランタイム・サーバー用に必要なものや、追加機能に使用できるものがあります。

postEvent 機能

機能	パラメーター	説明
カスタム・テーブルへの記録	UACICustomLoggerTableName	ランタイム・テーブルのデータ・ソースのテーブルの名前。このパラメーターと有効なテーブル名を指定した場合、ランタイム環境では選択したテーブルにすべてのセッション・データが書き込まれます。セッション・データ NameValuePair と一致する、テーブル内のすべての列名のデータが追加されます。ランタイム環境では、セッションの名前と値のペアが NULL と一致しない列のデータが追加されます。 customLogger 構成プロパティーを使用してデータベースに書き込むプロセスを管理できます。

機能	パラメーター	説明
複数のレスポンス・タイプ	UACILogToLearning	値が 1 または 0 の整数。1 はランタイム環境でイベントを学習用に承認として記録することを示します。 0 はランタイム環境でイベントを学習用に記録しないことを示します。このパラメーターを使用することで、学習に影響を及ぼさずに異なるレスポンス・タイプをログに記録する、複数の postEvent メソッドを作成することができます。コンタクト、承認、または拒否を記録するように設定されたイベントに対して、このパラメーターを定義する必要はありません。このパラメーターは UACIResponseTypeCode と共に使用する必要があります。 UACILOGTOLEARNING を定義しない場合、ランタイム環境では、デフォルト値である 0 と見なされます (イベントがログのコンタクト、承認、または拒否をトリガーする場合を除く)。
	UACIResponseTypeCode	レスポンス・タイプ・コードを表す値。値は、UA_UsrResponseType テーブルの有効なエントリーでなければなりません。
レスポンス・トラッキング	UACIOfferTrackingCode	オファーの処理コード。イベントでコンタクトまたはレスポンスの履歴への記録が行われる場合は、このパラメーターを定義する必要があります。イベントごとに処理コードを 1 つのみ渡すことができます。オファー・コンタクトの処理コードを渡さない場合、ランタイム環境では、オファーの最後の推奨リストにあるすべてのオファーについて、オファー・コンタクトが記録されます。応答の処理コードを渡さない場合、ランタイム環境でエラーが返されます。クロス・セッション・レスポンス・トラッキングを構成する場合は、UACIOfferTrackingcodeType パラメーターを使用して、処理コード以外に使用するトラッキング・コードのタイプを定義できます。
クロス・セッション・レスポンス・トラッキング	UACIOfferTrackingCodeType	トラッキング・コード・タイプを定義する数値。1 はデフォルトの処理コードで、2 はオファー・コードです。すべてのコードは、UACI_TrackingType テーブルの有効なエントリーでなければなりません。このテーブルにはその他のカスタム・コードを追加できます。

機能	パラメーター	説明
特定のフローチャートの実行	UACIExecuteFlowchartByName	セグメント化をトリガーするメソッド (再セグメント化をトリガーする startSession、setAudience、または postEvent) に対してこのパラメーターを定義する場合、現在のオーディエンス・レベルのすべてのフローチャートを実行する代わりに、Interact は指定されたフローチャートのみを実行します。フローチャートのリストをパイプ (|) 文字で区切って指定できます。

ランタイム環境の予約パラメーター

ランタイム環境では、以下の予約パラメーターが使用されます。イベント・パラメーターに以下の名前を使用しないでください。

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

AdvisoryMessage クラスについて

advisoryMessage クラスには、アドバイス・メッセージ・オブジェクトを定義するメソッドが含まれます。アドバイス・メッセージ・オブジェクトは、レスポンス・オブジェクトに含まれます。InteractAPI 内のメソッドはすべて、レスポンス・オブジェクトを返します。(batchResponse オブジェクトを返す executeBatch メソッドは除きます。) エラーまたは警告がある場合、Interact サーバーは、アドバイス・メッセージ・オブジェクトを挿入します。アドバイス・メッセージ・オブジェクトには、以下の属性が含まれています。

- **DetailMessage** — アドバイス・メッセージの詳細な説明。この属性は、すべてのアドバイス・メッセージに使用できるわけではありません。使用可能な場合、DetailMessage はローカライズされない可能性があります。
- **メッセージ** — アドバイス・メッセージの簡略説明。
- **MessageCode** — アドバイス・メッセージのコード番号。
- **StatusLevel** — アドバイス・メッセージの重大度のコード番号。

advisoryMessage オブジェクトは、getAdvisoryMessages メソッドを使用して取得します。

getDetailMessage

```
getDetailMessage()
```

getDetailMessage メソッドは、アドバイザリー・メッセージ・オブジェクトの詳細な説明を返します。

すべてのメッセージに詳細メッセージがあるわけではありません。

戻り値

アドバイザリー・メッセージ・オブジェクトは文字列を返します。

例

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

getMessage

```
getMessage()
```

getMessage メソッドは、アドバイザリー・メッセージ・オブジェクトの要旨を返します。

戻り値

アドバイザリー・メッセージ・オブジェクトは文字列を返します。

例

以下のメソッドは、AdvisoryMessage オブジェクトのメッセージと詳細メッセージを出力します。

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

getMessageCode

```
getMessageCode()
```

getMessageCode メソッドは、ステータス・レベルが 2 (STATUS_LEVEL_ERROR) の場合、アドバイザリー・メッセージ・オブジェクトに関連付けられた内部エラー・コードを返します。

戻り値

AdvisoryMessage オブジェクトは整数を返します。

例

以下のメソッドは、AdvisoryMessage オブジェクトのメッセージ・コードを出力します。

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

getStatusLevel

```
getStatusLevel()
```

getStatusLevel メソッドは、アドバイザリー・メッセージ・オブジェクトのステータス・レベルを返します。

戻り値

アドバイザリー・メッセージ・オブジェクトは整数を返します。

- 0 - STATUS_LEVEL_SUCCESS — 呼び出されたメソッドはエラーなく完了しました。
- 1 - STATUS_LEVEL_WARNING — 呼び出されたメソッドは 1 つ以上の警告を伴って完了しました (エラーはなし)。
- 2 - STATUS_LEVEL_ERROR — 呼び出されたメソッドは正常に完了しませんでした。1 つ以上のエラーがあります。

例

以下のメソッドは、AdvisoryMessage オブジェクトのステータス・レベルを出力します。

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

AdvisoryMessageCode クラスについて

advisoryMessageCode クラスには、アドバイス・メッセージ・コードを定義するメソッドが含まれます。アドバイス・メッセージ・コードは、getMessageCode メソッドを使用して取得します。

アドバイザリー・メッセージ・コード

コード	説明
1	INVALID_SESSION_ID - セッション ID が有効なセッションを参照していません。
2	ERROR_TRYING_TO_ABORT_SEGMENTATION - endSession でセグメント化を中止しようとしてエラーが発生しました。

コード	説明
3	INVALID_INTERACTIVE_CHANNEL - インタラクティブ・チャネル用に渡された引数が有効なインタラクティブ・チャネルを参照していません。
4	INVALID_EVENT_NAME - イベント用に渡された引数が、現在のインタラクティブ・チャネルに有効なイベントを参照していません。
5	INVALID_INTERACTION_POINT - インタラクション・ポイント用に渡された引数が、現在のインタラクティブ・チャネルに有効なインタラクション・ポイントを参照していません。
6	ERROR_WHILE_MAKING_INITIAL_SEGMENTATION_REQUEST - セグメント化要求の実行依頼時にエラーが発生しました。
7	SEGMENTATION_RUN_FAILED - セグメント化が一部実行され、その結果がエラーになりました。
8	PROFILE_LOAD_FAILED - プロファイル・テーブルまたはディメンション・テーブルのロードの試行に失敗しました。
9	OFFER_SUPPRESSION_LOAD_FAILED - オファー非表示テーブルのロードの試行に失敗しました。
10	COMMAND_METHOD_UNRECOGNIZED - `executeBatch` 内のコマンド用に指定されたコマンド・メソッドが有効ではありません。
11	ERROR_TRYING_TO_POST_EVENT_PARAMETERS - イベント・パラメーターの通知時にエラーが発生しました。
12	LOG_SYSTEM_EVENT_EXCEPTION - ロギング用のシステム・イベント (セッションの終了、オファーの取得、プロファイルの取得、オーディエンスの設定、デバッグの設定、またはセッションの開始) を実行依頼しようとして例外が発生しました。
13	LOG_USER_EVENT_EXCEPTION - ロギング用のユーザー・イベントを実行依頼しようとして例外が発生しました。
14	ERROR_TRYING_TO_LOOK_UP_EVENT - イベント名を検索しようとしてエラーが発生しました。
15	ERROR_TRYING_TO_LOOK_UP_INTERACTIVE_CHANNEL - インタラクティブ・チャネル名を検索しようとしてエラーが発生しました。
16	ERROR_TRYING_TO_LOOK_UP_INTERACTION_POINT - インタラクション・ポイント名を検索しようとしてエラーが発生しました。
17	RUNTIME_EXCEPTION_ENCOUNTERED - 予期しない実行時例外が発生しました。
18	ERROR_TRYING_TO_EXECUTE_ASSOCIATED_ACTION - 関連アクション (再セグメントのトリガー、オファー・コンタクトをログに記録、オファー承認をログに記録、またはオファー拒否をログに記録) を実行しようとしてエラーが発生しました。
19	ERROR_TRYING_RUN_FLOWCHART - フローチャートを実行しようとしてエラーが発生しました。
20	FLOWCHART_FAILED - フローチャートが失敗しました。
21	FLOWCHART_ABORTED - フローチャートが中止されました。
22	FLOWCHART_NEVER_RUN - フローチャートは実行されません。
23	FLOWCHART_STILL_RUNNING - フローチャートはまだ実行中です。
24	ERROR_WHILE_READING_PARAMETERS - パラメーターの読み取り中にエラーが発生しました。

コード	説明
25	ERROR_WHILE_LOADING_RECOMMENDED_OFFERS - 推奨オファーのロード中にエラーが発生しました。
26	ERROR_WHILE_LOGGING_DEFAULT_TEXT_STATISTICS - デフォルトのテキスト統計 (インタラクション・ポイントのデフォルト文字列が表示された回数) のロギング中にエラーが発生しました。
27	SCORE_OVERRIDE_LOAD_FAILED - スコア・オーバーライド・テーブルのロードに失敗しました。
28	NULL_AUDIENCE_ID - オーディエンス ID が空です。
29	UNRECOGNIZED_AUDIENCE_LEVEL - 認識されていないオーディエンス・レベルです。
30	MISSING_AUDIENCE_FIELD - オーディエンス・フィールドが見つかりません。
31	INVALID_AUDIENCE_FIELD_TYPE - 無効なオーディエンス・フィールド・タイプです。
32	UNSUPPORTED_AUDIENCE_FIELD_TYPE - サポートされていないオーディエンス・フィールド・タイプです。
33	TIMEOUT_REACHED_ON_GET_OFFERS_CALL - getOffers 呼び出しがオファーを返さずにタイムアウトしました。
34	INTERACT_INITIALIZATION_NOT_COMPLETED_SUCCESSFULLY - ランタイムの初期化は正常に完了しませんでした。
35	SESSION_ID_UNDEFINED - セッション ID が定義されていません。
36	INVALID_NUMBER_OF_OFFERS_REQUESTED - 要求されたオファーの数が無効です。
37	NO_SESSION_EXIST_BUT_WILL_CREATE_NEW_ONE
38	AUDIENCE_ID_NOT_FOUND_IN_PROFILE_TABLE
39	LOG_CUSTOM_LOGGER_EVENT_EXCEPTION - カスタム・ロギング・データ・イベントを実行依頼しようとして例外が発生しました。
40	SPECIFIED_FLOWCHART_FOR_EXECUTION_DOES_NOT_EXIST
41	AUDIENCE_NOT_DEFINED_IN_CONFIGURATION

BatchResponse クラスについて

BatchResponse クラスには、executeBatch メソッドの結果を定義するメソッドが含まれます。バッチ・レスポンス・オブジェクトには、以下の属性が含まれます。

- **BatchStatusCode** — executeBatch メソッドによって要求されるすべてのレスポンスのステータス・コードの最高値。
- **レスポンス** — executeBatch メソッドによって要求されるレスポンス・オブジェクトの配列。

getBatchStatusCode

```
getBatchStatusCode()
```

getBatchStatusCode メソッドは、executeBatch メソッドで実行されたコマンドの配列から、最も大きいステータス・コードを返します。

戻り値

getBatchStatusCode メソッドは整数を返します。

- 0 - STATUS_SUCCESS — 呼び出されたメソッドはエラーなく完了しました。
- 1 - STATUS_WARNING — 呼び出されたメソッドは 1 つ以上の警告を伴って完了しました (エラーはなし)。
- 2 - STATUS_ERROR — 呼び出されたメソッドは正常に完了しませんでした。1 つ以上のエラーがあります。

例

以下のサンプル・コードは、BatchStatusCode を取得する方法の例を示します。

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

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

getResponses

```
getResponses()
```

getResponses メソッドは、executeBatch メソッドで実行されたコマンドの配列に対応する、レスポンス・オブジェクトの配列を返します。

戻り値

getResponses メソッドは、Response オブジェクトの配列を返します。

例

以下の例では、すべてのレスポンスを選択し、コマンドが成功しなかった場合のアドバイザリー・メッセージがあれば出力します。

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

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

コマンド・インターフェースについて

executeBatch メソッドには、コマンド・インターフェースを実装するオブジェクトの配列を渡す必要があります。デフォルトの実装である CommandImpl を使用して、コマンド・オブジェクトを渡す必要があります。

次の表では、コマンド、そのコマンドが表す InteractAPI クラスのメソッド、および各コマンドに対して使用する必要があるコマンド・インターフェース・メソッドをリストしています。executeBatch メソッドには既にセッション ID が組み込まれているため、セッション ID を組み込む必要はありません。

コマンド	Interact API メソッド	コマンド・インターフェース・メソッド
COMMAND_ENDSESSION	endSession	なし。
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	なし。
COMMAND_GETVERSION	getVersion	なし。
COMMAND_POSTEVENT	postEvent	• setEvent • setEventParameters
COMMAND_SETAUDIENCE	setAudience	• setAudienceID • setAudienceLevel • setEventParameters
COMMAND_SETDEBUG	setDebug	setDebug
COMMAND_STARTSESSION	startSession	• setAudienceID • setAudienceLevel • setDebug • setEventParameters • setInteractiveChannel • setRelyOnExistingSession

setAudienceID

setAudienceID(*audienceID*)

setAudienceID メソッドは、setAudience コマンドおよび startSession コマンドの AudienceID を定義します。

- **audienceID** — AudienceID を定義する NameValuePair オブジェクトの配列。

戻り値

ありません。

例

以下の例は、startSession および setAudience を呼び出す executeBatch メソッドからの抜粋です。

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

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

setAudienceLevel

setAudienceLevel(*audienceLevel*)

setAudienceLevel メソッドは、setAudience コマンドおよび startSession コマンドのオーディエンス・レベルを定義します。

- *audienceLevel* — オーディエンス・レベルを含む文字列。

  重要: *audienceLevel* の名前は、Campaign で定義されているオーディエンス・レベルの名前と正確に一致する必要があります。大/小文字の区別があります。

戻り値

ありません。

例

以下の例は、startSession および setAudience を呼び出す executeBatch メソッドからの抜粋です。

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

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

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

setDebug

setDebug(*debug*)

setDebug メソッドは、startSession コマンドのデバッグ・レベルを定義します。true の場合、ランタイム・サーバーはデバッグ情報をランタイム・サーバー・ログに記録します。false の場合、ランタイム・サーバーはデバッグ情報をログに記録しません。各セッションに対して個々にデバッグ・フラグが設定されます。このため、個々のランタイム・セッションのデバッグ・データをトレースできます。

- **debug** — ブール (true または false)。

戻り値

ありません。

例

以下の例は、startSession および setDebug を呼び出す executeBatch メソッドからの抜粋です。

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

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

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

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

setEvent

setEvent(*event*)

setEvent メソッドは、postEvent コマンドが使用するイベントの名前を定義します。

- **event** — イベント名を含む文字列。

重要: *event* の名前は、インタラクティブ・チャネルで定義されているイベントの名前と正確に一致する必要があります。大/小文字の区別があります。

戻り値

ありません。

例

以下の例は、postEvent を呼び出す executeBatch メソッドからの抜粋です。

```
String eventName = "SearchExecution";

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

setEventParameters

```
setEventParameters(eventParameters)
```

setEventParameters メソッドは、postEvent コマンドで使用するイベント・パラメーターを定義します。これらの値はセッション・データに格納されます。

- **eventParameters** — イベント・パラメーターを定義する NameValuePair オブジェクトの配列。

例えば、イベントでオファーをコンタクト履歴に記録している場合は、オファーの処理コードを含める必要があります。

戻り値

ありません。

例

以下の例は、postEvent を呼び出す executeBatch メソッドからの抜粋です。

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

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

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

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

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

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

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

setGetOfferRequests

setGetOfferRequests(*numberRequested*)

setGetOfferRequests メソッドは、getOffersForMultipleInteractionPoints コマンドで使用するオファーを取得するためのパラメーターを設定します。

- **numberRequested** — オファーを取得するためのパラメーターを定義する GetOfferRequest オブジェクトの配列。

戻り値

ありません。

例

以下の例は、setGetOfferRequests を呼び出す GetOfferRequest メソッドからの抜粋です。

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

```
GetOfferRequest request2 = new GetOfferRequest(3, GetOfferRequest.ALLOW_DUPLICATION);
request2.setIpName("IP2");
OfferAttributeRequirements offerAttributes2 = new OfferAttributeRequirements();
offerAttributes2.setNumberRequested(3);
offerAttributes2.setAttributes(new NameValuePairImpl[] {new NameValuePairImpl("attr5",
     NameValuePair.DATA_TYPE_STRING, "value5")});
request2.setOfferAttributes(offerAttributes2);

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

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

setInteractiveChannel

setInteractiveChannel(*interactiveChannel*)

setInteractiveChannel メソッドは、startSession コマンドで使用するインタラクティブ・チャネルの名前を定義します。

- **interactiveChannel** — インタラクティブ・チャネル名を含む文字列。

重要: *interactiveChannel* は、Campaign で定義されているインタラクティブ・チャネルの名前と正確に一致する必要があります。大/小文字の区別があります。

戻り値

ありません。

例

以下の例は、startSession を呼び出す executeBatch メソッドからの抜粋です。

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

setInteractionPoint

setInteractionPoint(*interactionPoint*)

setInteractionPoint メソッドは、getOffers コマンドおよび postEvent コマンドで使用するインタラクション・ポイントの名前を定義します。

- **interactionPoint** — インタラクション・ポイント名を含む文字列。

重要: *interactionPoint* は、インタラクティブ・チャネルで定義されているインタラクション・ポイントの名前と正確に一致する必要があります。大/小文字の区別があります。

戻り値

ありません。

例

以下の例は、getOffers を呼び出す executeBatch メソッドからの抜粋です。

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

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

setMethodIdentifier

setMethodIdentifier(*methodIdentifier*)

setMethodIdentifier メソッドは、コマンド・オブジェクトに含まれるコマンドのタイプを定義します。

- **methodIdentifier** — コマンドのタイプを含む文字列。

  有効な値は以下のとおりです。

  - **COMMAND_ENDSESSION** — endSession メソッドを表します。
  - **COMMAND_GETOFFERS** — getOffers メソッドを表します。
  - **COMMAND_GETPROFILE** — getProfile メソッドを表します。
  - **COMMAND_GETVERSION** — getVersion メソッドを表します。
  - **COMMAND_POSTEVENT** — postEvent メソッドを表します。
  - **COMMAND_SETAUDIENCE** — setAudience メソッドを表します。
  - **COMMAND_SETDEBUG** — setDebug メソッドを表します。
  - **COMMAND_STARTSESSION** — startSession メソッドを表します。

戻り値

ありません。

例

以下の例は、getVersion および endSession を呼び出す executeBatch メソッドからの抜粋です。

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

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

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

setNumberRequested

setNumberRequested(*numberRequested*)

setNumberRequested メソッドは、getOffers コマンドから要求されるオファーの数を定義します。

- **numberRequested** — getOffers コマンドで要求されるオファーの数を定義する整数。

戻り値

ありません。

例

以下の例は、getOffers を呼び出す executeBatch メソッドからの抜粋です。

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

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

setRelyOnExistingSession

setRelyOnExistingSession(*relyOnExistingSession*)

setRelyOnExistingSession メソッドは、startSession コマンドで既存のセッションを使用するかどうかを定義するブールを定義します。

true の場合、executeBatch のセッション ID は既存のセッション ID と一致する必要があります。false の場合、executeBatch メソッドを使用して、新しいセッション ID を提供する必要があります。

- **relyOnExistingSession** — ブール (true または false)。

戻り値

ありません。

例

以下の例は、startSession を呼び出す executeBatch メソッドからの抜粋です。

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

NameValuePair インターフェースについて

Interact API のメソッドの多くは、NameValuePair オブジェクトを返すか、あるいは NameValuePair オブジェクトを引数として渡すことを要求します。引数としてメソッドに渡す場合は、デフォルトの実装である NameValuePairImpl を使用する必要があります。

getName

getName()

getName メソッドは、NameValuePair オブジェクトの名前コンポーネントを返します。

戻り値

getName メソッドは文字列を返します。

例

以下の例は、getProfile のレスポンス・オブジェクトを処理するメソッドからの例外です。

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

getValueAsDate

```
getValueAsDate()
```

getValueAsDate メソッドは、NameValuePair オブジェクトの値を返します。

正しいデータ型を参照していることを確認するには、getValueAsDate を使用する前に getValueDataType を使用する必要があります。

戻り値

getValueAsDate メソッドは日付を返します。

例

以下の例は、値が日付の場合に、NameValuePair を処理し、その値を出力するメソッドからの例外です。

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

getValueAsNumeric

```
getValueAsNumeric()
```

getValueAsNumeric メソッドは、NameValuePair オブジェクトの値を返します。

正しいデータ型を参照していることを確認するには、getValueAsNumeric を使用する前に getValueDataType を使用する必要があります。

戻り値

getValueAsNumeric メソッドは double を返します。例えば、プロファイル・テーブルに最初から格納されている値を Integer として取得している場合、getValueAsNumeric は double を返します。

例

以下の例は、値が数値の場合に、NameValuePair を処理し、その値を出力するメソッドからの例外です。

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

getValueAsString

```
getValueAsString()
```

getValueAsString メソッドは、NameValuePair オブジェクトの値を返します。

正しいデータ型を参照していることを確認するには、getValueAsString を使用する前に getValueDataType を使用する必要があります。

戻り値

getValueAsString メソッドは文字列を返します。例えば、プロファイル・テーブルに最初から格納されている値を char、varchar、または char[10] として取得している場合、getValueAsString は文字列を返します。

例

以下の例は、値が文字列の場合に、NameValuePair を処理し、その値を出力するメソッドからの例外です。

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

getValueDataType

```
getValueDataType()
```

getValueDataType メソッドは、NameValuePair オブジェクトのデータ型を返します。

正しいデータ型を参照していることを確認するには、getValueAsDate、getValueAsNumeric、または getValueAsString を使用する前に getValueDataType を使用する必要があります。

戻り値

getValueDataType メソッドは、NameValuePair にデータ、数値、または文字列が含まれているかどうかを示す文字列を返します。

有効な値は以下のとおりです。

- **DATA_TYPE_DATETIME** — 日時の値を含む日付。
- **DATA_TYPE_NUMERIC** — 数値を含む double。
- **DATA_TYPE_STRING** — テキスト値を含む文字列。

例

以下の例は、getProfile メソッドからのレスポンス・オブジェクトを処理するメソッドからの例外です。

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

setName

```
setName(name)
```

setName メソッドは、NameValuePair オブジェクトの名前コンポーネントを定義します。

- **name** — NameValuePair オブジェクトの名前コンポーネントを含む文字列。

戻り値

ありません。

例

以下の例は、NameValuePair の名前コンポーネントを定義する方法を示します。

```
NameValuePair custId = new NameValuePairImpl();
custId.setName("CustomerId");
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
```

setValueAsDate

```
setValueAsDate(valueAsDate)
```

setValueAsDate メソッドは、NameValuePair オブジェクトの値を定義します。

- **valueAsDate** — NameValuePair オブジェクトの日時の値を含む日付。

戻り値

ありません。

例

以下の例は、NameValuePair の値コンポーネントが日付の場合に、その値を定義する方法を示します。

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

setValueAsNumeric

`setValueAsNumeric(valueAsNumeric)`

`setValueAsNumeric` メソッドは、NameValuePair オブジェクトの値を定義します。

- **valueAsNumeric** — NameValuePair オブジェクトの数値を含む double。

戻り値

ありません。

例

以下の例は、NameValuePair の値コンポーネントが数値の場合に、その値を定義する方法を示します。

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

setValueAsString

`setValueAsString(valueAsString)`

`setValueAsString` メソッドは、NameValuePair オブジェクトの値を定義します。

- **valueAsString** — NameValuePair オブジェクトの値を含む文字列

戻り値

ありません。

例

以下の例は、NameValuePair の値コンポーネントが数値の場合に、その値を定義する方法を示します。

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

setValueDataType

`getValueDataType(valueDataType)`

`setValueDataType` メソッドは、NameValuePair オブジェクトのデータ型を定義します。

有効な値は以下のとおりです。

- **DATA_TYPE_DATETIME** — 日時の値を含む日付。
- **DATA_TYPE_NUMERIC** — 数値を含む double。

- **DATA_TYPE_STRING** — テキスト値を含む文字列。

戻り値

ありません。

例

以下の例は、NameValuePair の値のデータ型を設定する方法を示します。

```
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);
```

オファー・クラスについて

「オファー」クラスには、オファー・オブジェクトを定義するメソッドが含まれます。このオファー・オブジェクトには、Campaign 内のオファーの同じプロパティーが多数含まれます。オファー・オブジェクトには、以下の属性が含まれています。

- **AdditionalAttributes** — Campaign で定義した任意のカスタム・オファー属性が含まれる NameValuePairs。
- **説明** — オファーの説明。
- **EffectiveDate** — オファーの発効日。
- **ExpirationDate** — オファーの有効期限。
- **OfferCode** — オファーのオファー・コード。
- **OfferName** — オファーの名前。
- **TreatmentCode** — オファーの処理コード。
- **スコア** — オファーのマーケティング・スコア、または、enableScoreOverrideLookup プロパティーが true の場合には、ScoreOverrideTable によって定義されているスコア。

getAdditionalAttributes

```
getAdditionalAttributes()
```

getAdditionalAttributes メソッドは、Campaign で定義されるカスタム・オファー属性を返します。

戻り値

getAdditionalAttributes メソッドは、NameValuePair オブジェクトの配列を返します。

例

以下の例では、すべての追加属性間でソートし、発効日と有効期限をチェックして、その他の属性を出力します。

```
for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
{
    // check to see if the effective date exists
    if(offerAttribute.getName().equalsIgnoreCase("effectiveDate"))
    {
        System.out.println("Found effective date");
    }
    // check to see if the expiration date exists
    else if(offerAttribute.getName().equalsIgnoreCase("expirationDate"))
    {
        System.out.println("Found expiration date");
    }
    printNameValuePair(offerAttribute);
    }
}
public static void printNameValuePair(NameValuePair nvp)
{
    // print out the name:
    System.out.println("Name:"+nvp.getName());

    // based on the datatype, call the appropriate method to get the value
    if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_DATETIME)
        System.out.println("DateValue:"+nvp.getValueAsDate());
    else if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_NUMERIC)
        System.out.println("NumericValue:"+nvp.getValueAsNumeric());
    else
        System.out.println("StringValue:"+nvp.getValueAsString());
}
```

getDescription

```
getDescription()
```

getDescription メソッドは、Campaign で定義されるオファーの説明を返します。

戻り値

getDescription メソッドは文字列を返します。

例

以下の例は、オファーの説明を出力します。

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
     System.out.println("Offer Description:"+offer.getDescription());
}
```

getOfferCode

```
getOfferCode()
```

getOfferCode メソッドは、Campaign で定義されているオファーのオファー・コードを返します。

戻り値

getOfferCode メソッドは、オファーのオファー・コードを含む文字列の配列を返します。

例

以下の例は、オファーのオファー・コードを出力します。

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
   System.out.println("Offer Code:"+offer.getOfferCode());
}
```

getOfferName

```
getOfferName()
```

getOfferName メソッドは、Campaign で定義されているオファーの名前を返します。

戻り値

getOfferName メソッドは文字列を返します。

例

以下の例は、オファーの名前を出力します。

```
for(Offer offer : offerList.getRecommendedOffers())
{
// print offer
System.out.println("Offer Name:"+offer.getOfferName());
}
```

getScore

```
getScore()
```

getScore メソッドは、次のうちのいずれか 1 つを返します。

- デフォルト・オファー・テーブル、スコア・オーバーライド・テーブル、または組み込み学習を有効にしていない場合、このメソッドは、インタラクション方法タブで定義されているオファーのマーケティング・スコアを返します。
- デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルを有効にしていて、組み込み学習を有効にしていない場合、このメソッドは、デフォルト・オファー・テーブル、マーケティング担当者のスコア、およびスコア・オーバーライド・テーブルの間での優先順位によって定義されているオファーのスコアを返します。
- 組み込み学習を有効にしている場合、このメソッドは、組み込み学習でオファーを並べ替えるために使用した最終スコアを返します。

戻り値

getScore メソッドは、オファーのスコアを示す整数を返します。

例

以下の例は、オファーのスコアを出力します。

```
for(Offer offer : offerList.getRecommendedOffers())
{
// print offer
System.out.println("Offer Score:"+offer.getOfferScore());
}
```

getTreatmentCode

```
getTreatmentCode()
```

getTreatmentCode メソッドは、Campaign で定義されているオファーの処理コードを返します。

Campaign が処理コードを使用して、提供されるオファーのインスタンスを識別するため、postEvent メソッドを使用して、オファーのコンタクト、承認、または拒否のイベントを記録するときに、イベント・パラメーターとしてこのコードを返す必要があります。オファーの承認または拒否を記録している場合は、UACIOfferTrackingCode に対する処理コードを示す NameValuePair の名前の値を設定する必要があります。

戻り値

getTreatmentCode メソッドは文字列を返します。

例

以下の例は、オファーの処理コードを出力します。

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // print offer
    System.out.println("Offer Treatment Code:"+offer.getTreatmentCode());
}
```

OfferList クラスについて

OfferList クラスには、getOffers メソッドの結果を定義するメソッドが含まれます。OfferList オブジェクトには、以下の属性が含まれています。

- **DefaultString** — インタラクティブ・チャネル内のインタラクション・ポイント用に定義されているデフォルトのストリング。
- **RecommendedOffers** — getOffers メソッドによって要求されるオファー・オブジェクトの配列。

OfferList クラスは、オファーのリストに関する作業を行います。このクラスは、Campaign オファー・リストとは関係ありません。

getDefaultString

```
getDefaultString()
```

getDefaultString メソッドは、Campaign で定義されているインタラクション・ポイントのデフォルトの文字列を返します。

RecommendedOffers オブジェクトが空の場合は、何らかのコンテンツを示すようにするため、この文字列を表すタッチポイントを構成する必要があります。RecommendedOffers オブジェクトが空である場合にのみ、Interact は DefaultString オブジェクトを取り込みます。

戻り値

getDefaultString メソッドは文字列を返します。

例

以下の例は、offerList オブジェクトにオファーが含まれていない場合、デフォルトの文字列を取得します。

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
  System.out.println("Offer Name:"+offer.getOfferName());
    }
}
else // count on the default Offer String
    System.out.println("Default offer:"+offerList.getDefaultString());
```

getRecommendedOffers

```
getRecommendedOffers()
```

getRecommendedOffers メソッドは、getOffers メソッドによって要求される Offer オブジェクトの配列を返します。

getRecommendedOffer への応答が空の場合、タッチポイントが getDefaultString の結果を示す必要があります。

戻り値

getRecommendedOffers メソッドは Offer オブジェクトを返します。

例

以下の例は、OfferList オブジェクトを処理し、すべての推奨オファーのオファー名を出力します。

```
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());
```

レスポンス・クラスについて

「レスポンス」クラスには、任意の InteractAPI クラス・メソッドの結果を定義するメソッドが含まれます。レスポンス・オブジェクトには、以下の属性が含まれます。

- **AdvisoryMessages** — アドバイス・メッセージの配列。この属性は、メソッドの実行中に警告またはエラーが発生した場合にのみ、追加されます。
- **ApiVersion** — API バージョンが含まれているストリング。この属性は、`getVersion` メソッドによって追加されます。
- **OfferList** — OfferList オブジェクトには、`getOffers` メソッドによって要求されるオファーが含まれます。
- **ProfileRecord** — プロファイル・データが含まれる NameValuePairs の配列。この属性は、`getProfile` メソッドによって追加されます。
- **SessionID** — セッション ID を定義するストリング。これはすべての `InteractAPI` クラス・メソッドによって返されます。
- **StatusCode** — メソッドが、エラーなし、警告あり、またはエラーありのどの状態で実行されたかを示す数値。これはすべての `InteractAPI` クラス・メソッドによって返されます。

getAdvisoryMessages

```
getAdvisoryMessages()
```

`getAdvisoryMessages` メソッドは、レスポンス・オブジェクトからのアドバイザリー・メッセージの配列を返します。

戻り値

`getAdvisoryMessages` メソッドは、アドバイザリー・メッセージ・オブジェクトの配列を返します。

例

以下の例は、レスポンス・オブジェクトから AdvisoryMessage オブジェクトを取得し、そのオブジェクトをすべて繰り返し適用して、メッセージを出力します。

```
AdvisoryMessage[] messages = response.getAdvisoryMessages();
    for(AdvisoryMessage msg : messages)
    {
        System.out.println(msg.getMessage());
        // Some advisory messages may have additional detail:
        System.out.println(msg.getDetailMessage());
    }
```

getApiVersion

```
getApiVersion()
```

`getApiVersion` メソッドは、レスポンス・オブジェクトの API バージョンを返します。

`getVersion` メソッドは、レスポンス・オブジェクトの ApiVersion 属性のデータを設定します。

戻り値

レスポンス・オブジェクトは文字列を返します。

例

以下の例は、getVersion のレスポンス・オブジェクトを処理するメソッドからの例外です。

```
if(response.getStatusCode() == Response.STATUS_SUCCESS)
{
    System.out.println("getVersion call processed with no warnings or errors");
    System.out.println("API Version:" + response.getApiVersion());
}
```

getOfferList

```
getOfferList()
```

getOfferList メソッドは、レスポンス・オブジェクトの OfferList オブジェクトを返します。

getOffers メソッドは、レスポンス・オブジェクトの OfferList オブジェクトのデータを設定します。

戻り値

レスポンス・オブジェクトは OfferList オブジェクトを返します。

例

以下の例は、getOffers のレスポンス・オブジェクトを処理するメソッドからの例外です。

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
    // print offer
    System.out.println("Offer Name:"+offer.getOfferName());
    }
}
```

getAllOfferLists

```
getAllOfferLists()
```

getAllOfferLists メソッドは、レスポンス・オブジェクトのすべての OfferLists の配列を返します。

これは、レスポンス・オブジェクトの OfferList 配列オブジェクトのデータを設定する getOffersForMultipleInteractionPoints メソッドによって使用されます。

戻り値

レスポンス・オブジェクトは OfferList 配列を返します。

例

以下の例は、getOffers のレスポンス・オブジェクトを処理するメソッドからの例外です。

```
OfferList[] allOfferLists = response.getAllOfferLists();
 if (allOfferLists != null) {
  for (OfferList ol : allOfferLists) {
   System.out.println("The following offers are delivered for interaction point "
     + ol.getInteractionPointName() + ":");
   for (Offer o : ol.getRecommendedOffers()) {
    System.out.println(o.getOfferName());
   }
  }
 }
```

getProfileRecord

```
getProfileRecord()
```

getProfileRecord メソッドは、NameValuePair オブジェクトの配列として、現在のセッションのプロファイル・レコードを返します。これらのプロファイル・レコードには、ランタイム・セッションの初期に追加された eventParameters も含まれています。

getProfile メソッドは、レスポンス・オブジェクトのプロファイル・レコード NameValuePair オブジェクトのデータを設定します。

戻り値

レスポンス・オブジェクトは、NameValuePair オブジェクトの配列を返します。

例

以下の例は、getOffers のレスポンス・オブジェクトを処理するメソッドからの例外です。

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

getSessionID

```
getSessionID()
```

getSessionID メソッドはセッション ID を返します。

戻り値

getSessionID メソッドは文字列を返します。

例

以下の例は、エラーが関連するセッションを示すために、エラー処理の最後または最初に表示できるメッセージを示します。

```
System.out.println("This response pertains to sessionId:"+response.getSessionID());
```

getStatusCode

```
getStatusCode()
```

getStatusCode メソッドは、レスポンス・オブジェクトのステータス・コードを返します。

戻り値

レスポンス・オブジェクトは整数を返します。

- 0 - STATUS_SUCCESS - 呼び出されたメソッドはエラーなく完了しました。アドバイザリー・メッセージはある場合とない場合があります。
- 1 - STATUS_WARNING - 呼び出されたメソッドは 1 つ以上の警告メッセージを伴って完了しました (エラーはなし)。詳細については、アドバイザリー・メッセージを照会してください。
- 2 - STATUS_ERROR - 呼び出されたメソッドは正常に完了しませんでした。1 つ以上のエラー・メッセージがあります。詳細については、アドバイザリー・メッセージを照会してください。

例

以下に、エラー処理で getStatusCode の使用方法の例を示します。

```
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());
}
```

第 8 章 ExternalCallout API について

Interact は、インタラクティブ・フローチャートで使用する拡張可能マクロ EXTERNALCALLOUT を提供します。このマクロを使用すれば、フローチャートの実行時に外部システムと通信するためのカスタム・ロジックを実行できます。例えば、フローチャートの実行時に顧客のクレジット・スコアを計算する場合、Java クラス (コールアウト) を作成して計算し、インタラクティブ・フローチャートの選択プロセスで EXTERNALCALLOUT マクロを使用して、コールアウトからクレジット・スコアを取得できます。

EXTERNALCALLOUT の構成には主な 2 つのステップがあります。まず、ExternalCallout API を実装する Java クラスを作成する必要があります。次に、Interact | フローチャート | ExternalCallouts カテゴリーでランタイム・サーバー上の必要な Marketing Platform 構成プロパティーを構成しなければなりません。

このセクションの情報に加え、 Interact ランタイム・サーバー上の Interact/docs/externalCalloutJavaDoc ディレクトリーにある ExternalCallout API の JavaDoc を使用できます。

IAffiniumExternalCallout インターフェース

ExternalCallout API は IAffiniumExternalCallout インターフェースに含まれています。 EXTERNALCALLOUT マクロを使用するには、IAffiniumExternalCallout インターフェースを実装する必要があります。

IAffiniumExternalCallout を実装するクラスには、ランタイム・サーバーによる初期化を可能にするコンストラクターが必要です。

- クラスにコンストラクターがない場合は、Java コンパイラーがデフォルトのコンストラクターを作成しますので、それで十分です。
- 引数を含むコンストラクターがある場合は、引数なしのパブリック・コンストラクター (ランタイム・サーバーが使用) を指定する必要があります。

外部コールアウトを作成する場合は、以下の点に注意してください。

- 外部コールアウトでのそれぞれの式評価でクラスの新規インスタンスが作成されます。クラスにおける静的メンバーのスレッド・セーフティー問題を管理する必要があります。
- 外部コールアウトでファイルやデータベース接続などのシステム・リソースを使用する場合は、接続を管理する必要があります。ランタイム・サーバーには、接続を自動的にクリーンアップする機能はありません。

IBM Unica Interact ランタイム環境のインストール先の lib ディレクトリーにある interact_externalcallout.jar に対して、実装アプリケーションをコンパイルする必要があります。

IAffiniumExternalCallout を使用すれば、ランタイム・サーバーは Java クラスのデータを要求できます。インターフェースは以下の 4 つのメソッドで構成されています。

- getNumberOfArguments
- getValue
- initialize
- shutdown

EXTERNALCALLOUT で使用する Web サービスを追加するには

EXTERNALCALLOUT マクロは、適切な構成プロパティーを定義した場合にのみ、コールアウトを認識します。

ランタイム環境用の Marketing Platform では、「Interact」>「フローチャート」>「externalCallouts」カテゴリーの以下の構成プロパティーを追加または定義します。

構成プロパティー	設定
externalCallouts カテゴリー	外部コールアウトの新規カテゴリーを作成します。
class	外部コールアウトのクラス名
classpath	外部コールアウト・クラス・ファイルへのクラスパス
パラメーター・データ・カテゴリー	外部コールアウトでパラメーターが必要な場合は、そのパラメーターの新しいパラメーター構成プロパティーを作成して、それぞれに value を割り当てます。

getNumberOfArguments

```
getNumberOfArguments()
```

getNumberOfArguments メソッドは、統合している対象の Java クラスで想定されている引数の数を返します。

戻り値

getNumberOfArguments メソッドは整数を返します。

例

以下の例は、引数の数の出力を示します。

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

```
getValue(audienceID, configData, arguments)
```

getValue メソッドは、コールアウトの中核機能を実行し、その結果を返します。

getValue メソッドには以下のパラメーターが必要です。

- **audienceID** — オーディエンス ID を識別する値。
- **configData** — コールアウトで必要な構成データのキーと値のペアによるマップ。
- **arguments** — コールアウトで必要な引数。各引数は、String、Double、Date、またはこのいずれかの List です。 List の引数には NULL 値を含めることができますが、例えば、String と Double を一緒に含めることはできません。

  引数タイプのチェックは、使用している実装内で行う必要があります。

getValue メソッドは、何らかの理由により失敗した場合、CalloutException を返します。

戻り値

getValue メソッドは String のリストを返します。

例

```
public List<String> getValue(AudienceId audienceId, Map<String,
  String> configurationData, Object... arguments) throws CalloutException
{
 Long customerId = (Long) audienceId.getComponentValue("Customer");
 // now query scoreQueryUtility for the credit score of customerId
 Double score = scoreQueryUtility.query(customerId);
 String str = Double.toString(score);
 List<String> list = new LinkedList<String>();
 list.add(str);
 return list;
}
```

initialize

```
initialize(configData)
```

initialize メソッドは、ランタイム・サーバーの始動時に一度呼び出されます。実行時にパフォーマンスの妨げになる可能性のある、データベース・テーブルのロードなどの操作がある場合は、その操作をこのメソッドで実行する必要があります。

initialize メソッドには以下のパラメーターが必要です。

- **configData** — コールアウトで必要な構成データのキーと値のペアによるマップ。

  Interact では、「Interact」>「フローチャート」>「外部コールアウト (External Callouts)」>「[外部コールアウト]」>「パラメーター・データ (Parameter Data)」カテゴリーで定義されている外部コールアウト・パラメーターからこれらの値の読み取りが行われます。

initialize メソッドは、何らかの理由により失敗した場合、CalloutException を返します。

戻り値

ありません。

例

```
public void initialize(Map<String, String> configurationData) throws CalloutException
{
    // configurationData has the key-value pairs specific to the environment
    // the server is running in
    // initialize scoreQueryUtility here
}
```

shutdown

```
shutdown(configData)
```

shutdown メソッドは、ランタイム・サーバーのシャットダウン時に 1 回呼び出されます。コールアウトで必要なクリーンアップ・タスクがある場合は、現時点でそのタスクを実行する必要があります。

shutdown メソッドには以下のパラメーターが必要です。

- **configData** — コールアウトで必要な構成データのキーと値のペアによるマップ。

shutdown メソッドは、何らかの理由により失敗した場合、CalloutException を返します。

戻り値

ありません。

例

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
    // shutdown scoreQueryUtility here
}
```

ExternalCallout API の例

1. 以下の内容を含む GetCreditScore.java というファイルを作成します。このファイルでは、モデリング・アプリケーションからスコアをフェッチする ScoreQueryUtility というクラスがあることを前提とします。

```
import java.util.Map;
import com.unicacorp.interact.session.AudienceId;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.IAffiniumExternalCallout;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.CalloutException;
import java.util.Random;

public class GetCreditScore implements IAffiniumExternalCallout
{
 // the class that has the logic to query an external system for a customer's credit score
 private static ScoreQueryUtility scoreQueryUtility;
 public void initialize(Map<String, String> configurationData) throws CalloutException
 {
  // configurationData has the key- value pairs specific to the environment the server is running in
  // initialize scoreQueryUtility here
 }

 public void shutdown(Map<String, String> configurationData) throws CalloutException
 {
 // shutdown scoreQueryUtility here
 }

 public int getNumberOfArguments()
```

```
{
 // do not expect any additional arguments other than the customer's id
 return 0;
}

public List<String> getValue(AudienceId audienceId, Map<String, String> configurationData,
 Object... arguments) throws CalloutException
{
 Long customerId = (Long) audienceId.getComponentValue("Customer");
 // now query scoreQueryUtility for the credit score of customerId
 Double score = scoreQueryUtility.query(customerId);
 String str = Double.toString(score);
 List<String> list = new LinkedList<String>();
 list.add(str);
 return list;
}
}
```

2. GetCreditScore.java を GetCreditScore.class にコンパイルします。
3. GetCreditScore.class およびこのファイルで使用する他のクラス・ファイルを含む creditscore.jar という JAR ファイルを作成します。
4. ランタイム・サーバー上の任意の場所 (/data/interact/creditscore.jar など) に JAR ファイルをコピーします。
5. 「構成管理」ページで、externalCallouts カテゴリーに名前が GetCreditScore でクラスパスが /data/interact/creditscore.jar の外部コールアウトを作成します。
6. インタラクティブ・フローチャートで、コールアウトを EXTERNALCALLOUT('GetCreditScore') として使用できます。

IInteractProfileDataService インターフェース

プロファイル・データ・サービス API は、インターフェース iInteractProfileDataService に含まれています。このインターフェースでは、Interact セッション開始時または Interact セッションのオーディエンス ID 変更時に、1 つ以上の外部データ・ソース (フラット・ファイル、Web サービスなど) を経由して階層データを Interact セッションにインポートできるようにします。

プロファイル・データ・サービス API を使用して階層データのインポートを開発するには、情報をなんらかのデータ・ソースから引き出し、それを ISessionDataRootNode オブジェクトにマップする Java クラスを記述し、それから、インタラクティブ・フローチャートの選択プロセスで EXTERNALCALLOUT マクロを使用してそのマップされたデータを参照することが必要です。

IBM Unica Interact ランタイム環境のインストール先の lib ディレクトリーにある interact_externalcallout.jar に対して、実装アプリケーションをコンパイルする必要があります。

このインターフェースを使用するための完全な Javadoc ドキュメンテーション・セットを確認するには、任意の Web ブラウザーで *Interact_home*/docs/externalCalloutJavaDoc にあるファイルを参照してください。

プロファイル・データ・サービスの使用方法を示すサンプル実装 (サンプルをどのように実装したかについての説明コメントを含む) を確認するには、*Interact_home*/samples/externalcallout/XMLProfileDataService.java を参照してください。

プロファイル・データ・サービスで使用するデータ・ソースを追加するには

適切な構成プロパティーが定義してあれば、EXTERNALCALLOUT マクロは、プロファイル・データ・サービス階層データのデータ・ソースをインポート専用として認識します。

ランタイム環境の Marketing Platform で、「Interact」>「プロファイル」>「オーディエンス・レベル」> [AudienceLevelName] >「プロファイル・データ・サービス」カテゴリーの以下の構成プロパティーを追加または定義します。

構成プロパティー	設定
「新規カテゴリー名」カテゴリー	定義しているデータ・ソースの名前。ここで入力する名前は、同一オーディエンス・レベルのデータ・ソース間で固有でなければなりません。
enabled	データ・ソースが定義されたオーディエンス・レベルで有効になっているかどうかを示します。
className	IInteractProfileDataService を実装するデータ・ソース・クラスの完全修飾名。
classPath	使用するプロファイル・データ・サービスのクラス・ファイルへのクラスパス。省略すると、デフォルトで、収容アプリケーション・サーバーのクラスパスが使用されます。
priority カテゴリー	このオーディエンス・レベル内でのこのデータ・ソースの優先度。各オーディエンス・レベルにおいて、すべてのデータ・ソース間で固有な値でなければなりません。(つまり、あるデータ・ソースで優先度を 100 に設定した場合、そのオーディエンス・レベルでは、他のどのデータ・ソースも優先度 100 にすることはできません。)

第 9 章 IBM Unica Interact ユーティリティー

このセクションでは、Interact で使用できる管理ユーティリティーについて説明します。

配置実行ユーティリティー (runDeployment.sh/.bat)

runDeployment コマンド・ライン・ツールを使用すると、コマンド・ラインから、deployment.properties ファイルで指定された設定を使用して特定のサーバー・グループにインタラクティブ・チャネルを配置できます。deployment.properties ファイルは使用可能なすべてのパラメーターの概要を示しており、runDeployment ツール自体と同じ場所にあります。インタラクティブ・チャネルの配置をコマンド・ラインから実行できる点は、OffersBySQL 機能を使用する場合に特に有用です。例えば、Campaign バッチ・フローチャートを構成して、定期的に実行できます。フローチャートの実行が完了すると、このコマンド・ライン・ツールを使用して、OffersBySQL テーブル内のオファーの配置を初期化するためトリガーを呼び出すことができます。

説明

runDeployment コマンド・ライン・ツールは、 Interact 設計時サーバーの次の場所に自動的にインストールされます。

Interact_home/interactDT/tools/deployment/runDeployment.sh (Windows Server の場合は runDeployment.bat)

コマンドに渡される唯一の引数は deployment.properties というファイルの場所であり、このファイルに、インタラクティブ・チャネル/ランタイム・サーバー・グループの組み合わせの配置に必要な使用可能なすべてのパラメーターが記述されます。サンプル・ファイルが参照用に提供されています。

注: runDeployment ユーティリティーを使用するには、まず任意のテキスト・エディターを使用して、サーバー上の Java ランタイム環境の場所を指定するように編集する必要があります。例えば、このユーティリティーが使う Java ランタイムが含まれているディレクトリー *Interact_home*/jre あるいは *Platform_home*/jre などのパスを指定します。代わりに、IBM Unica 製品のこのリリースでの使用がサポートされている任意の Java ランタイム環境へのパスを指定することもできます。

runDeployment ユーティリティーのセキュア (SSL) 環境での使用

Interact サーバーでセキュリティーが有効になっているときに (従って、SSL ポートで接続するときに) runDeployment ユーティリティーを使用するには、次のようにしてトラストストア Java プロパティーを追加する必要があります。

1. 使用するインタラクティブ・チャネルの配置用の deployment.properties ファイルを編集する際に、deploymentURL プロパティーをセキュア SSL URL を使用するように変更します。このサンプルの場合、次のようになります。

deploymentURL=https://<*HOST*>.<*DOMAIN*>:<*PORT*>/Campaign/interact/InvokeDeploymentServlet

2. 任意のテキスト・エディターを使用して runDeployment.sh または runDeployment.bat スクリプトを編集して、${JAVA_HOME} で始まる行に次の引数を追加します。

   -Djavax.net.ssl.trustStore=<*TrustStorePath*>

   例えば、行にトラストストア引数を追加すると次のようになります。

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

   <*TrustStorePath*> を実際の SSL トラストストアへのパスに置き換えます。

ユーティリティーの実行

Java ランタイム環境を指定するようユーティリティーを編集し、環境に合わせて deployment.properties ファイルのコピーをカスタマイズしたら、次のコマンドでユーティリティーを実行できます。

Interact_home/interactDT/tools/deployment/runDeployment.sh *deployment.properties*

Interact_home を Interact 設計時インストール済み環境の実際の値に置き換え、*deployment.properties* をこの配置用にカスタマイズしたプロパティー・ファイルの実際のパスと名前に置き換えます。

サンプル deployment.properties ファイル

サンプル deployment.properties ファイルには、使用する環境に合わせてカスタマイズする必要があるすべてのパラメーターのコメント付きリストが含まれています。また、サンプル・ファイルには各パラメーターの解説と、その特定の値をカスタマイズする必要がある理由を説明するコメントが含まれています。

```
##############################################################################
#
# The following properties feed into the InvokeDeploymentClient program.
# The program will look for a deploymentURL setting.  The program will post a
# request against that url; all other settings are posted as parameters in
# that request. The program then checks the status of the deployment and
# returns back when the deployment is at a terminal state (or if the
# specified waitTime has been reached).
#
# the output of the program will be of this format:
# <STATE> : <Misc Detail>
#
# where state can be one of the following:
# ERROR
# RUNNING
# SUCCESS
#
# Misc Detail is data that would normally populate the status message area
#  in the deployment gui of the IC summary page. NOTE: HTML tags may exist
# in the Misc Detail
#
##############################################################################
```

```
################################################################################
# deploymentURL: url to the InvokeDeployment servlet that resides in Interact
# Design time.  should be in the following format:
# http://dt_host:port/Campaign/interact/InvokeDeploymentServlet
################################################################################
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet

################################################################################
# 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.
```

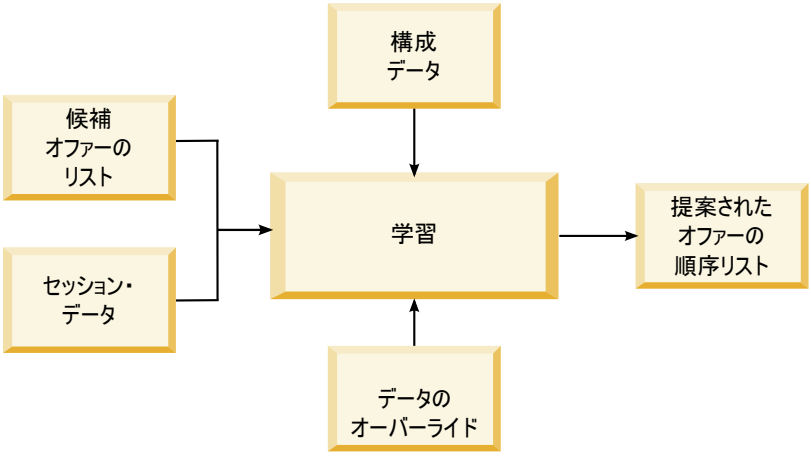
```
################################################################################
waitTime=5

################################################################################
# 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
```

第 10 章 学習 API について

Interact では、訪問者の操作をモニターし、(承認に) 最適なオファーを提案するために単純なベイズ・アルゴリズムを使用する学習モジュールを提供します。独自の学習モジュールを作成する場合は、学習 API を使用する独自のアルゴリズムを使用して同じ Java インターフェースを実装できます。

注: 外部の学習を使用する場合、学習に関するサンプル・レポート (「対話式オファー学習の詳細」レポートおよび「対話式セグメントの上昇分析」レポート) では有効なデータは返されません。

端的に言うと、学習 API はランタイム環境からデータを収集して、推奨オファーの番号付きリストを返すメソッドを提供します。



Interact から以下のデータを収集できます。

- オファー・コンタクト・データ
- データ承認データ
- すべてのセッション・データ
- Campaign 固有のオファー・データ
- 設計環境の**学習**カテゴリーおよびランタイム環境の**オファー配信**カテゴリーに定義されている構成プロパティー

ご使用のアルゴリズムでこのデータを使用して、提案されるオファーのリストを作成できます。その後、推奨順位の高い順に並べた推奨オファーのリストを返します。

図には示されていませんが、学習実装環境用のデータを収集する場合にも学習 API を使用することができます。このデータをメモリーに保持するか、ファイルまたはデータベースに記録して、さらに分析することができます。

Java クラスを作成したら、それらを JAR ファイルに変換できます。JAR ファイルを作成した場合は、構成プロパティーを編集して、外部学習モジュールを認識する

ようにランタイム環境を構成する必要もあります。Java クラスまたは JAR ファイルは、外部学習モジュールを使用するすべてのランタイム・サーバーにコピーする必要があります。

本書の情報に加え、ランタイム・サーバー上の Interact/docs/learningOptimizerJavaDoc ディレクトリーにある学習オプティマイザー API の JavaDoc を使用できます。

Interact ランタイム環境のインストール先の lib ディレクトリーにある interact_learning.jar に対して、実装アプリケーションをコンパイルする必要があります。

カスタム学習実装環境を作成する際には、以下のガイドラインに注意してください。

- パフォーマンスが重要である。
- マルチスレッドを使用する必要があり、スレッド・セーフである必要がある。
- 障害モードとパフォーマンスを考慮して、すべての外部リソースを管理する必要がある。
- 例外、ロギング (log4j)、およびメモリーを適切に使用する。

外部学習を有効にするには

学習 Java API を使用して、独自の学習モジュールを作成できます。Marketing Platform の学習ユーティリティーを認識するために、ランタイム環境を構成する必要があります。

ランタイム環境の Marketing Platform で、「Interact」>「offerserving」カテゴリーの以下の構成プロパティーを編集します。 Learning Optimizer API の構成プロパティーは、「Interact」>「offerserving」>「外部学習構成 (External Learning Config)」カテゴリーにあります。

構成プロパティー	設定
optimizationType	**ExternalLearning**
externalLearningClass	外部学習のクラス名
externalLearningClassPath	外部学習用のランタイム・サーバーのクラス・ファイルまたは JAR ファイルへのパス。サーバー・グループを使用していて、すべてのランタイム・サーバーが Marketing Platform の同じインスタンスを参照する場合、すべてのサーバーの同じ場所にクラス・ファイルまたは JAR ファイルのコピーを置く必要があります。

これらの変更を有効にするために、Interact ランタイム・サーバーを再始動する必要があります。

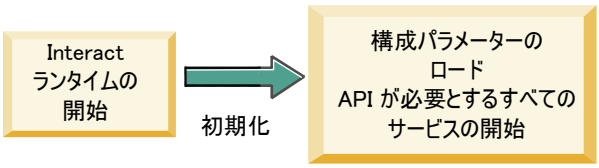
ILearning インターフェース

学習 API は、ILearning インターフェースを中心にして構築されています。学習モジュールのカスタマイズ・ロジックをサポートするために ILearning インターフェースを実装する必要があります。

特に、ILearning インターフェースを使用すると、Java クラス用にランタイム環境からデータを収集して、推奨するオファーのリストをランタイム・サーバーに送り返すことができます。

initialize

```
initialize(ILearningConfig config, boolean debug)
```



initialize メソッドは、ランタイム・サーバーの始動時に一度呼び出されます。繰り返す必要はないが、データベース・テーブルからの静的データの読み込みなど、実行時にパフォーマンスを低下させる可能性のある操作がある場合は、このメソッドで実行する必要があります。

- **config** — ILearningConfig オブジェクトは学習に関するすべての構成プロパティーを定義します。
- **debug** — ブール値。true の場合は、ランタイム環境システムのロギング詳細レベルが debug に設定されていることを示します。最良の結果を得るには、ログに書き込む前にこの値を選択します。

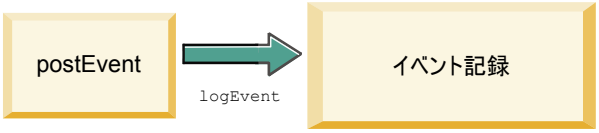
initialize メソッドがなんらかの理由で失敗した場合は、LearningException がスローされます。

戻り値

なし。

logEvent

```
logEvent(ILearningContext context,
              IOffer offer,
              IClientArgs clientArgs,
              IInteractSession session,
              boolean debug)
```



logEvent メソッドは、コンタクトまたレスポンスとしてログに記録されるように構成されているイベントが Interact API で通知されたときに、ランタイム・サーバーによって呼び出されます。このメソッドは、レポートおよび学習目的でデータベースまたはファイルにコンタクトおよびレスポンス・データをログ記録する場合に使用します。例えば、基準に基づいて顧客がオファーを受け入れる可能性を、アルゴリズムを使用して判断するには、このメソッドを使用してデータをログします。

- **context** — コンタクト、承認、拒否などの、イベントの学習コンテキストを定義する ILearningContext オブジェクト。
- **offer** — ログに記録されるイベントに関するオファーを定義する IOffer オブジェクト。
- **clientArgs** — すべてのパラメーターを定義する IClientArgs オブジェクト。現在、logEvent では clientArgs を必要としないため、このパラメーターは空である可能性があります。
- **session** — すべてのセッション・データを定義する IInteractSession オブジェクト。
- **debug** — ブール値。true の場合は、ランタイム環境システムのロギング詳細レベルが debug に設定されていることを示します。最良の結果を得るには、ログに書き込む前にこの値を選択します。

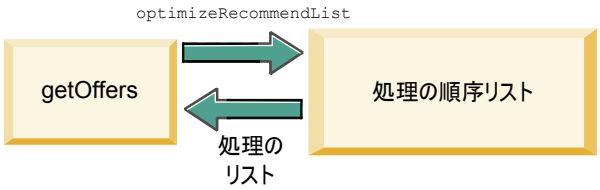
logEvent メソッドが失敗した場合、LearningException はスローされます。

戻り値

なし。

optimizeRecommendList

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```



optimizeRecommendList メソッドは、推奨されるオファーのリストとセッション・データを取り、オファーの要求数を含むリストを返す必要があります。
optimizeRecommendList メソッドは、ユーザー独自の学習アルゴリズムを使用して、なんらかの方法でオファーを配列する必要があります。オファーのリストは、最初に提供するオファーがリストの先頭になるように、配列する必要があります。例えば、学習アルゴリズムでベスト・オファーのスコアを低くした場合、オファーは 1、2、3 と配列されなければなりません。学習アルゴリズムでベスト・オファーのスコアを高くした場合、オファーは 100、99、98 と配列されなければなりません。

optimizeRecommendList メソッドには以下のパラメーターが必要です。

- **recList** — ランタイム環境で推奨される処理オブジェクト (オファー) のリスト。

- **clientArg** — ランタイム環境で要求される数以上のオファーを含む IClientArgs オブジェクト。
- **session** — すべてのセッション・データを含む IInteractSession オブジェクト。
- **debug** — ブール値。true の場合は、ランタイム環境システムのロギング詳細レベルが debug に設定されていることを示します。最良の結果を得るには、ログに書き込む前にこの値を選択します。

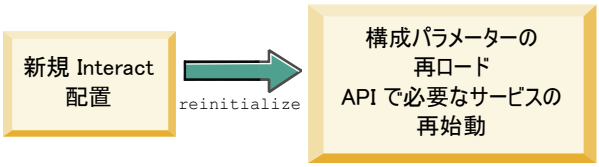
optimizeRecommendList メソッドが失敗した場合、LearningException はスローされます。

戻り値

optimizeRecommendList メソッドは ITreatment オブジェクトのリストを返します。

reinitialize

```
reinitialize(ILearningConfig config,
                  boolean debug)
```



ランタイム環境では、配置が新しくなるたびに reinitialize メソッドを呼び出します。このメソッドはすべての学習構成データを渡します。構成プロパティーを読み取る、学習 API で必要なサービスがある場合は、このインターフェースでサービスを再始動する必要があります。

- **config** — すべての構成プロパティーを含む ILearningConfig オブジェクト。
- **debug** — ブール値。true の場合は、ランタイム環境システムのロギング詳細レベルが debug に設定されていることを示します。最良の結果を得るには、ログに書き込む前にこの値を選択します。

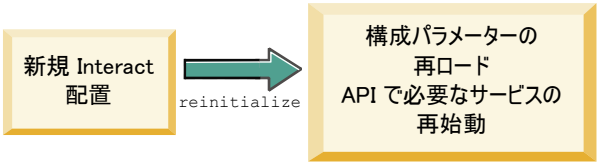
logEvent メソッドが失敗した場合、LearningException はスローされます。

戻り値

なし。

shutdown

```
shutdown(ILearningConfig config, boolean debug)
```



ランタイム環境では、ランタイム・サーバーのシャットダウン時に shutdown メソッドが呼び出されます。学習モジュールで必要なクリーンアップ・タスクがある場合は、この時点で実行する必要があります。

shutdown メソッドには以下のパラメーターが必要です。

- **config** — すべての構成プロパティーを定義する ILearningConfig オブジェクト。
- **debug** — ブール値。true の場合は、ランタイム環境システムのロギング詳細レベルが debug に設定されていることを示します。最良の結果を得るには、ログに書き込む前にこの値を選択します。

shutdown メソッドがなんらかの理由で失敗した場合は、LearningException がスローされます。

戻り値

なし。

IAudienceID インターフェース

IAudienceID インターフェースでは IInteractSession インターフェースがサポートされます。これは、オーディエンス ID のインターフェースです。オーディエンス ID は複数のパーツで構成されている場合があるため、このインターフェースを使用すれば、オーディエンス ID やオーディエンス・レベル名のすべての要素にアクセスできます。

getAudienceLevel

```
getAudienceLevel()
```

getAudienceLevel メソッドは、オーディエンス・レベルを返します。

戻り値

getAudienceLevel メソッドは、オーディエンス・レベルを定義する文字列を返します。

getComponentNames

```
getComponentNames()
```

getComponentNames メソッドは、オーディエンス ID を含む、コンポーネントの名前セットを取得します。例えば、オーディエンス ID が customerName および accountID の値から構成されている場合、getComponentNames は customerName と accountID の文字列を含めたセットを返します。

戻り値

オーディエンス ID のコンポーネントの名前を含む文字列セット。

getComponentValue

getComponentValue(String *componentName*)

getComponentValue メソッドは、指定されたコンポーネントの値を返します。

- **componentName** — 値を取得するコンポーネントの名前を定義する文字列。この文字列の大/小文字は区別されません。

戻り値

getComponentValue メソッドは、コンポーネントの値を定義するオブジェクトを返します。

IClientArgs

IClientArgs インターフェースでは ILearning インターフェースがサポートされます。このインターフェースは、セッション・データに対応していないタッチポイントからサーバーに渡されるデータを対応可能なものに抽象化します。例えば、Interact API getOffers メソッドによって、オファーの数が要求されたとします。このデータはマップに保管されます。

getValue

getValue(int *clientArgKey*)

getValue メソッドは、要求されたマップ要素の値を返します。

マップには以下の要素が必要です。

- **1** — NUMBER_OF_OFFERS_REQUESTED。Interact API の getOffers メソッドによって要求されるオファーの数。この定数は整数を返します。

戻り値

getValue メソッドは、要求されたマップ定数の値を定義するオブジェクトを返します。

IInteractSession

IInteractSession インターフェースでは ILearning インターフェースがサポートされます。これは、ランタイム環境の現行セッションのインターフェースです。

getAudienceId

getAudienceId()

getAudienceId メソッドは、AudienceID オブジェクトを返します。 IAudienceID インターフェースを使用して、値を抽出します。

戻り値

getAudienceId メソッドは、AudienceID オブジェクトを返します。

getSessionData

```
getSessionData()
```

getSessionData メソッドは、セッション変数の名前がキーである、セッション・データの変更できないマップを返します。セッション変数の名前は必ず大文字になります。IInteractSessionData インターフェースを使用して値を抽出します。

戻り値

getSessionData メソッドは、IInteractSessionData オブジェクトを返します。

IInteractSessionData インターフェース

IInteractSessionData インターフェースでは ILearning インターフェースがサポートされます。これは、現在の訪問者のランタイム・セッション・データのインターフェースです。セッション・データは名前と値のペアのリストとして保管されます。このインターフェースを使用して、ランタイム・セッションのデータ値を変更することもできます。

getDataType

```
getDataType(string parameterName)
```

getDataType メソッドは、指定されたパラメーター名のデータ型を返します。

戻り値

getDataType メソッドは、InteractDataType オブジェクトを返します。IntearctDataType は、Unknown、String、Double、Date、または List によって表される Java enum です。

getParameterNames

```
getParameterNames()
```

getParameterNames メソッドは、現在のセッションのデータのすべての名前セットを返します。

戻り値

getParameterNames メソッドは、値を設定済みの名前のセットを返します。セット内の名前をそれぞれ getValue(String) に渡して、値を返すことができます。

getValue

```
getValue(parameterName)
```

getValue メソッドは、指定された parameterName に対応するオブジェクト値を返します。オブジェクトは String、Double、または Date のいずれかです。

getValue メソッドには以下のパラメーターが必要です。

- **parameterName** — セッション・データの名前と値のペアの名前を定義する文字列。

戻り値

getValue メソッドは、指定されたパラメーターの値を含むオブジェクトを返します。

setValue

setValue(string *parameterName*, object *value*)

setValue メソッドを使用して、指定された parameterName の値を設定できます。値は String、Double、または Date のいずれかです。

setValue メソッドには以下のパラメーターが必要です。

- **parameterName** — セッション・データの名前と値のペアの名前を定義する文字列。
- **value** — 指定されたパラメーターの値を定義するオブジェクト。

戻り値

ありません。

ILearningAttribute

ILearningAttribute インターフェースでは ILearningConfig インターフェースがサポートされます。これは、learningAttributes カテゴリーの構成プロパティーに定義されている学習属性のインターフェースです。

getName

getName()

getName メソッドは、学習属性の名前を返します。

戻り値

getName メソッドは、学習属性の名前を定義する文字列を返します。

ILearningConfig

ILearningConfig インターフェースは ILearning インターフェースをサポートします。これは、学習の構成プロパティーに対するインターフェースです。以下のメソッドはすべて、プロパティーの値を返します。

このインターフェースは、以下の 15 メソッドで構成されています。

- **getAdditionalParameters** — 「外部学習構成 (External Learning Config)」カテゴリーで定義された追加プロパティーのマップを返します。
- **getAggregateStatsIntervalInMinutes** — 整数を返します。
- **getConfidenceLevel** — 整数を返します。
- **getDataSourceName** — ストリングを返します。
- **getDataSourceType** — ストリングを返します。

- **getInsertRawStatsIntervalInMinutes** — 整数を返します。
- **getLearningAttributes** — ILearningAttribute オブジェクトのリストを返します。
- **getMaxAttributeNames** — 整数を返します。
- **getMaxAttributeValues** — 整数を返します。
- **getMinPresentCountThreshold** — 整数を返します。
- **getOtherAttributeValue** — ストリングを返します。
- **getPercentRandomSelection** — 整数を返します。
- **getRecencyWeightingFactor** — 浮動小数を返します。
- **getRecencyWeightingPeriod** — 整数を返します。
- **isPruningEnabled** — ブールを返します。

ILearningContext

ILearningContext インターフェースは ILearning インターフェースをサポートします。

getLearningContext

```
getLearningContext()
```

getLearningContext メソッドは、これがコンタクト、承認、拒否のどのシナリオであるかを示す定数を返します。

- **1** — LOG_AS_CONTACT
- **2** — LOG_AS_ACCEPT
- **3** — LOG_AS_REJECT

4 と 5 は将来的な使用のために予約されています。

戻り値

getLearningContext メソッドは整数を返します。

getResponseCode

```
getResponseCode()
```

getResponseCode メソッドは、このオファーに割り当てられたレスポンス・コードを返します。この値は、Campaign システム・テーブルの UA_UsrResponseType テーブルに存在する必要があります。

戻り値

getResponseCode メソッドは、レスポンス・コードを定義する文字列を返します。

IOffer

IOffer インターフェースは ITreatment インターフェースをサポートします。これは、設計環境で定義されたオファー・オブジェクトに対するインターフェースです。IOffer インターフェースを使用して、ランタイム環境からオファーの詳細を収集します。

getCreateDate

```
getCreateDate()
```

getCreateDate メソッドは、オファーが作成された日付を返します。

戻り値

getCreateDate メソッドは、オファーが作成された日付を定義する日付を返します。

getEffectiveDateFlag

```
getEffectiveDateFlag()
```

getEffectiveDateFlag メソッドは、オファーの発効日を定義する数値を返します。

- **0** — 発効日は、2010 年 3 月 15 日など、絶対的な日付です。
- **1** — 発効日は、推奨の日付です。

戻り値

getEffectiveDateFlag メソッドは、オファーの発効日を定義する整数を返します。

getExpirationDateFlag

```
getExpirationDateFlag()
```

getExpirationDateFlag メソッドは、オファーの有効期限を示す整数値を返します。

- **0** — 2010 年 3 月 15 日などの絶対的な日付。
- **1** — 推奨後の日数 (14 など)。
- **2** — 推奨後の月末。オファーが 3 月 31 日に提示された場合、オファーはその当日に期限が切れます。

戻り値

getExpirationDateFlag メソッドは、オファーの有効期限を示す整数を返します。

getOfferAttributes

```
getOfferAttributes()
```

getOfferAttributes メソッドは、IOfferAttributes オブジェクトとしてオファー用に定義されたオファー属性を返します。

戻り値

getOfferAttributes メソッドは、IOfferAttributes オブジェクトを返します。

getOfferCode

getOfferCode()

getOfferCode メソッドは、Campaign で定義されているオファーのオファー・コードを返します。

戻り値

getOfferCode メソッドは、IOfferCode オブジェクトを返します。

getOfferDescription

getOfferDescription()

getOfferDescription メソッドは、Campaign で定義されているオファーの説明を返します。

戻り値

getOfferDescription メソッドは文字列を返します。

getOfferID

getOfferID()

getOfferID メソッドは、Campaign で定義されているオファー ID を返します。

戻り値

getOfferID メソッドは、オファー ID を定義する long を返します。

getOfferName

getOfferName()

getOfferName メソッドは、Campaign で定義されているオファーの名前を返します。

戻り値

getOfferName メソッドは文字列を返します。

getUpdateDate

getUpdateDate()

getUpdateDate メソッドは、オファーが最後に更新された日付を返します。

戻り値

getUpdateDate メソッドは、オファーが最後に更新されたときを定義する日付を返します。

IOfferAttributes

IOfferAttributes インターフェースは IOffer インターフェースをサポートします。これは、設計環境でオファー用に定義されたオファー属性に対するインターフェースです。IOfferAttributes インターフェースを使用して、ランタイム環境からオファー属性を収集します。

getParameterNames

getParameterNames()

getParameterNames メソッドは、オファーのパラメーター名のリストを返します。

戻り値

getParameterNames メソッドは、オファーのパラメーター名のリストを定義するセットを返します。

getValue

getValue(String *parameterName*)

getValue メソッドは、指定されたオファー属性の値を返します。

戻り値

getValue メソッドは、オファー属性の値を定義するオブジェクトを返します。

IOfferCode インターフェース

IOfferCode インターフェースは ILearning インターフェースをサポートします。これは、設計環境でオファー用に定義されたオファー・コードに対するインターフェースです。オファー・コードは、1 対多のストリングで構成可能です。IOfferCode インターフェースを使用して、ランタイム環境からオファー・コードを収集します。

getPartCount

getPartCount()

getPartCount メソッドは、オファー・コードを作成する部分の数を返します。

戻り値

getPartCount メソッドは、オファー・コードの部分の数を定義する整数を返します。

getParts

```
getParts()
```

getParts メソッドは、オファー・コード部分の変更できないリストを取得します。

戻り値

getParts メソッドは、オファー・コード部分の変更できないリストを返します。

LearningException

LearningException クラスは ILearning インターフェースをサポートします。インターフェース内の一部のメソッドでは、java.lang.Exception の単純なサブクラスである LearningException をスローするための実装が必要です。ルート例外が存在する場合は、デバッグ目的のために、LearningException をルート例外で構成することが強く推奨されます。

IScoreOverride

IScoreOverride インターフェースは ITreatment インターフェースをサポートします。このインターフェースを使用すると、スコア・オーバーライド・テーブルまたはデフォルト・オファー・テーブルで定義されたデータを読み取ることができます。

getOfferCode

```
getOfferCode()
```

getOfferCode メソッドは、このオーディエンス・メンバーのスコア・オーバーライド・テーブルのオファー・コード列の値を返します。

戻り値

getOfferCode メソッドは、スコア・オーバーライド・テーブルのオファー・コード列の値を定義する IOfferCode オブジェクトを返します。

getParameterNames

```
getParameterNames()
```

getParameterNames メソッドは、パラメーターのリストを返します。

戻り値

getParameterNames メソッドは、パラメーターのリストを定義するセットを返します。

IScoreOverride メソッドには以下のパラメーターが含まれています。特に示されない限り、これらのパラメーターはスコア・オーバーライド・テーブルと同じです。

- ADJ_EXPLORE_SCORE_COLUMN
- CELL_CODE_COLUMN

- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT — 現在の推定カウントをオーバーライドする場合 (オファーの重み計算時)
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN — オファーの資格を決定するブール式を作成する場合
- PREDICATE_SCORE — 数値スコアを求める式を作成する場合
- SCORE_COLUMN
- ZONE_COLUMN

列と同じ名前を使用して、スコア・オーバーライド・テーブルまたはデフォルト・オファー・テーブルに追加する列を参照することもできます。

getValue

`getValue(String parameterName)`

`getValue` メソッドは、このオーディエンス・メンバーのスコア・オーバーライド・テーブルのゾーン列の値を返します。

- **parameterName** — 値を求めるパラメーターの名前を定義する文字列。

戻り値

`getValue` メソッドは、要求されたパラメーターの値を定義するオブジェクトを返します。

ISelectionMethod

`ISelection` インターフェースは、推奨リスト作成のために使用されるメソッドを示します。処理オブジェクトのデフォルト値は EXTERNAL_LEARNING であるため、この値を設定する必要はありません。値は、レポート目的のため、最終的に詳細コンタクト履歴に保管されます。

後の分析で使用するためにデータを保管する場合は、既存の定数を超えてこのインターフェースを拡張できます。例えば、2 つの異なる学習モジュールを作成して、それらを個別のサーバー・グループに実装できます。`ISelection` インターフェースを拡張して、SERVER_GROUP_1 と SERVER_GROUP_2 を組み込むことができます。その後、2 つの学習モジュールの結果を比較できます。

ITreatment インターフェース

`ITreatment` インターフェースは、処理情報へのインターフェースとして `ILearning` インターフェースをサポートします。処理では、設計環境で定義された特定のセルに割り当てられたオファーが表されます。このインターフェースから、割り当てられたマーケティング・スコアだけでなく、セル情報とオファー情報を取得できます。

getCellCode

```
getCellCode()
```

getCellCode メソッドは、Campaign で定義されているセル・コードを返します。セルは、このオファーに関連付けられたスマート・セグメントに割り当てられるセルです。

戻り値

getCellCode メソッドは、セル・コードを定義する文字列を返します。

getCellId

```
getOfferName()
```

getCellId メソッドは、Campaign に定義されているセルの内部 ID を返します。セルは、このオファーに関連付けられたスマート・セグメントに割り当てられるセルです。

戻り値

getCellId メソッドは、セル ID を定義する long を返します。

getCellName

```
getCellName()
```

getCellName メソッドは、Campaign で定義されているセルの名前を返します。セルは、このオファーに関連付けられたスマート・セグメントに割り当てられるセルです。

戻り値

getCellName メソッドは、セル名を定義する文字列を返します。

getLearningScore

```
getLearningScore()
```

getLearningScore メソッドは、この処理のスコアを返します。優先順位は次のとおりです。

1. IScoreoveride.PREDICATE_SCORE_COLUMN によってキー付けされたオーバーライド値のマップに存在する場合は、オーバーライド値を返します。
2. 値が NULL でない場合は、述語スコアを返します。
3. IScoreoveride.SCORE によってキー付けされたオーバーライド値のマップに存在する場合は、マーケティング担当者のスコアを返します。
4. マーケティング担当者のスコアを返します。

戻り値

getLearningScore メソッドは、学習アルゴリズムによって決まるスコアを定義する整数を返します。

getMarketerScore

```
getMarketerScore()
```

getMarketerScore メソッドは、オファーのインタラクション方法タブにあるスライダーで定義するマーケティング担当者のスコアを返します。

インタラクション方法タブの詳細オプションによって定義されるマーケティング担当者のスコアを取得するには、getPredicateScore を使用します。

実際に処理で使用されるマーケティング担当者のスコアを取得するには、getLearningScore を使用します。

戻り値

getMarketerScore メソッドは、マーケティング担当者のスコアを定義する整数を返します。

getOffer

```
getOffer()
```

getOffer メソッドは、処理のオファーを返します。

戻り値

getOffer メソッドは、この処理のオファーを定義する IOffer オブジェクトを返します。

getOverrideValues

```
getOverrideValues()
```

getOverrideValues メソッドは、デフォルト・オファー・テーブルまたはスコア・オーバーライド・テーブルで定義されたオーバーライドを返します。

戻り値

getOverrideValues メソッドは、IScoreOverride オブジェクトを返します。

getPredicate

```
getPredicate()
```

getPredicate メソッドは、デフォルト・オファー・テーブルの述語列、スコア・オーバーライド・テーブルの述語列、または処理ルールの詳細オプションで定義された述語を返します。

戻り値

getPredicate メソッドは、デフォルト・オファー・テーブルの述語列、スコア・オーバーライド・テーブルの述語列、または処理ルールの詳細オプションで定義された述語を定義する文字列を返します。

getPredicateScore

```
getPredicateScore()
```

getPredicateScore メソッドは、デフォルト・オファー・テーブルの述語列、スコア・オーバーライド・テーブルの述語列、または処理ルールの詳細オプションによって設定されたスコアを返します。

戻り値

getPredicateScore メソッドは、デフォルト・オファー・テーブルの述語列、スコア・オーバーライド・テーブルの述語列、または処理ルールの詳細オプションによって設定されたスコアを定義する double を返します。

getScore

```
getScore()
```

getScore メソッドは、次のうちのいずれか 1 つを返します。

- enableScoreOverrideLookup プロパティーが false に設定されている場合、Campaign のインタラクション方法タブで定義されているオファーのマーケティング・スコア。
- enableScoreOverrideLookup プロパティーが true に設定されている場合、scoreOverrideTable で定義されているオファーのスコア。

戻り値

getScore メソッドは、オファーのスコアを示す整数を返します。

getTreatmentCode

```
getTreatmentCode()
```

getTreatmentCode メソッドは処理コードを返します。

戻り値

getTreatmentCode メソッドは、処理コードを定義する文字列を返します。

setActualValueUsed

```
setActualValueUsed(string parmName, object value)
```

setActualValueUsed メソッドを使用して、学習アルゴリズムの実行でさまざまなステージで使用する値を定義します。

例えば、このメソッドを使用してコンタクトおよびレスポンスの履歴テーブルに書き込む場合に、既存のサンプル・レポートを変更するときは、レポートの学習アルゴリズムからのデータを含めることができます。

- **parmName** — 設定しているパラメーターの名前を定義する文字列。
- **value** — 設定しているパラメーターの値を定義するオブジェクト。

戻り値

ありません。

学習 API の例

このセクションには、ILearningInterface の実装例が含まれています。この実装は単なる例であり、実稼働環境で使用するためのものではないことに注意してください。

以下の例では、受け入れおよびコンタクトの件数をトラッキングし、特定のオファーの受け入れとコンタクトの比率をオファーの受け入れ確立比として使用します。例には示されていませんが、より優先度の高い推奨オファーがあります。少なくとも 1 つのコンタクトを含むオファーが、降順の受け入れ確立比に基づいて配列されています。

以下の例では、すべての件数がメモリー内に保持されています。これは、ランタイム・サーバーでメモリー不足が発生するため、現実的なシナリオではありません。In a real production scenario, the counts should be persisted into a database.

```
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 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#optimizeRecommendList
 *   (java.util.List, com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
 *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
 */
```

```
    public List<ITreatment> optimizeRecommendList(List<ITreatment> recList,
        IClientArgs clientArgs, IInteractSession session, boolean debug)
        throws LearningException
    {
        System.out.println("Calling optimizeRecommendList for SampleLearning");

        // Sort the candidate treatments by calling the sorter defined in this class and return the sorted list
        Collections.sort(recList,new MyOfferSorter());

        // now just return what was asked for via "numberRequested" variable
        List<ITreatment> result = new ArrayList<ITreatment>();

        for(int x=0;x<(Integer)clientArgs.getValue(IClientArgs.NUMBER_OF_OFFERS_REQUESTED) && x<recList.size();x++)
        {
            result.add(recList.get(x));
        }
        return result;
    }

    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#shutdown
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void shutdown(ILearningConfig config, boolean debug) throws LearningException
    {
        // If any remote connections exist, this would be a good place to gracefully
        // disconnect from them as this method is called at the shutdown of the Interact runtime
        // webapp.   For this example, there is nothing really to do
        // except print out a statement for debugging.
        System.out.println("Calling shutdown for SampleLearning");

    }
    // Sort by:
    // 1. offers with zero contacts  - for ties, order is based on original input
    // 2. descending accept probability rate - for ties, order is based on original input

    public class MyOfferSorter implements Comparator<ITreatment>
    {
        private static final long serialVersionUID = 1L;

        /* (non-Javadoc)
         * @see java.lang.Comparable#compareTo(java.lang.Object)
         */
        public int compare(ITreatment treatment1, ITreatment treatment2)
        {

            // get contact count for both treatments
            Integer contactCount1 = _offerToContactCount.get(treatment1.getOffer().getOfferId());
            Integer contactCount2 = _offerToContactCount.get(treatment2.getOffer().getOfferId());

            // if treatment hasn't been contacted, then that wins
            if(contactCount1 == null || contactCount1 == 0)
                return -1;

            if(contactCount2 == null || contactCount2 == 0)
                return 1;

            // get accept counts
            Integer acceptCount1 = _offerToAcceptCount.get(treatment1.getOffer().getOfferId());
            Integer acceptCount2 = _offerToAcceptCount.get(treatment2.getOffer().getOfferId());

            float acceptProbability1 = (float) acceptCount1 / (float) contactCount1;
            float acceptProbability2 = (float) acceptCount2 / (float) contactCount2;

            // descending order
            return (int) (acceptProbability2 - acceptProbability1);

        }
    }

}
```

付録 A. IBM Unica Interact WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://soap.api.interact.unicacorp.com" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" bloop="http://api.interact.unicacorp.com/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" targetNamespace="http://soap.api.interact.unicacorp.com">
 <wsdl:types>
  <xs:schema xmlns:ns="http://soap.api.interact.unicacorp.com" attributeFormDefault="qualified"
   elementFormDefault="qualified" targetNamespace="http://soap.api.interact.unicacorp.com">
   <xs:element name="executeBatch">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="commands" nillable="false" type="ns1:CommandImpl"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="executeBatchResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:BatchResponse"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="endSession">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="endSessionResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getOffers">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
      <xs:element minOccurs="1" name="iPoint" nillable="false" type="xs:string"/>
      <xs:element minOccurs="1" name="numberRequested" type="xs:int"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getOffersResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getProfile">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="sessionID" nillable="false" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getProfileResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="getVersionResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element minOccurs="1" name="return" nillable="false" type="ns1:Response"/>
```

```
    </xs:sequence>
   </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>
```

付録 B. Interact ランタイム環境の構成プロパティー

このセクションでは、Interact ランタイム環境のすべての構成プロパティーについて説明します。

Interact | 全般

これらの構成プロパティーは、ランタイム環境の一般的な設定を定義します。これには、デフォルトのロギング・レベルやロケールの設定が含まれます。

log4jConfig

説明

log4j プロパティーが含まれているファイルのロケーション。これは、INTERACT_HOME 環境変数からの相対パスにする必要があります。INTERACT_HOME は、Interact のインストール・ディレクトリーのロケーションです。

既定値

./conf/interact_log4j.properties

asmUserForDefaultLocale

説明

asmUserForDefaultLocale プロパティーが定義する IBM Unica Marketing ユーザーから、Interact はそのロケール設定を派生させます。

ロケール設定は、設計時の表示をどの言語で行うか、および Interact API からのアドバイス・メッセージをどの言語で表示するかを定義します。ロケール設定がそのマシンのオペレーティング・システムの設定と一致しない場合でも Interact は機能しますが、設計時の表示やアドバイス・メッセージは、別の言語で表示される可能性があります。

既定値

既定値が定義されていません。

Interact | 全般 | learningTablesDataSource

これらの構成プロパティーは、組み込み学習テーブルのデータ・ソースの設定を定義します。Interact の組み込み学習を使用する場合は、このデータ・ソースを定義する必要があります。

学習 API を使用して独自の学習実装環境を作成する場合は、カスタムの学習実装環境を構成し、ILearningConfig インターフェースを使用してそれらの値を読み取らせることができます。

jndiName

説明

この jndiName プロパティーを使用して、アプリケーション・サーバー (Websphere または WebLogic) で Interact ランタイム・サーバーがアクセスする学習テーブル用に定義されている Java Naming and Directory Interface (JNDI) データ・ソースを識別します。

学習テーブルは aci_lrntab ddl ファイルによって作成されます。これには、UACI_AttributeValue や UACI_OfferStats などのテーブルが含まれます。

既定値

既定値が定義されていません。

タイプ

説明

Interact ランタイム・サーバーがアクセスする学習テーブルによって使用されるデータ・ソースのデータベース・タイプ。

学習テーブルは aci_lrntab ddl ファイルによって作成されます。これには、UACI_AttributeValue や UACI_OfferStats などのテーブルが含まれます。

既定値

SQLServer

有効な値

SQLServer | DB2 | ORACLE

connectionRetryPeriod

説明

ConnectionRetryPeriod プロパティーは、学習テーブルへのデータベース接続要求が失敗した場合に、Interact によって自動的に再試行される時間を秒単位で指定します。Interact は、この長さの時間、データベースへの再接続を自動的に試行してから、データベース・エラーまたは失敗を報告します。この値を 0 に設定すると、Interact は無制限に再試行します。この値を -1 に設定すると、再試行は行われません。

学習テーブルは aci_lrntab ddl ファイルによって作成されます。これには、UACI_AttributeValue や UACI_OfferStats などのテーブルが含まれます。

既定値

-1

connectionRetryDelay

説明

ConnectionRetryDelay プロパティーは、Interact が学習テーブルへのデータベース接続に失敗した場合に、再接続を試行するまでの待ち時間を秒数で指定します。この値を -1 に設定すると、再試行は行われません。

学習テーブルは aci_lrntab ddl ファイルによって作成されます。これには、UACI_AttributeValue や UACI_OfferStats などのテーブルが含まれます。

既定値

-1

スキーマ

説明

組み込み学習モジュールのテーブルが含まれているスキーマの名前。Interact は、このプロパティーの値をすべてのテーブル名の前に挿入します。例えば、UACI_IntChannel は schema.UACI_IntChannel になります。

スキーマを定義する必要はありません。スキーマを定義しない場合、Interact は、テーブルの所有者はスキーマと同じであると想定します。あいまいさを排除するには、この値を設定してください。

既定値

既定値が定義されていません。

Interact | 全般 | prodUserDataSource

これらの構成プロパティーは、運用プロファイル・テーブルのデータ・ソースの設定を定義します。このデータ・ソースは定義する必要があります。これは、配置したインタラクティブ・フローチャートを実行する際に、ランタイム環境が参照するデータ・ソースです。

jndiName

説明

この jndiName プロパティーを使用して、アプリケーション・サーバー (Websphere または WebLogic) で Interact ランタイム・サーバーがアクセスする顧客テーブル用に定義されている Java Naming and Directory Interface (JNDI) データ・ソースを識別します。

既定値

既定値が定義されていません。

タイプ

説明

Interact ランタイム・サーバーがアクセスする顧客テーブルのデータベース・タイプ。

既定値

SQLServer

有効な値

SQLServer | DB2 | ORACLE

aliasPrefix

説明

AliasPrefix プロパティーは、ディメンション・テーブルを使用していて、Interact ランタイム・サーバーがアクセスする顧客テーブルに新しいテーブルを書き込む際に、Interact により自動的に作成される別名を、Interact がどのように形成するかを指定します。

各データベースには、それぞれ ID の最大長があります。使用しているデータベースのドキュメンテーションを調べて、設定する値がデータベースの最大 ID 長を超えないものであることを確認してください。

既定値

A

connectionRetryPeriod

説明

ConnectionRetryPeriod プロパティーは、ランタイム顧客テーブルへのデータベース接続要求が失敗した場合に、Interact によって自動的に再試行される時間を秒単位で指定します。Interact は、この長さの時間、データベースへの再接続を自動的に試行してから、データベース・エラーまたは失敗を報告します。この値を 0 に設定すると、Interact は無制限に再試行します。この値を -1 に設定すると、再試行は行われません。

既定値

-1

connectionRetryDelay

説明

ConnectionRetryDelay プロパティーは、Interact が Interact ランタイム顧客テーブルへのデータベース接続に失敗した場合に、再接続を試行するまでの待ち時間を秒数で指定します。この値を -1 に設定すると、再試行は行われません。

既定値

-1

スキーマ

説明

プロファイル・データ・テーブルが含まれているスキーマの名前。Interact は、このプロパティーの値をすべてのテーブル名の前に挿入します。例えば、UACI_IntChannel は schema.UACI_IntChannel になります。

スキーマを定義する必要はありません。スキーマを定義しない場合、Interact は、テーブルの所有者はスキーマと同じであると想定します。あいまいさを排除するには、この値を設定してください。

既定値

既定値が定義されていません。

Interact | 全般 | systemTablesDataSource

これらの構成プロパティーは、ランタイム環境用システム・テーブルのデータ・ソースの設定を定義します。このデータ・ソースは定義する必要があります。

jndiName

説明

この jndiName プロパティーを使用して、アプリケーション・サーバー (Websphere または WebLogic) でランタイム環境テーブル用に定義されている Java Naming and Directory Interface (JNDI) データ・ソースを識別します。

ランタイム環境データベースは、aci_runtime および aci_populate_runtime の各 dll スクリプトが取り込まれたデータベースで、例えば UACI_CHOfferAttrib や UACI_DefaultedStat などが含まれます。

既定値

既定値が定義されていません。

タイプ

説明

ランタイム環境のシステム・テーブルのデータベース・タイプ。

ランタイム環境データベースは、aci_runtime および aci_populate_runtime の各 dll スクリプトが取り込まれたデータベースで、例えば UACI_CHOfferAttrib や UACI_DefaultedStat などが含まれます。

既定値

SQLServer

有効な値

SQLServer | DB2 | ORACLE

connectionRetryPeriod

説明

ConnectionRetryPeriod プロパティーは、ランタイム・システム・テーブルへのデータベース接続要求が失敗した場合に、Interact によって自動的に再試行される時間を秒単位で指定します。Interact は、この長さの時間、データベースへの再接続を自動的に試行してから、データベース・エラーまたは失敗を報告します。この値を 0 に設定すると、Interact は無制限に再試行します。この値を -1 に設定すると、再試行は行われません。

ランタイム環境データベースは、aci_runtime および aci_populate_runtime の各 dll スクリプトが取り込まれたデータベースで、例えば UACI_CHOfferAttrib や UACI_DefaultedStat などが含まれます。

既定値

-1

connectionRetryDelay

説明

ConnectionRetryDelay プロパティーは、Interact が Interact ランタイム・システム・テーブルへのデータベース接続に失敗した場合に、再接続を試行するまでの待ち時間を秒数で指定します。この値を -1 に設定すると、再試行は行われません。

ランタイム環境データベースは、aci_runtime および aci_populate_runtime の各 dll スクリプトが取り込まれたデータベースで、例えば UACI_CHOfferAttrib や UACI_DefaultedStat などが含まれます。

既定値

-1

スキーマ

説明

ランタイム環境のテーブルが含まれているスキーマの名前。Interact は、このプロパティーの値をすべてのテーブル名の前に挿入します。例えば、UACI_IntChannel は schema.UACI_IntChannel になります。

スキーマを定義する必要はありません。スキーマを定義しない場合、Interact は、テーブルの所有者はスキーマと同じであると想定します。あいまいさを排除するには、この値を設定してください。

既定値

既定値が定義されていません。

Interact | 全般 | systemTablesDataSource | loaderProperties

これらの構成プロパティーは、ランタイム環境用システム・テーブルのデータベース・ローダー・ユーティリティーの設定を定義します。データベース・ローダー・ユーティリティーのみを使用している場合は、これらのプロパティーを定義する必要があります。

databaseName

説明

データベース・ローダーが接続するデータベースの名前。

既定値

既定値が定義されていません。

LoaderCommandForAppend

説明

LoaderCommandForAppend パラメーターは、Interact において、データベース・ロード・ユーティリティーを起動して、コンタクトとレスポンスの履歴

ステージング・データベース・テーブルにレコードを付加するために発行するコマンドを指定します。コンタクトとレスポンスの履歴データに対してデータベース・ローダー・ユーティリティーを使用可能にするには、このパラメーターを設定する必要があります。

このパラメーターは、データベース・ロード・ユーティリティーの実行可能ファイルまたはデータベース・ロード・ユーティリティーを起動するスクリプトの絶対パス名として指定します。スクリプトを使用することで、ロード・ユーティリティーを呼び出す前に、追加のセットアップを実行することができます。

ほとんどのデータベース・ロード・ユーティリティーでは、正常に起動するために複数の引数が必要です。その中には、ロード元となるデータ・ファイルと制御ファイル、およびロード先となるデータベースとテーブルを指定するものが含まれることがあります。コマンドが実行されると、指定された要素によってトークンが置換されます。

データベース・ロード・ユーティリティー呼び出しで使用する正しい構文については、データベース・ロード・ユーティリティーのドキュメンテーションを参照してください。

このパラメーターは、既定では未定義です。

`LoaderCommandForAppend` で使用可能なトークンについて、以下の表で説明します。

トークン	**説明**
<CONTROLFILE>	このトークンは、`LoaderControlFileTemplate` パラメーターで指定されるテンプレートに従って、Interact によって生成される一時制御ファイルの絶対パスとファイル名に置換されます。
<DATABASE>	このトークンは、Campaign がデータをロードする先のデータ・ソースの名前に置換されます。これは、このデータ・ソースのカテゴリー名で使用されるのと同じデータ・ソース名です。
<DATAFILE>	このトークンは、ロード・プロセスで Interact によって作成される一時データ・ファイルの絶対パスとファイル名に置換されます。このファイルは、Interact 一時ディレクトリー `UNICA_ACTMPDIR` に入っています。
<DBCOLUMNNUMBER>	このトークンは、データベース中の列順序に置換されます。
<FIELDLENGTH>	このトークンは、データベース中にロードされている項目の長さに置換されます。

トークン	説明
<FIELDNAME>	このトークンは、データベース中にロードされている項目の名前に置換されます。
<FIELDNUMBER>	このトークンは、データベース中にロードされている項目の番号に置換されます。
<FIELDTYPE>	このトークンは、リテラル CHAR( ) に置換されます。 () の中により、このフィールドの長さが指定されます。データベースでフィールド・タイプ CHAR が認識されていない場合、フィールド・タイプとして適切なテキストを手動で指定し、<FIELDLENGTH> トークンを使用することができます。例えば、SQLSVR および SQL2000 の場合、SQLCHAR(<FIELDLENGTH>) を使用します。
<NATIVETYPE>	このトークンは、このフィールドのロード先であるデータベースのタイプに置換されます。
<NUMFIELDS>	このトークンは、テーブル中の項目の数に置換されます。
<PASSWORD>	このトークンは、現在のフローチャートからデータ・ソースへの接続のデータベース・パスワードに置換されます。
<TABLENAME>	このトークンは、Campaign がデータをロードする先のデータベース・テーブル名に置換されます。
<USER>	このトークンは、現在のフローチャート接続からデータ・ソースへのデータベース・ユーザーに置換されます。

既定値

既定値が定義されていません。

LoaderControlFileTemplateForAppend

説明

LoaderControlFileTemplateForAppend プロパティーは、Interact で以前に構成された制御ファイル・テンプレートの絶対パスとファイル名を指定します。このパラメーターが設定されている場合、Interact は、ここで指定されているテンプレートに基づいて一時制御ファイルを動的に作成します。この一時制御ファイルのパスおよび名前は、LoaderCommandForAppend プロパティーから利用可能な <CONTROLFILE> トークンから利用可能です。

Interact をデータベース・ローダー・ユーティリティー・モードで使用するには、その前に、このパラメーターによって指定される制御ファイル・テンプレートを構成することが必要です。制御ファイル・テンプレートでは、以下のトークンがサポートされています。それらは、Interact によって一時制御ファイルが作成される際に動的に置換されます。

制御ファイルで必要な正しい構文については、データベース・ローダー・ユーティリティーのドキュメンテーションを参照してください。制御ファイル・テンプレートで利用可能なトークンは、LoaderControlFileTemplate プロパティーのものと同じです。

このパラメーターは、既定では未定義です。

既定値

既定値が定義されていません。

LoaderDelimiterForAppend

説明

LoaderDelimiterForAppend プロパティーは、Interact の一時データ・ファイルが固定幅であるか、それとも区切りフラット・ファイルであるかを指定します。また、区切りファイルの場合には、区切りとして使用する文字または文字の集合を指定します。

値が未定義の場合、Interact は、固定幅フラット・ファイルとして一時データ・ファイルを作成します。

値を指定する場合、それは、ローダーが呼び出された時点で、空であるとは認識されていないテーブルのデータを設定するために使用されます。Interact は、このプロパティーの値を区切り文字として使用することにより、区切りフラット・ファイルとして一時データ・ファイルを作成します。

このプロパティーは、既定では未定義です。

既定値

有効な値

文字 (必要に応じて二重引用符で囲むことが可能)。

LoaderDelimiterAtEndForAppend

説明

一部の外部ロード・ユーティリティーでは、データ・ファイルを区切る必要があります。また、各行は区切り文字で終わる必要があります。この要件を満たすためには、LoaderDelimiterAtEndForAppend の値を TRUE に設定することにより、ローダーが起動して、空として認識されてはいないテーブルのデータを設定する際に、Interact が各行の末尾に区切り文字を使用するようにします。

既定値

FALSE

有効な値

TRUE | FALSE

LoaderUseLocaleDP

説明

LoaderUseLocaleDP プロパティーは、Interact が、データベース・ロード・ユーティリティーによってロードされるファイルに数値を書き込む際に、小数点としてロケール固有の記号を使用するかどうかを指定します。

ピリオド (.) を小数点として指定するには、この値を FALSE に設定します。

ロケールにふさわしい小数点記号を使用することを指定するには、この値を TRUE に設定します。

既定値

FALSE

有効な値

TRUE | FALSE

Interact | 全般 | testRunDataSource

これらの構成プロパティーは、Interact 設計環境用テスト実行テーブルのデータ・ソースの設定を定義します。使用するランタイム環境の最低 1 つで、このデータ・ソースを定義する必要があります。これらは、インタラクティブ・フローチャートのテスト実行を行う際に使用されるテーブルです。

jndiName

説明

この jndiName プロパティーを使用して、設計環境でインタラクティブ・フローチャートのテスト実行を行う際にアクセスする顧客テーブル用にアプリケーション・サーバー (Websphere または WebLogic) で定義されている、Java Naming and Directory Interface (JNDI) データ・ソースを識別します。

既定値

既定値が定義されていません。

タイプ

説明

設計環境でインタラクティブ・フローチャートのテスト実行を行う際にアクセスする顧客テーブルのデータベース・タイプ

既定値

SQLServer

有効な値

SQLServer | DB2 | ORACLE

aliasPrefix

説明

AliasPrefix プロパティーは、ディメンション・テーブルを使用していて、設計環境がインタラクティブ・フローチャートのテスト実行を行うときにアクセスする顧客テーブルに新しいテーブルを書き込む際に、Interact により自動的に作成される別名を、Interact がどのように形成するかを指定します。

各データベースには、それぞれ ID の最大長があります。使用しているデータベースのドキュメンテーションを調べて、設定する値がデータベースの最大 ID 長を超えないものであることを確認してください。

既定値

A

connectionRetryPeriod

説明

ConnectionRetryPeriod プロパティーは、テスト実行テーブルへのデータベース接続要求が失敗した場合に、Interact によって自動的に再試行される時間を秒単位で指定します。Interact は、この長さの時間、データベースへの再接続を自動的に試行してから、データベース・エラーまたは失敗を報告します。この値を 0 に設定すると、Interact は無制限に再試行します。この値を -1 に設定すると、再試行は行われません。

既定値

-1

connectionRetryDelay

説明

ConnectionRetryDelay プロパティーは、Interact がテスト実行テーブルへのデータベース接続に失敗した場合に、再接続を試行するまでの待ち時間を秒数で指定します。この値を -1 に設定すると、再試行は行われません。

既定値

-1

スキーマ

説明

インタラクティブ・フローチャートのテスト実行のテーブルが含まれているスキーマの名前。Interact は、このプロパティーの値をすべてのテーブル名の前に挿入します。例えば、UACI_IntChannel は schema.UACI_IntChannel になります。

スキーマを定義する必要はありません。スキーマを定義しない場合、Interact は、テーブルの所有者はスキーマと同じであると想定します。あいまいさを排除するには、この値を設定してください。

既定値

既定値が定義されていません。

Interact | 全般 | contactAndResponseHistoryDataSource

これらの構成プロパティーは、Interact のクロス・セッション・レスポンス・トラッキングに必要なコンタクトとレスポンスの履歴データ・ソースの接続設定を定義します。

これらの設定は、コンタクトとレスポンスの履歴モジュールとは関係ありません。

jndiName

説明

この jndiName プロパティーを使用して、アプリケーション・サーバー (WebSphere または WebLogic) で定義されている、Interact のクロス・セッション・レスポンス・トラッキングに必要なコンタクトとレスポンスの履歴データ・ソース用の Java Naming and Directory Interface (JNDI) データ・ソースを識別します。

既定値

タイプ

説明

Interact のクロス・セッション・レスポンス・トラッキングに必要なコンタクトとレスポンスの履歴データ・ソースによって使用されるデータ・ソースのデータベース・タイプ。

既定値

SQLServer

有効な値

SQLServer | DB2 | ORACLE

connectionRetryPeriod

説明

ConnectionRetryPeriod プロパティーは、Interact のクロス・セッション・レスポンス・トラッキングへのデータベース接続要求が失敗した場合に、Interact によって自動的に再試行される時間を秒単位で指定します。Interact は、この長さの時間、データベースへの再接続を自動的に試行してから、データベース・エラーまたは失敗を報告します。この値を 0 に設定すると、Interact は無制限に再試行します。この値を -1 に設定すると、再試行は行われません。

既定値

-1

connectionRetryDelay

説明

ConnectionRetryDelay プロパティーは、Interact が Interact のクロス・セッション・レスポンス・トラッキングへのデータベース接続に失敗した場合に、再接続を試行するまでの待ち時間を秒数で指定します。この値を -1 に設定すると、再試行は行われません。

既定値

-1

スキーマ

説明

Interact のクロス・セッション・レスポンス・トラッキングのテーブルが含まれているスキーマの名前。Interact は、このプロパティーの値をすべてのテーブル名の前に挿入します。例えば、UACI_IntChannel は schema.UACI_IntChannel になります。

スキーマを定義する必要はありません。スキーマを定義しない場合、Interact は、テーブルの所有者はスキーマと同じであると想定します。あいまいさを排除するには、この値を設定してください。

既定値

既定値が定義されていません。

Interact | 全般 | idsByType

これらの構成プロパティーは、コンタクトとレスポンスの履歴モジュールで使用される ID 番号の設定を定義します。

initialValue

説明

UACI_IDsByType テーブルを使用して ID を生成するときに使用される、ID の初期値。

既定値

1

有効な値

0 より大きい任意の値。

再試行 (retries)

説明

UACI_IDsByType テーブルを使用して ID を生成するときの例外を生成する前に再試行する回数。

既定値

20

有効な値

0 より大きい任意の整数。

Interact | フローチャート

このセクションでは、インタラクティブ・フローチャートの構成設定を定義します。

defaultDateFormat

説明

Interact が日付からストリングへ、およびストリングから日付への変換に使用するデフォルトの日付形式。

既定値

MM/dd/yy

idleFlowchartThreadTimeoutInMinutes

説明

Interact で、インタラクティブ・フローチャートの専用スレッドをアイドル状態にしておける分数。その後、そのスレッドは解放されます。

既定値

5

idleProcessBoxThreadTimeoutInMinutes

説明

Interact で、インタラクティブ・フローチャート・プロセスの専用スレッドをアイドル状態にしておける分数。その後、そのスレッドは解放されます。

既定値

5

maxSizeOfFlowchartEngineInboundQueue

説明

Interact がキューに保持するフローチャート実行要求の最大数。この要求数に到達すると、Interact は要求の受け入れを停止します。

既定値

1000

maxNumberOfFlowchartThreads

説明

インタラクティブ・フローチャート要求の専用スレッドの最大数。

既定値

25

maxNumberOfProcessBoxThreads

説明

インタラクティブ・フローチャート・プロセス専用スレッドの最大数。

既定値

50

maxNumberOfProcessBoxThreadsPerFlowchart

説明

フローチャート・インスタンスごとのインタラクティブ・フローチャート・プロセス専用スレッドの最大数。

既定値

3

minNumberOfFlowchartThreads

説明

インタラクティブ・フローチャート要求の専用スレッドの最小数。

既定値

10

minNumberOfProcessBoxThreads

説明

インタラクティブ・フローチャート・プロセス専用スレッドの最小数。

既定値

20

sessionVarPrefix

説明

セッション変数の接頭部。

既定値

SessionVar

Interact | フローチャート | ExternalCallouts | [ExternalCalloutName]

このセクションは、外部コールアウト API を使用して作成した、カスタムの外部コールアウトのクラス設定を定義します。

クラス

説明

この外部コールアウトによって表される Java クラスの名前。

これは、IBM Unica のマクロ EXTERNALCALLOUT でアクセスできる Java クラスです。

既定値

既定値が定義されていません。

クラスパス

説明

この外部コールアウトによって表される Java クラスのクラスパス。クラスパスは、ランタイム環境サーバー上の JAR ファイルを参照する必要があります。サーバー・グループを使用しており、すべてのランタイム・サーバーが同じ Marketing Platform を使用している場合は、すべてのサーバーの同じロケーションに JAR ファイルのコピーが存在する必要があります。クラスパスは、JAR ファイルの絶対ロケーションで構成されている必要があります。そのロケーションは、そのランタイム環境サーバーのオペレーティング・システムのパス区切り文字 (例えば、Windows ではセミコロン (;)、UNIX システムではコロン (:)) で区切られます。クラス・ファイルが含まれているディレクトリーは承認されません。例えば、Unix システムでは、`/path1/file1.jar:/path2/file2.jar` のようになります。

このクラスパスは、1024 文字未満でなければなりません。.jar ファイル内のマニフェスト・ファイルを使用して、他の .jar ファイルを指定することができます。そのため、クラスパス内に存在する .jar ファイルは 1 つのみにする必要があります。

これは、IBM Unica のマクロ `EXTERNALCALLOUT` でアクセスできる Java クラスです。

既定値

既定値が定義されていません。

Interact | フローチャート | ExternalCallouts | [ExternalCalloutName] | パラメーター・データ (Parameter Data) | [parameterName]

このセクションは、外部コールアウト API を使用して作成したカスタムの外部コールアウトのパラメーター設定を定義します。

値

説明

外部コールアウトのクラスに必要な任意のパラメーターの値。

既定値

既定値が定義されていません。

例

外部コールアウトが外部サーバーのホスト名を必要とする場合は、「ホスト」という名前のパラメーター・カテゴリーを作成し、「値」プロパティーをサーバー名として定義します。

Interact | 監視

この構成プロパティーのセットは、JMX 監視設定を定義できるようにします。これらのプロパティーを構成する必要があるのは、JMX 監視を使用する場合のみです。

Interact 設計環境の構成プロパティーで、コンタクトとレスポンスの履歴モジュール用に定義される JMX 監視プロパティーは、別に存在します。

プロトコル (protocol)

説明

Interact メッセージング・サービス用のプロトコルを定義します。

JMXMP を選択する場合は、以下の JAR ファイルを、以下の順序でクラスパスに組み込む必要があります。

```
Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar
```

既定値

JMXMP

有効な値

JMXMP | RMI

ポート

説明

メッセージング・サービスのポート番号。

既定値

9998

enableSecurity

説明

Interact ランタイム・サーバーの JMXMP メッセージング・サービスのセキュリティーを有効または無効にするブール値。`true` に設定する場合は、Interact のランタイム JMX サービスにアクセスするためのユーザー名とパスワードを提供する必要があります。このランタイム・サーバー用のユーザー資格情報は、Marketing Platform によって認証されます。Jconsole では、空のパスワードでのログインは許可されていません。

プロトコルが RMI の場合、このプロパティーは無効です。Campaign の JMX では、このプロパティーは無効です (Interact の設計時)。

既定値

True

有効な値

True | False

Interact | プロファイル

この構成プロパティーのセットは、オファー非表示およびスコア・オーバーライドを含む、オプションのオファー配信機能の一部を制御します。

enableScoreOverrideLookup

説明

True に設定した場合、Interact はセッションの作成時に、スコア・オーバーライド・データを scoreOverrideTable からロードします。False の場合、Interact はセッションの作成時に、マーケティング・スコア・オーバーライド・データをロードしません。

true の場合は、「Unica」>「Interact」>「プロファイル」>「オーディエンス・レベル」>「(オーディエンス・レベル)」>「scoreOverrideTable」プロパティーも構成する必要があります。定義する必要があるのは、必要なオーディエンス・レベルの scoreOverrideTable プロパティーのみです。オーディエンス・レベルの scoreOverrideTable をブランクにすると、そのオーディエンス・レベルのスコア・オーバーライド・テーブルは使用不可になります。

既定値

False

有効な値

True | False

enableOfferSuppressionLookup

説明

True に設定した場合、Interact はセッションの作成時に、オファー非表示データを offerSuppressionTable からロードします。False の場合、Interact はセッションの作成時に、オファー非表示データをロードしません。

true の場合は、「Unica」>「Interact」>「プロファイル」>「オーディエンス・レベル」>「(オーディエンス・レベル)」>「offerSuppressionTable」プロパティーも構成する必要があります。定義する必要があるのは、必要なオーディエンス・レベルの enableOfferSuppressionLookup プロパティーのみです。

既定値

False

有効な値

True | False

enableProfileLookup

説明

新しくインストールした Interact では、このプロパティーは推奨されていません。アップグレードした Interact のインストール済み環境では、このプロパティーは最初の配置まで有効です。

テーブルのロードの動作は、インタラクティブ・フローチャートで使用されますが、インタラクティブ・チャネルではマップされません。True に設定した場合、Interact はセッションの作成時に、プロファイル・データを profileTable からロードします。

true の場合は、「Unica」>「Interact」>「プロファイル」>「オーディエンス・レベル」>「(オーディエンス・レベル)」>「profileTable」プロパティーも構成する必要があります。

インタラクティブ・チャネル・テーブル・マッピング・ウィザードの「**訪問セッションの開始時にこのデータをメモリーに読み込む**」 設定は、この構成プロパティーをオーバーライドします。

既定値

False

有効な値

True | False

defaultOfferUpdatePollPeriod

説明

システムが、キャッシュに入っているデフォルト・オファー・テーブルのデフォルト・オファーを更新する前に待機する秒数。-1 に設定すると、システムは、ランタイム・サーバーの始動時に初期リストがキャッシュにロードされた後、キャッシュ内のデフォルト・オファーを更新しません。

既定値

-1

Interact | プロファイル | オーディエンス・レベル | [AudienceLevelName]

この構成プロパティーのセットは、Interact の追加機能に必要なテーブル名を定義できるようにします。テーブル名の定義が必要なのは、関連機能を使用している場合のみです。

scoreOverrideTable

説明

オーディエンス・レベルのスコア・オーバーライド情報が含まれているテーブルの名前。このプロパティーは、enableScoreOverrideLookup を true に設定した場合に適用されます。このプロパティーは、スコア・オーバーライド・テーブルを使用可能にするオーディエンス・レベルに対して定義する必要があります。そのオーディエンス・レベルにスコア・オーバーライド・テーブルがない場合は、enableScoreOverrideLookup が true に設定されている場合でも、このプロパティーは未定義のままにすることができます。

Interact は、prodUserDataSource プロパティーによって定義されている Interact ランタイム・サーバーがアクセスする顧客テーブルから、このテーブルを探します。

このデータ・ソースの「スキーマ」プロパティーを定義した場合、Interact はそのスキーマをこのテーブル名の前に付加します。例えば、schema.UACI_ScoreOverride のようになります。例えば mySchema.UACI_ScoreOverride のような完全修飾名を入力した場合、Interact はスキーマ名を前に付加しません。

既定値

UACI_ScoreOverride

offerSuppressionTable

説明

オーディエンス・レベルのオファー非表示情報が含まれているテーブルの名前。このプロパティーは、オファー非表示テーブルを使用可能にするオーディエンス・レベルに対して定義する必要があります。そのオーディエンス・レベルにオファー非表示テーブルがない場合は、enableOfferSuppressionLookup が true に設定されている場合でも、このプロパティーは未定義のままにすることができます。

Interact は、prodUserDataSource プロパティーによって定義されているランタイム・サーバーがアクセスする顧客テーブルから、このテーブルを探します。

既定値

UACI_BlackList

profileTable

説明

新しくインストールした Interact では、このプロパティーは推奨されていません。アップグレードした Interact のインストール済み環境では、このプロパティーは最初の配置まで有効です。

オーディエンス・レベルのプロファイル・データが含まれているテーブルの名前。

Interact は、prodUserDataSource プロパティーによって定義されているランタイム・サーバーがアクセスする顧客テーブルから、このテーブルを探します。

このデータ・ソースの「スキーマ」プロパティーを定義した場合、Interact はそのスキーマをこのテーブル名の前に付加します。例えば、schema.UACI_usrProd のようになります。例えば mySchema.UACI_usrProd のような完全修飾名を入力した場合、Interact はスキーマ名を前に付加しません。

既定値

既定値が定義されていません。

contactHistoryTable

説明

このオーディエンス・レベルのコンタクト履歴データのステージング・テーブル名。

このテーブルは、ランタイム環境テーブル (systemTablesDataSource) に格納されます。

このデータ・ソースの「スキーマ」プロパティーを定義した場合、Interact はそのスキーマをこのテーブル名の前に付加します。例えば、schema.UACI_CHStaging のようになります。例えば mySchema.UACI_CHStaging のような完全修飾名を入力した場合、Interact はスキーマ名を前に付加しません。

既定値

UACI_CHStaging

chOfferAttribTable

説明

このオーディエンス・レベルのコンタクト履歴オファー属性テーブルの名前。

このテーブルは、ランタイム環境テーブル (systemTablesDataSource) に格納されます。

このデータ・ソースの「スキーマ」プロパティーを定義した場合、Interact はそのスキーマをこのテーブル名の前に付加します。例えば、schema.UACI_CHOfferAttrib のようになります。例えば mySchema.UACI_CHOfferAttrib のような完全修飾名を入力した場合、Interact はスキーマ名を前に付加しません。

既定値

UACI_CHOfferAttrib

responseHistoryTable

説明

このオーディエンス・レベルのレスポンス履歴ステージング・テーブルの名前。

このテーブルは、ランタイム環境テーブル (systemTablesDataSource) に格納されます。

このデータ・ソースの「スキーマ」プロパティーを定義した場合、Interact はそのスキーマをこのテーブル名の前に付加します。例えば、schema.UACI_RHStaging のようになります。例えば mySchema.UACI_RHStaging のような完全修飾名を入力した場合、Interact はスキーマ名を前に付加しません。

既定値

UACI_RHStaging

crossSessionResponseTable

説明

レスポンス・トラッキング機能からアクセス可能なコンタクトとレスポンスの履歴テーブルでのクロス・セッション・レスポンス・トラッキングに必要な、このオーディエンス・レベルのテーブルの名前。

このデータ・ソースの「スキーマ」プロパティーを定義した場合、Interact はそのスキーマをこのテーブル名の前に付加します。例えば、schema.UACI_XSessResponse のようになります。例えば mySchema.UACI_XSessResponse のような完全修飾名を入力した場合、Interact はスキーマ名を前に付加しません。

既定値

UACI_XSessResponse

Interact | プロファイル | オーディエンス・レベル | [AudienceLevelName] | 未加工 SQL によるオファー (Offers by Raw SQL)

この構成プロパティーのセットは、Interact の追加機能に必要なテーブル名を定義できるようにします。テーブル名の定義が必要なのは、関連機能を使用している場合のみです。

enableOffersByRawSQL

説明

True に設定すると、Interact はこのオーディエンス・レベルの offersBySQL 機能を使用可能にします。これにより、ランタイムに SQL コードを実行して、必要なオファー候補のセットを作成するように構成することができます。False の場合、Interact は offersBySQL 機能を使用しません。

このプロパティーを true に設定する場合は、「Unica」|「Interact」|「プロファイル」|「オーディエンス・レベル」|「(オーディエンス・レベル)」|「未加工 SQL によるオファー (Offers by Raw SQL)」|「SQL テンプレート (SQL Template)」プロパティーを構成して、1 つ以上の SQL テンプレートを定義することも可能です。

既定値

False

有効な値

True | False

cacheSize

説明

OfferBySQL クエリーの結果の保管に使用されるキャッシュのサイズ。クエリーの結果がほとんどのセッションに対して一意の場合、キャッシュを使用すると悪い影響が出る可能性がありますので、注意してください。

既定値

-1 (オフ)

有効な値

-1 | 値

cacheLifeInMinutes

説明

キャッシュが有効な場合、これは、キャッシュの内容が古くなるのを避けるために、システムがキャッシュを消去するまでの分数を示します。

既定値

-1 (オフ)

有効な値

-1 | 値

defaultSQLTemplate

説明

使用する SQL テンプレートの名前 (API 呼び出しで指定されていない場合)。

既定値

なし

有効な値

SQL テンプレート名

Interact | プロファイル | オーディエンス・レベル | [AudienceLevelName] | SQL テンプレート (SQL Template)

これらの構成プロパティーを使用して、Interact の offersBySQL 機能で使用する 1 つ以上の SQL クエリー・テンプレートを定義することができます。

名前

説明

この SQL クエリー・テンプレートに割り当てる名前。API 呼び出しでこの SQL テンプレートを使用する際に意味のある記述名を入力してください。offerBySQL 処理の「インタラクト・リスト」プロセス・ボックスで定義されている名前と*同一の* 名前をここで使用した場合、ここに入力した SQL ではなく、そのプロセス・ボックス内の SQL が使用されます。

既定値

なし

SQL

説明

このテンプレートによって呼び出される SQL クエリーが入ります。SQL クエリーには、訪問者のセッション・データ (プロファイル) の一部になっている変数名への参照が含まれている場合があります。例えば、`select * from MyOffers where category = ${preferredCategory}` は、`preferredCategory` という名前の変数が含まれているセッションに依存します。

この機能で使用するために設計時に作成した特定のオファー・テーブルを照会するように、SQL を構成する必要があります。ここではストアード・プロシージャーはサポートされていませんので、注意してください。

既定値

なし

Interact | プロファイル | オーディエンス・レベル | [AudienceLevelName] | プロファイル・データ・サービス | [DataSource]

この構成プロパティーのセットは、Interact の追加機能に必要なテーブル名を定義できるようにします。テーブル名の定義が必要なのは、関連機能を使用している場合のみです。プロファイル・データ・サービスのカテゴリーによって、すべてのオーディエンス・レベルに対して作成される組み込みデータ・ソース (データベースと呼ばれる) に関する情報が提供されます。これは、事前構成で優先度 100 に設定されます。ただし、変更したり無効にしたりすることもできます。このカテゴリーには、追加の外部データ・ソース用のテンプレートも含まれています。「**外部データ・サービス (External Data Services)**」というテンプレートをクリックすると、ここに記載する構成設定を入力できます。

新規カテゴリー名

説明

(デフォルトのデータベース項目には使用できません。) 定義しているデータ・ソースの名前。ここで入力する名前は、同一オーディエンス・レベルのデータ・ソース間で固有でなければなりません。

既定値

なし

有効な値

任意のテキスト・ストリングを使用できます。

enabled

説明

`True` に設定されると、このデータ・ソースは割り当てられたオーディエンス・レベルで有効になります。`False` の場合、Interact はこのオーディエンス・レベルでこのデータ・ソースを使用しません。

既定値

`True`

有効な値

True | False

className

説明

(デフォルトのデータベース項目には使用できません。)
`IInteractProfileDataService` を実装するデータ・ソース・クラスの完全修飾名。

既定値

なし。

有効な値

完全修飾クラス名を指定するストリング。

classPath

説明

(デフォルトのデータベース項目には使用できません。) オプションの構成設定で、このデータ・ソース実装クラスをロードするためのパスを指定します。省略すると、デフォルトで、収容アプリケーション・サーバーのクラスパスが使用されます。

既定値

表示されません。ただし、ここで値を指定しない場合はデフォルトで、収容アプリケーション・サーバーのクラスパスが使用されます。

有効な値

クラスパスを指定するストリング。

priority

説明

このオーディエンス・レベル内でのこのデータ・ソースの優先度。各オーディエンス・レベルにおいて、すべてのデータ・ソース間で固有な値でなければなりません。(つまり、あるデータ・ソースで優先度を 100 に設定した場合、そのオーディエンス・レベルでは、他のどのデータ・ソースも優先度 100 にすることはできません。)

既定値

デフォルト・データベースでは 100。ユーザー定義データ・ソースでは 200

有効な値

任意の負でない整数を使用できます。

Interact | offerserving

これらの構成プロパティーは、一般的な学習構成プロパティーを定義します。

組み込み学習を使用する場合、学習実装環境をチューニングするには、設計環境の構成プロパティーを使用します。

optimizationType

説明

optimizationType プロパティーは、オファーの割り当てを支援するために、Interact で学習エンジンを使用するかどうかを定義します。NoLearning に設定すると、Interact は学習を使用しません。BuiltInLearning に設定すると、Interact は Interact を使用して作成された baysean 学習エンジンを使用します。ExternalLearning に設定すると、Interact は指定された学習エンジンを使用します。ExternalLearning を選択する場合は、externalLearningClass プロパティーおよび externalLearningClassPath プロパティーを定義する必要があります。

既定値

NoLearning

有効な値

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

説明

ランタイム・サーバーが、オファーを取得する前に、インタラクティブ・フローチャートの完了を待つ最大ミリ秒数。

既定値

5000

treatmentCodePrefix

説明

コードを処理するため、前に付加される接頭部。

既定値

既定値が定義されていません。

Interact | offerserving | 組み込み学習の構成 (Built-in Learning Config)

これらの構成プロパティーは、組み込み学習のデータベースへの書き込み設定を定義します。

学習実装環境をチューニングするには、設計環境の構成プロパティーを使用します。

insertRawStatsIntervalInMinutes

説明

Interact 学習モジュールが、学習ステージング・テーブルにさらに行を挿入する前に待機する分数。この時間は、ご使用の環境で学習モジュールが処理するデータ量に基づいて、変更が必要になる場合があります。

既定値

5

aggregateStatsIntervalInMinutes

説明

Interact 学習モジュールが待機する、学習ステージング・テーブル内のデータの集約が行われる間隔 (分数)。この時間は、ご使用の環境で学習モジュールが処理するデータ量に基づいて、変更が必要になる場合があります。

既定値

15

有効な値

ゼロより大きい整数。

Interact | offerserving | 外部学習構成 (External Learning Config)

これらの構成プロパティーは、学習 API を使用して作成する外部学習モジュールのクラス設定を定義します。

クラス

説明

optimizationType を ExternalLearning に設定している場合は、externalLearningClass を外部学習エンジンのクラス名に設定します。

既定値

既定値が定義されていません。

使用可能性

このプロパティーは、optimizationType が ExternalLearning に設定されている場合にのみ適用されます。

classPath

説明

optimizationType を ExternalLearning に設定している場合は、externalLearningClass を外部学習エンジンのクラスパスに設定します。

クラスパスは、ランタイム環境サーバー上の JAR ファイルを参照する必要があります。サーバー・グループを使用しており、すべてのランタイム・サーバーが同じ Marketing Platform を使用している場合は、すべてのサーバーの同じロケーションに JAR ファイルのコピーが存在する必要があります。クラスパスは、JAR ファイルの絶対ロケーションで構成されている必要があります。そのロケーションは、そのランタイム環境サーバーのオペレーティング・システムのパス区切り文字 (例えば、Windows ではセミコロン (;)、UNIX システムではコロン (:)) で区切られます。クラス・ファイルが含まれているディレクトリーは承認されません。例えば、Unix システムでは、/path1/file1.jar:/path2/file2.jar のようになります。

このクラスパスは、1024 文字未満でなければなりません。.jar ファイル内のマニフェスト・ファイルを使用して、他の .jar ファイルを指定することができます。そのため、クラスパス内に存在する .jar ファイルは 1 つのみにする必要があります。

既定値

既定値が定義されていません。

使用可能性

このプロパティーは、optimizationType が ExternalLearning に設定されている場合にのみ適用されます。

Interact | offerserving | 外部学習構成 (External Learning Config) | パラメーター・データ (Parameter Data) | [parameterName]

これらの構成プロパティーは、外部学習モジュールの任意のパラメーターを定義します。

値

説明

外部学習モジュールのクラスに必要な任意のパラメーターの値。

既定値

既定値が定義されていません。

例

外部学習モジュールにアルゴリズム・ソルバー・アプリケーションのパスが必要な場合は、solverPath というパラメーター・カテゴリーを作成し、「値」プロパティーをそのアプリケーションのパスとして定義します。

Interact | サービス (services)

このカテゴリーの構成プロパティーは、コンタクトとレスポンスの履歴データの収集や、レポートを作成し、ランタイム環境のシステム・テーブルに書き込むための統計の収集を管理する、すべてのサービスの設定を定義します。

externalLoaderStagingDirectory

説明

このプロパティーは、データベース・ロード・ユーティリティーのステージング・ディレクトリーのロケーションを定義します。

既定値

既定値が定義されていません。

有効な値

ステージング・ディレクトリーの絶対パス、または Interact のインストール・ディレクトリーからの相対パス。

データベース・ロード・ユーティリティーを使用可能にする場合は、contactHist カテゴリーおよび responstHist カテゴリーの cacheType プロパティーを、「外部ローダー・ファイル (External Loader File)」に設定する必要があります。

Interact | サービス (services) | contactHist

このカテゴリーの構成プロパティーは、コンタクト履歴ステージング・テーブルのデータを収集するサービスの設定を定義します。

enableLog

説明

true の場合、コンタクト履歴データを記録するためにデータを収集するサービスが有効になります。false の場合、データは収集されません。

既定値

True

有効な値

True | False

cacheType

説明

コンタクト履歴用に収集されたデータを、メモリー (メモリー・キャッシュ) またはファイル (外部ローダー・ファイル) に保持するかどうかを定義します。外部ローダー・ファイルは、データベース・ローダー・ユーティリティーを使用するように Interact を構成した場合にのみ使用できます。

メモリー・キャッシュを選択する場合には、「キャッシュ」カテゴリーの設定を使用します。外部ローダー・ファイルを選択する場合には、fileCache カテゴリーの設定を使用します。

既定値

メモリー・キャッシュ

有効な値

メモリー・キャッシュ | 外部ローダー・ファイル (External Loader File)

Interact | サービス (services) | contactHist | キャッシュ

このカテゴリーの構成プロパティーは、コンタクト履歴ステージング・テーブルのデータを収集するサービスのキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集したコンタクト履歴データをデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | contactHist | fileCache

このカテゴリーの構成プロパティーは、データベース・ローダー・ユーティリティーを使用している場合に、コンタクト履歴データを収集するサービスのキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集したコンタクト履歴データをデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | defaultedStats

このカテゴリーの構成プロパティーは、対話点のデフォルト・ストリングの使用回数に関する統計を収集するサービスの設定を定義します。

enableLog

説明

`true` の場合、`UACI_DefaultedStat` テーブルに対して対話点のデフォルト・ストリングが使用された回数に関する統計を収集するサービスが有効になります。`false` の場合、デフォルト・ストリングの統計は収集されません。

IBM レポートを使用しない場合は、データ収集は必要ないため、このプロパティーは `false` に設定できます。

既定値

True

有効な値

True | False

Interact | サービス (services) | defaultedStats | キャッシュ

このカテゴリーの構成プロパティーは、対話点のデフォルト・ストリングの使用回数に関する統計を収集するサービスのキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集したデフォルト・ストリングの統計をデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | eligOpsStats

このカテゴリーの構成プロパティーは、対象となるオファーの統計を書き込むサービスの設定を定義します。

enableLog

説明

`true` の場合、対象となるオファーの統計を収集するサービスが有効になります。`false` の場合、対象となるオファーの統計は収集されません。

IBM レポートを使用しない場合は、データ収集は必要ないため、このプロパティーは `false` に設定できます。

既定値

True

有効な値

True | False

Interact | サービス (services) | eligOpsStats | キャッシュ

このカテゴリーの構成プロパティーは、対象となるオファーの統計を収集するサービスのキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集した対象となるオファーの統計をデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | eventActivity

このカテゴリーの構成プロパティーは、イベント・アクティビティーの統計を収集するサービスの設定を定義します。

enableLog

説明

true の場合、イベント・アクティビティーの統計を収集するサービスが有効になります。false の場合、イベントの統計は収集されません。

IBM レポートを使用しない場合は、データ収集は必要ないため、このプロパティーは false に設定できます。

既定値

True

有効な値

True | False

Interact | サービス (services) | eventActivity | キャッシュ

このカテゴリーの構成プロパティーは、イベント・アクティビティーの統計を収集するサービスのキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集したイベント・アクティビティーの統計をデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | customLogger

このカテゴリーの構成プロパティーは、テーブルに書き込むカスタム・データを収集するサービス (UACICustomLoggerTableName イベント・パラメーターを使用するイベント) の設定を定義します。

enableLog

説明

true の場合、テーブルへのカスタム・ログ機能が有効になります。false の場合、UACICustomLoggerTableName イベント・パラメーターは無効です。

既定値

True

有効な値

True | False

Interact | サービス (services) | customLogger | キャッシュ

このカテゴリーの構成プロパティーは、テーブルに入れるカスタム・データを収集するサービス (UACICustomLoggerTableName イベント・パラメーターを使用するイベント) のキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集したカスタム・データをデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | responseHist

このカテゴリーの構成プロパティーは、レスポンス履歴ステージング・テーブルに書き込むサービスの設定を定義します。

enableLog

説明

true の場合、レスポンス履歴ステージング・テーブルに書き込むサービスが有効になります。false の場合、レスポンス履歴ステージング・テーブルへのデータの書き込みは行われません。

レスポンス履歴ステージング・テーブルは、オーディエンス・レベルの responseHistoryTable プロパティーで定義されます。デフォルトは UACI_RHStaging です。

既定値

True

有効な値

True | False

cacheType

説明

キャッシュをメモリーに保持するか、ファイルに保持するかを定義します。外部ローダー・ファイルは、データベース・ローダー・ユーティリティーを使用するように Interact を構成した場合にのみ使用できます。

メモリー・キャッシュを選択する場合には、「キャッシュ」カテゴリーの設定を使用します。外部ローダー・ファイルを選択する場合には、fileCache カテゴリーの設定を使用します。

既定値

メモリー・キャッシュ

有効な値

メモリー・キャッシュ | 外部ローダー・ファイル (External Loader File)

Interact | サービス (services) | responseHist | キャッシュ

このカテゴリーの構成プロパティーは、レスポンス履歴データを収集するサービスのキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集したレスポンス履歴データをデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | responseHist | fileCache

このカテゴリーの構成プロパティーは、データベース・ローダー・ユーティリティーを使用している場合に、レスポンス履歴データを収集するサービスのキャッシュ設定を定義します。

しきい値

説明

Interact がレコードをデータベースに書き込む前に、累積されるレコードの数。

responseHist - オーディエンス・レベルの responseHistoryTable プロパティーで定義されるテーブル。デフォルトは UACI_RHStaging です。

既定値

100

insertPeriodInSecs

説明

データベースへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | crossSessionResponse

このカテゴリーの構成プロパティーは、crossSessionResponse サービスと xsession プロセスの一般的な設定を定義します。これらの設定は、Interact のクロス・セッション・レスポンス・トラッキングを使用している場合にのみ、構成する必要があります。

enableLog

説明

true の場合は、crossSessionResponse サービスが有効になり、Interact はクロス・セッション・レスポンス・トラッキングのステージング・テーブルにデータを書き込みます。false の場合、crossSessionResponse サービスは無効になります。

既定値

False

xsessionProcessIntervalInSecs

説明

xsession プロセスの実行間隔 (秒数)。このプロセスは、クロス・セッション・レスポンス・トラッキングのステージング・テーブルから、レスポンス履歴のステージング・テーブルと組み込み学習モジュールにデータを移動します。

既定値

180

有効な値

ゼロより大きい整数。

purgeOrphanResponseThresholdInMinutes

説明

crossSessionResponse サービスが、コンタクトとレスポンスの履歴テーブル内のコンタクトと一致しないすべてのレスポンスにマーク付けする前に待機する分数。

レスポンスと一致するものがコンタクトとレスポンスの履歴テーブル内に存在しない場合、purgeOrphanResponseThresholdInMinutes の分数が経過すると、Interact は xSessResponse ステージング・テーブルの Mark 列に -1 の値を書き込んで、そのレスポンスにマーク付けします。その後、これらのレスポンスを手動でマッチングするか、削除することができます。

既定値

180

Interact | サービス (services) | crossSessionResponse | キャッシュ

このカテゴリーの構成プロパティーは、クロス・セッション・レスポンス・データを収集するサービスのキャッシュ設定を定義します。

しきい値

説明

flushCacheToDB サービスが収集したクロス・セッション・レスポンス・データをデータベースに書き込む前に、累積されるレコードの数。

既定値

100

insertPeriodInSecs

説明

XSessResponse テーブルへの書き込みを強制する間隔 (秒数)。

既定値

3600

Interact | サービス (services) | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

このセクションのプロパティーは、クロス・セッション・レスポンス・トラッキングで、処理コードをコンタクトとレスポンスの履歴とマッチングする方法を定義します。

SQL

説明

このプロパティーは、Interact がシステムによって生成された SQL を使用するか、OverrideSQL プロパティーで定義されているカスタムの SQL を使用するかを定義します。

既定値

システムによって生成された SQL を使用します。

有効な値

システムによって生成された SQL を使用する (Use System Generated SQL) | SQL をオーバーライドする (Override SQL)

OverrideSQL

説明

処理コードとコンタクトとレスポンスの履歴のマッチングにデフォルトの SQL コマンドを使用しない場合は、SQL またはストアード・プロシージャーをここに入力します。

SQL が「システムによって生成された SQL を使用する (Use System Generated SQL)」に設定されている場合、この値は無視されます。

既定値

useStoredProcedure

説明

true に設定する場合、処理コードをコンタクトとレスポンスの履歴とマッチングするストアード・プロシージャーへの参照が、OverrideSQL に含まれている必要があります。

false に設定する場合、OverrideSQL は (使用するのであれば) SQL クエリーになっている必要があります。

既定値

false

有効な値

true | false

タイプ

説明

ランタイム環境テーブルの UACI_TrackingType テーブルで定義されている、関連する TrackingCodeType。UACI_TrackingType テーブルを変更する場合を除き、Type は 1 にする必要があります。

既定値

1

有効な値

UACI_TrackingType テーブルで定義されている整数。

Interact | サービス (services) | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

このセクションのプロパティーは、クロス・セッション・レスポンス・トラッキングでオファー・コードをコンタクトとレスポンスの履歴とマッチングする方法を定義します。

SQL

説明

このプロパティーは、Interact がシステムによって生成された SQL を使用するか、OverrideSQL プロパティーで定義されているカスタムの SQL を使用するかを定義します。

既定値

システムによって生成された SQL を使用します。

有効な値

システムによって生成された SQL を使用する (Use System Generated SQL) | SQL をオーバーライドする (Override SQL)

OverrideSQL

説明

オファー・コードとコンタクトとレスポンスの履歴のマッチングにデフォルトの SQL コマンドを使用しない場合は、SQL またはストアード・プロシージャーをここに入力します。

SQL が「システムによって生成された SQL を使用する (Use System Generated SQL)」に設定されている場合、この値は無視されます。

既定値

useStoredProcedure

説明

true に設定する場合、オファー・コードをコンタクトとレスポンスの履歴とマッチングするストアード・プロシージャーへの参照が、OverrideSQL に含まれている必要があります。

false に設定する場合、OverrideSQL は (使用するのであれば) SQL クエリーになっている必要があります。

既定値

false

有効な値

true | false

タイプ

説明

ランタイム環境テーブルの UACI_TrackingType テーブルで定義されている、関連する TrackingCodeType。UACI_TrackingType テーブルを変更する場合を除き、Type は 2 にする必要があります。

既定値

2

有効な値

UACI_TrackingType テーブルで定義されている整数。

Interact | サービス (services) | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

このセクションのプロパティーは、クロス・セッション・レスポンス・トラッキングでユーザー定義の代替コードをコンタクトとレスポンスの履歴とマッチングする方法を定義します。

名前

説明

このプロパティーは、代替コードの名前を定義します。これは、ランタイム環境テーブルの UACI_TrackingType テーブル内にある「名前」の値と一致する必要があります。

既定値

OverrideSQL

説明

代替コードをコンタクトとレスポンスの履歴のオファー・コードまたは処理コードとマッチングする、SQL コマンドまたはストアード・プロシージャー。

既定値

useStoredProcedure

説明

true に設定する場合、代替コードをコンタクトとレスポンスの履歴とマッチングするストアード・プロシージャーへの参照が、OverrideSQL に含まれている必要があります。

false に設定する場合、OverrideSQL は (使用するのであれば) SQL クエリーになっている必要があります。

既定値

false

有効な値

true | false

タイプ

説明

ランタイム環境テーブルの UACI_TrackingType テーブルで定義されている、関連する TrackingCodeType。

既定値

3

有効な値

UACI_TrackingType テーブルで定義されている整数。

Interact | サービス (services) | threadManagement | contactAndResponseHist

このカテゴリーの構成プロパティーは、コンタクトとレスポンスの履歴ステージング・テーブルのデータを収集するサービスのスレッド管理設定を定義します。

corePoolSize

説明

コンタクトとレスポンスの履歴データの収集用に、プール内に保持するスレッドの数 (アイドル状態のものも含む)。

既定値

5

maxPoolSize

説明

コンタクトとレスポンスの履歴データの収集用に、プール内に保持するスレッドの最大数。

既定値

5

keepAliveTimeSecs

説明

コンタクトとレスポンスの履歴データを収集するためのスレッドの数がコアよりも多い場合に、超過しているアイドル状態のスレッドが新規タスクを待機する最大時間。この時間が経過すると、それらのスレッドは終了します。

既定値

5

queueCapacity

説明

コンタクトとレスポンスの履歴データを収集するためのスレッド・プールによって使用されるキューのサイズ。

既定値

1000

termWaitSecs

説明

ランタイム・サーバーのシャットダウン時に、コンタクトとレスポンスの履歴データを収集しているサービス・スレッドの完了を待機する秒数。

既定値

5

Interact | サービス (services) | threadManagement | allOtherServices

このカテゴリーの構成プロパティーは、オファーの資格統計、イベント・アクティビティー統計、デフォルト・ストリングの使用統計、およびカスタム・ログを収集してテーブル・データにするサービスのスレッド管理設定を定義します。

corePoolSize

説明

オファーの資格統計、イベント・アクティビティー統計、デフォルト・ストリングの使用統計、およびカスタム・ログを収集してテーブル・データにするサービス用に、プール内に保持するスレッドの数 (アイドル状態のものも含む)。

既定値

5

maxPoolSize

説明

オファーの資格統計、イベント・アクティビティー統計、デフォルト・ストリングの使用統計、およびカスタム・ログを収集してテーブル・データにするサービス用に、プール内に保持するスレッドの最大数。

既定値

5

keepAliveTimeSecs

説明

オファーの資格統計、イベント・アクティビティー統計、デフォルト・ストリングの使用統計、およびカスタム・ログを収集してテーブル・データにするサービス用のスレッドの数がコアよりも多い場合に、超過しているアイドル状態のスレッドが新規タスクを待機する最大時間。この時間が経過すると、それらのスレッドは終了します。

既定値

5

queueCapacity

説明

オファーの資格統計、イベント・アクティビティー統計、デフォルト・ストリングの使用統計、およびカスタム・ログを収集してテーブル・データにするサービスのスレッド・プールによって使用されるキューのサイズ。

既定値

1000

termWaitSecs

説明

ランタイム・サーバーのシャットダウン時に、オファーの資格統計、イベント・アクティビティー統計、デフォルト・ストリングの使用統計、およびカスタム・ログを収集してテーブル・データにするサービスのサービス・スレッドの完了を待機する秒数。

既定値

5

Interact | サービス (services) | threadManagement | flushCacheToDB

このカテゴリーの構成プロパティーは、キャッシュ内にある収集されたデータをランタイム環境のデータベース・テーブルに書き込むスレッドの、スレッド管理設定を定義します。

corePoolSize

説明

キャッシュに入れられたデータをデータ・ストアに書き込むスケジュール済みのスレッド用に、プール内に保持するスレッドの数。

既定値

5

maxPoolSize

説明

キャッシュに入れられたデータをデータ・ストアに書き込むスケジュール済みのスレッド用に、プール内に保持するスレッドの最大数。

既定値

5

keepAliveTimeSecs

説明

キャッシュに入れられたデータをデータ・ストアに書き込むスケジュール済みのスレッドの数がコアよりも多い場合に、超過しているアイドル状態のスレッドが新規タスクを待機する最大時間。この時間が経過すると、それらのスレッドは終了します。

既定値

5

queueCapacity

説明

キャッシュに入れられたデータをデータ・ストアに書き込むスケジュール済みのスレッドのスレッド・プールによって使用されるキューのサイズ。

既定値

1000

termWaitSecs

説明

ランタイム・サーバーのシャットダウン時に、キャッシュに入れられたデータをデータ・ストアに書き込むスケジュール済みのスレッドについて、サービス・スレッドの完了を待機する秒数。

既定値

5

Interact | sessionManagement

この構成プロパティーのセットは、ランタイム・セッションの設定を定義します。

cacheType

説明

ランタイム・サーバーのキャッシュ方法のタイプを定義します。

既定値

ローカル

有効な値

配布済み | ローカル

maxNumberOfSessions

説明

キャッシュに 1 度に保持できるランタイム・セッションの最大数。キャッシュがこの最大数に到達した後で新しいランタイム・セッションの追加要求が発生すると、そのキャッシュの最も古い非アクティブなランタイム・セッションが削除されます。

既定値

999999999

有効な値

0 より大きい整数。

multicastIPAddress

説明

cacheType が「配布済み」の場合は、分散キャッシュによって使用される IP アドレスを入力します。multicastPort も定義する必要があります。

cacheType が「ローカル」の場合は、multicastIPAddress を未定義のままにすることができます。

既定値

230.0.0.1

有効な値

任意の有効な IP アドレス。

multicastPort

説明

cacheType が「配布済み」の場合は、分散キャッシュによって使用されるポート番号を入力します。multicastIPAddress も定義する必要があります。

cacheType が「ローカル」の場合は、multicastPort を未定義のままにすることができます。

既定値

6363

有効な値

1024 – 49151

sessionTimeoutInSecs

説明

セッションを非アクティブのままにしておける時間 (秒単位)。
sessionTimeout の秒数が経過すると、Interact はそのセッションを終了します。

既定値

300

有効な値

ゼロより大きい任意の整数。

付録 C. Interact 設計環境の構成プロパティー

このセクションでは、Interact 設計環境のすべての構成プロパティーについて説明します。

Campaign | パーティション | パーティション[n] | レポート

これらの構成プロパティーは、レポートのフォルダーを定義します。

offerAnalysisTabCachedFolder

説明

offerAnalysisTabCachedFolder プロパティーは、ナビゲーション・ペインの「分析」リンクをクリックして「分析」タブに移動した際に、そのタブ上にリストされる満杯の (拡張された) オファー・レポートの仕様を入れるフォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

説明

segmentAnalysisTabOnDemandFolder プロパティーは、セグメントの「分析」タブにリストされるセグメント・レポートを入れるフォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

説明

offerAnalysisTabOnDemandFolder プロパティーは、オファーの「分析」タブにリストされるオファー・レポートを入れるフォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

segmentAnalysisTabCachedFolder

説明

segmentAnalysisTabCachedFolder プロパティーは、ナビゲーション・ペインの「分析」リンクをクリックして「分析」タブに移動した際に、そのタブ上にリストされる満杯の (拡張された) セグメント・レポートの仕様を入れるフォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='segment']

analysisSectionFolder

説明

analysisSectionFolder プロパティーは、レポート仕様を格納するルート・フォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

/content/folder[@name='Affinium Campaign']

campaignAnalysisTabOnDemandFolder

説明

campaignAnalysisTabOnDemandFolder プロパティーは、キャンペーンの「分析」タブにリストされるキャンペーン・レポートを入れるフォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='campaign']

campaignAnalysisTabCachedFolder

説明

campaignAnalysisTabCachedFolder プロパティーは、ナビゲーション・ペインの「分析」リンクをクリックして「分析」タブに移動した際に、そのタブ上にリストされる満杯の (拡張された) キャンペーン・レポートの仕様を入れるフォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='campaign']/folder[@name='cached']

campaignAnalysisTabEmessageOnDemandFolder

説明

campaignAnalysisTabEmessageOnDemandFolder プロパティーは、キャンペーンの「分析」タブにリストされる eMessage レポートを入れるフォルダーの場所を指定します。パスは、XPath 表記を使用して指定されます。

既定値

/content/folder[@name='Affinium Campaign']/folder[@name='eMessage Reports']

campaignAnalysisTabInteractOnDemandFolder

説明

Interact レポートのレポート・サーバー・フォルダー・ストリングです。

既定値

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

interactiveChannelAnalysisTabOnDemandFolder

説明

「インタラクティブ・チャネル」分析タブ・レポートのレポート・サーバー・フォルダー・ストリングです。

既定値

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking

これらの構成プロパティーは、Interact コンタクトとレスポンスの履歴モジュールの設定を定義します。

isEnabled

説明

「はい」に設定すると、Interact のコンタクトとレスポンスの履歴を Interact ランタイムのステージング・テーブルから Campaign のコンタクトとレスポンスの履歴テーブルにコピーする、Interact のコンタクトとレスポンスの履歴モジュールが有効になります。プロパティー `interactInstalled` も「はい」に設定する必要があります。

既定値

no

有効な値

yes | no

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

runOnceADay

説明

コンタクトとレスポンスの履歴 ETL を 1 日 1 回実行するかどうかを指定します。このプロパティーを「はい」に設定すると、preferredStartTime および preferredEndTime で指定され、スケジュールされた時間間隔内に ETL が実行されます。

ETL の実行時間が 24 時間を超過し、次の日の開始時間にかかる場合は、その日の実行はスキップされ、翌日のスケジュールされている時間に実行されます。例えば、ETL が 午前 1 時から午前 3 時の間に実行されるように構成されている場合に、月曜日の午前 1 時に処理が開始され、火曜日の午前 2 時に完了すると、本来火曜日の午前 1 時にスケジュールされていた次の実行はスキップされ、次回の ETL は水曜日の午前 1 時に開始されます。

ETL スケジューリングは、夏時間調整による変更には対応していません。例えば、午前 1 時から午前 3 時までの間に実行するようにスケジュールされている ETL は、夏時間調整による変更があると、午前 0 時または午前 2 時に実行される可能性があります。

既定値

いいえ

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

processSleepIntervalInMinutes

説明

Interact ランタイムのステージング・テーブルから Campaign のコンタクトとレスポンスの履歴テーブルにデータをコピーする間、Interact のコンタクトとレスポンスの履歴モジュールが待機する分数。

既定値

60

有効な値

ゼロより大きい任意の整数。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

preferredStartTime

説明

毎日の ETL 処理用に設定済みの開始時間。このプロパティーを preferredEndTime プロパティーと組み合わせて使用すると、ETL をその間に実行する時間間隔が設定されます。ETL は、指定された時間間隔の中で開始され、最大で maxJDBCFetchBatchSize を使用して指定された数のレコードを処理します。形式は HH:mm:ss AM または PM で、12 時間クロックを使用します。

既定値

12:00:00 AM

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

preferredEndTime

説明

毎日の ETL 処理用に設定済みの完了時間。このプロパティーを preferredStartTime プロパティーと組み合わせて使用すると、ETL をその間に実行する時間間隔が設定されます。ETL は、指定された時間間隔の中で開始され、最大で maxJDBCFetchBatchSize を使用して指定された数のレコードを処理します。形式は HH:mm:ss AM または PM で、12 時間クロックを使用します。

既定値

2:00:00 AM

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

purgeOrphanResponseThresholdInMinutes

説明

Interact のコンタクトとレスポンスの履歴モジュールが、対応するコンタクトがないレスポンスをパージする前に待機する分数。これにより、コンタクトのログ記録がないレスポンスがログに記録されないようにします。

既定値

180

有効な値

ゼロより大きい任意の整数。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

maxJDBCInsertBatchSize

説明

照会をコミットする前の JDBC バッチ・レコードの最大数。これは、単一の反復の中で Interact のコンタクトとレスポンスの履歴モジュールが処理するレコードの最大数ではありません。それぞれの反復の間は、Interact のコンタクトとレスポンスの履歴モジュールは、ステージング・テーブルから使用可能なすべてのレコードを処理します。ただし、それらのレコードはすべて maxJDBCInsertSize のチャンクに分割されます。

既定値

1000

有効な値

ゼロより大きい任意の整数。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

maxJDBCFetchBatchSize

説明

ステージング・データベースからフェッチする JDBC バッチ・レコードの最大数。コンタクトとレスポンスの履歴モジュールのパフォーマンスを調整するために、この値を大きくする必要がある場合があります。

例えば、1 日に 250 万個のコンタクト履歴レコードを処理するには、maxJDBCFetchBatchSize を 250 万より大きな数に設定して、1 日分のレコードがすべて処理されるようにする必要があります。

その後、maxJDBCFetchChunkSize と maxJDBCInsertBatchSize を、それよりも小さな値 (この例の場合は、それぞれ 50,000 と 10,000 など) に設定します。翌日のレコードの一部も処理される可能性がありますが、その後は翌日まで保持されます。

既定値

1000

有効な値

ゼロより大きい任意の整数

maxJDBCFetchChunkSize

説明

ETL (抽出、変換、ロード) 中に読み取られるデータの JDBC チャンク・サイズの最大数値。チャンク・サイズを挿入サイズより大きくすることで、ETL 処理の速度が向上する場合もあります。

既定値

1000

有効な値

ゼロより大きい任意の整数

deleteProcessedRecords

説明

処理後にコンタクト履歴とレスポンス履歴のレコードを保持するかどうかを指定します。

既定値

はい

completionNotificationScript

説明

ETL の完了時に実行するスクリプトの絶対パスを指定します。スクリプトを指定すると、4 つの引数 (開始時刻、終了時刻、処理された CH レコードの合計数、および処理された RH レコードの合計数) が完了通知スクリプトに渡されます。開始時刻と終了時刻は、1970 年から経過したミリ秒数を表す数値です。

既定値

なし

fetchSize

説明

ステージング・テーブルからレコードを取得する場合に JDBC fetchSize を設定できるようにします。

特に Oracle データベースでは、この設定は、ネットワークの往復ごとに JDBC が取得する必要があるレコード数に合わせて調整してください。100K 以上の大きな規模の場合には、10000 で試行してください。この値は大きくしすぎないように注意してください。使用する値が大きすぎると、メモリーの使用量に影響するのに対し、効果はほとんどありません。

既定値

なし

Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

これらの構成プロパティーは、Interact コンタクトとレスポンスの履歴モジュールのデータ・ソースを定義します。

jndiName

説明

systemTablesDataSource プロパティーを使用して、アプリケーション・サーバー (Websphere または WebLogic) で Interact ランタイム・テーブル用に定義されている Java Naming and Directory Interface (JNDI) データ・ソースを識別します。

Interact ランタイム・データベースは、aci_runtime および aci_populate_runtime の各 dll スクリプトが取り込まれたデータベースで、例えば UACI_CHOfferAttrib や UACI_DefaultedStat などのテーブルが含まれます。

既定値

既定値が定義されていません。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

databaseType

説明

Interact ランタイム・データ・ソースのデータベース・タイプ。

既定値

SQLServer

有効な値

SQLServer | Oracle | DB2

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

schemaName

説明

コンタクトとレスポンスの履歴モジュールのステージング・テーブルが含まれているスキーマの名前。これは、ランタイム環境テーブルと同じにする必要があります。

スキーマを定義する必要はありません。

既定値

既定値が定義されていません。

Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

これらの構成プロパティーは、レポート作成または学習目的で「コンタクト」にマップするキャンペーンからのコンタクト・タイプを定義します。

コンタクト済み

説明

Campaign システム・テーブル内の UA_DtlContactHist テーブルの ContactStatusID 列に割り当てられる、オファー・コンタクト用の値。この

値は、UA_ContactStatus テーブルの有効なエントリーである必要があります。コンタクト・タイプの追加について詳しくは、「*Campaign 管理者ガイド*」を参照してください。

既定値

2

有効な値

ゼロより大きい整数。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

これらの構成プロパティーは、レポート作成または学習を承認するか拒否するかというレスポンスを定義します。

承認

説明

Campaign システム・テーブル内の UA_ResponseHistory テーブルの ResponseTypeID 列に割り当てられる、承認済みオファー用の値。値は、UA_UsrResponseType テーブルの有効なエントリーでなければなりません。CountsAsResponse 列に、レスポンスを意味する値 1 を割り当てる必要があります。

レスポンス・タイプの追加について詳しくは、「*Campaign 管理者ガイド*」を参照してください。

既定値

3

有効な値

ゼロより大きい整数。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

拒否

説明

Campaign システム・テーブル内の UA_ResponseHistory テーブルの ResponseTypeID 列に割り当てられる、拒否済みのオファー用の値。値は、UA_UsrResponseType テーブルの有効なエントリーでなければなりません。CountsAsResponse 列に、拒否を意味する値 2 を割り当てる必要があります。レスポンス・タイプの追加について詳しくは、「*Campaign 管理者ガイド*」を参照してください。

既定値

8

有効な値

ゼロより大きい任意の整数。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | レポート

これらの構成プロパティーは、Cognos と統合した場合のレポート名を定義します。

interactiveCellPerformanceByOfferReportName

説明

オファー別のインタラクティブ・セル実績レポートの名前。この名前は、Cognos サーバー上のこのレポートの名前と一致している必要があります。

既定値

オファー別のインタラクティブ・セル実績

treatmentRuleInventoryReportName

説明

処理ルール・インベントリー・レポートの名前。この名前は、Cognos サーバー上のこのレポートの名前と一致している必要があります。

既定値

チャネル処理ルール・インベントリー

deploymentHistoryReportName

説明

配置履歴レポートの名前。この名前は、Cognos サーバー上のこのレポートの名前と一致している必要があります。

既定値

チャネル配置履歴

Campaign | パーティション | パーティション[n] | Interact | 学習

これらの構成プロパティーによって、組み込み学習モジュールを調整できます。

confidenceLevel

説明

学習ユーティリティーがどの程度確実と判断してから、調査から利用に切り替えるかを、パーセンテージで示します。値 0 は、調査を事実上シャットオフします。

このプロパティーは、Interact ランタイムの「Interact」>「offerserving」>「optimizationType」プロパティーが BuiltInLearning に設定されている場合にのみ適用されます。

既定値

95

有効な値

0 から 95 までの間の 5 で割り切れる整数、または 99。

enableLearning

説明

「はい」に設定すると、Interact 設計時には学習が有効になっていると想定されます。enableLearning を「はい」に設定する場合は、「Interact」>「offerserving」>「optimizationType」を BuiltInLearning または ExternalLearning に構成する必要があります。

「いいえ」に設定すると、Interact 設計時には学習が無効になっていると想定されます。enableLearning を「いいえ」に設定する場合は、「Interact」>「offerserving」>「optimizationType」を NoLearning に構成する必要があります。

既定値

いいえ

maxAttributeNames

説明

Interact 学習ユーティリティーがモニターする学習属性の最大数。

このプロパティーは、Interact ランタイムの「Interact」>「offerserving」>「optimizationType」プロパティーが BuiltInLearning に設定されている場合にのみ適用されます。

既定値

10

有効な値

任意の整数。

maxAttributeValues

説明

各学習属性について、Interact 学習モジュールがトラッキングする値の最大数。

このプロパティーは、Interact ランタイムの「Interact」>「offerserving」>「optimizationType」プロパティーが BuiltInLearning に設定されている場合にのみ適用されます。

既定値

5

otherAttributeValue

説明

maxAttributeValues を超えるすべての属性値を表すために使用される属性値のデフォルトの名前。

このプロパティーは、Interact ランタイムの「Interact」>「offerserving」>「optimizationType」プロパティーが BuiltInLearning に設定されている場合にのみ適用されます。

既定値

その他

有効な値

ストリングまたは数値。

例

maxAttributeValues が 3 に設定されており、かつ otherAttributeValue が「その他」に設定されている場合、学習モジュールは最初の 3 つの値をトラッキングします。その他の値はすべて「その他」カテゴリーに割り当てられます。例えば、訪問者の属性である髪色をトラッキングするとします。最初の 5 人の訪問者の髪色が黒、茶、ブロンド、赤、およびグレイの場合、学習ユーティリティーがトラッキングする髪色は、黒、茶、およびブロンドです。赤およびグレイの各色は、otherAttributeValue の「その他」にグループ化されます。

percentRandomSelection

説明

学習モジュールがランダムにオファーする回数のパーセント。例えば、percentRandomSelection が 5 に設定されている場合、これは、学習モジュールが、全体の回数のうち 5% (100 回推奨されるごとに 5 回) でランダム・オファーを行うことを意味します。

既定値

5

有効な値

0 から 100 までの任意の整数。

recencyWeightingFactor

説明

recencyWeightingPeriod によって定義されているデータのセットのパーセンテージを表す 10 進表記。例えば、デフォルト値の .15 は、recencyWeightingPeriod が示す学習ユーティリティーによって使用されるデータの 15% を意味します。

このプロパティーは、Interact ランタイムの「Interact」>「offerserving」>「optimizationType」プロパティーが BuiltInLearning に設定されている場合にのみ適用されます。

既定値

0.15

有効な値

1 より小さな 10 進数値。

recencyWeightingPeriod

説明

学習モジュールによって recencyWeightingFactor のパーセンテージの重みが付与されたデータの期間 (時間単位)。例えば、デフォルト値の 120 は、学習モジュールによって使用されるデータの recencyWeightingFactor が、過去 120 時間以内のものであることを意味します。

このプロパティーは、optimizationType が builtInLearning に設定されている場合にのみ適用されます。

既定値

120

minPresentCountThreshold

説明

データが計算に使用され、学習モジュールが調査モードに入る前に、オファーされる必要がある最小回数。

既定値

0

有効な値

ゼロ以上の整数。

enablePruning

説明

「はい」に設定する場合、Interact 学習モジュールは、学習属性 (標準または動的) が予測ではないときをアルゴリズムによって判別します。学習属性が予測ではない場合、学習モジュールはオファーの重みを決定するときにその属性について考慮しません。これは、学習モジュールが学習データを集約するまで継続します。

「いいえ」に設定すると、学習モジュールは常にすべての学習属性を使用します。予測ではない属性のプルーニングを行わないと、学習モジュールの正確性は本来よりも低くなります。

既定値

はい

有効な値

はい | いいえ

Campaign | パーティション | パーティション[n] | Interact | 学習 | learningAttributes | [learningAttribute]

これらの構成プロパティーは、学習属性を定義します。

attributeName

説明

各 attributeName は、学習モジュールがモニターする訪問者属性の名前です。これは、セッション・データ内の名前と値のペアの名前と一致している必要があります。

このプロパティーは、Interact ランタイムの「Interact」>「offerserving」>「optimizationType」プロパティーが BuiltInLearning に設定されている場合にのみ適用されます。

既定値

既定値が定義されていません。

Campaign | パーティション | パーティション[n] | Interact | 配置

これらの構成プロパティーは、配置の設定を定義します。

chunkSize

説明

各 Interact 配置パッケージのフラグメントの最大サイズ (KB 単位)。

既定値

500

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | serverGroups | [serverGroup]

これらの構成プロパティーは、サーバー・グループの設定を定義します。

serverGroupName

説明

Interact ランタイム・サーバー・グループの名前。これは、インタラクティブ・チャネルの「サマリー」タブに表示される名前です。

既定値

既定値が定義されていません。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

これらの構成プロパティーは、Interact ランタイム・サーバーを定義します。

instanceURL

説明

Interact ランタイム・サーバーの URL。サーバー・グループにはいくつかの Interact ランタイム・サーバーを含めることができますが、それらのサーバーはそれぞれ新しいカテゴリーの下に作成する必要があります。

既定値

既定値が定義されていません。

例

`http://server:port/interact`

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | フローチャート

これらの構成プロパティーは、インタラクティブ・フローチャートのテスト実行に使用される Interact ランタイム環境を定義します。

serverGroup

説明

Campaign がテスト実行に使用する Interact サーバー・グループの名前。この名前は、`serverGroups` の下に作成するカテゴリー名と一致する必要があります。

既定値

既定値が定義されていません。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

dataSource

説明

dataSource プロパティーを使用して、インタラクティブ・フローチャートのテスト実行時に使用する Campaign の物理データ・ソースを識別します。このプロパティーは、Interact の設計時用に定義されているテスト実行データ・ソースの 「Campaign」>「パーティション」>「パーティション[n]」>「dataSources」プロパティーで定義されるデータ・ソースと一致している必要があります。

既定値

既定値が定義されていません。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

これらの構成プロパティーは、デフォルトのオファー・テーブルのデフォルトのセル・コードを定義します。これらのプロパティーを構成する必要があるのは、グローバルなオファーの割り当てを定義する場合のみです。

DefaultCellCode

説明

デフォルトのオファー・テーブルでセル・コードを定義していない場合に、Interact が使用するデフォルトのセル・コード。

既定値

既定値が定義されていません。

有効な値

Campaign で定義されているセル・コードの形式と一致するストリング。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

これらの構成プロパティーは、offersBySQL テーブルのデフォルトのセル・コードを定義します。これらのプロパティーを構成する必要があるのは、SQL 照会を使用して必要なオファー候補のセットを取得する場合のみです。

DefaultCellCode

説明

OffersBySQL テーブル内のセル・コード列に NULL 値が入っている (または、セル・コード列が完全に存在しない) 任意のオファーに Interact が使用する、デフォルトのセル・コード。この値はセル・コードとして有効な値にする必要があります。

既定値

既定値が定義されていません。

有効な値

Campaign で定義されているセル・コードの形式と一致するストリング。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

これらの構成プロパティーは、スコア・オーバーライド・テーブルのデフォルトのセル・コードを定義します。これらのプロパティーを構成する必要があるのは、個々のオファーの割り当てを定義する場合のみです。

DefaultCellCode

説明

スコア・オーバーライド・テーブルでセル・コードを定義していない場合に、Interact が使用するデフォルトのセル・コード。

既定値

既定値が定義されていません。

有効な値

Campaign で定義されているセル・コードの形式と一致するストリング。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

Campaign | パーティション | パーティション[n] | サーバー (server) | 内部 (internal)

このカテゴリーのプロパティーは、選択された Campaign パーティションの統合設定と internalID の制限を指定します。 Campaign のインストール済み環境に複数のパーティションがある場合は、反映させるパーティションごとにこれらのプロパティーを設定します。

internalIdLowerLimit

説明

internalIdUpperLimit プロパティーと internalIdLowerLimit プロパティーは、Campaign 内部 ID を指定の範囲に制限します。それらのプロパティーでは境界上の値が含まれるので、Campaign は上限と下限のどちらの値も使用できます。

既定値

0 (ゼロ)

internalIdUpperLimit

説明

internalIdUpperLimit プロパティーと internalIdLowerLimit プロパティーは、Campaign 内部 ID を指定の範囲に制限します。それらのプロパティーでは境界上の値が含まれるので、Campaign は上限と下限のどちらの値も使用できます。

既定値

4294967295

eMessageInstalled

説明

eMessage がインストールされていることを示します。yes を選択すると、eMessage 機能が Campaign インターフェースで使用できます。

IBM インストーラーは、eMessage インストールの既定のパーティションに関してこのプロパティーを yes に設定します。eMessage をインストールした追加パーティションについては、このプロパティーを手動で構成する必要があります。

既定値

no

有効な値

yes | no

interactInstalled

説明

Interact 設計環境をインストール後、この構成プロパティーを yes に設定し、Campaign で Interact 設計環境を有効にしてください。

Interact をインストールしていない場合、no に設定してください。このプロパティーを no に設定しても、Interact メニューとオプションがユーザー・インターフェースから削除されることはありません。メニューとオプションを削除するには、configTool ユーティリティーを使用して Interact を手動で登録抹消しなければなりません。

既定値

no

有効な値

yes | no

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

MO_UC_integration

説明

このパーティションに対して、Marketing Operations との統合を有効にします。以下の 3 つのオプションのいずれかを「はい」に設定する予定の場合は、**MO_UC_integration** も「はい」に設定する必要があります。この統合の構成について詳しくは、「*IBM Unica Marketing Operations and Campaign 統合ガイド*」を参照してください。

既定値

no

有効な値

yes | no

MO_UC_BottomUpTargetCells

説明

このパーティションのターゲット・セル・スプレッドシートについて、ボトムアップのセルを許可します。「はい」に設定すると、トップダウンとボトムアップの両方のターゲット・セルが表示されますが、ボトムアップのターゲット・セルは読み取り専用です。**MO_UC_integration** が有効になっている必要があります。この統合の構成について詳しくは、「*IBM Unica Marketing Operations and Campaign 統合ガイド*」を参照してください。

既定値

no

有効な値

yes | no

Legacy_campaigns

説明

MO_UC_integration プロパティーが **Yes** に設定されていると、**Legacy_campaigns** プロパティーによって、統合が有効になる前に作成されたキャンペーン (Campaign 7.x で作成されたキャンペーンおよび Plan 7.x プロジェクトにリンクされたキャンペーンを含む) にアクセスできるようになります。この統合の構成について詳しくは、「*IBM Unica Marketing Operations and Campaign 統合ガイド*」を参照してください。

既定値

no

有効な値

yes | no

IBM Unica Marketing Operations - Offer の統合

説明

Marketing Operations を使用してこのパーティションに対してオファーのライフサイクル管理タスクを実行する機能を有効にします。(**MO_UC_integration** が有効になっている必要があります。また、「**設定**」>「**構成**」>「**Unica**」>「**プラットフォーム**」で、「**キャンペーンの統合**」が有効になっている必要があります。) この統合の構成について詳しくは、「*IBM Unica Marketing Operations and Campaign 統合ガイド*」を参照してください。

既定値

no

有効な値

yes | no

UC_CM_integration

説明

キャンペーン・パーティションの IBM Coremetrics® オンライン・セグメント統合を有効にします。このオプションを「はい」に設定すると、フローチャート内の「選択」プロセス・ボックスに、入力として「**IBM Coremetrics セグメント**」を選択するオプションが表示されます。パーティションごとに統合を構成するには、「**設定**」>「**構成**」>「**Campaign**」|「**パーティション**」|「**パーティション[n]**」|「**Coremetrics**」 を選択します。

既定値

no

有効な値

yes | no

Campaign | 監視

このカテゴリーのプロパティーは、操作監視機能を有効にするかどうか、操作監視サーバーの URL、およびキャッシング動作を指定します。操作監視機能ではアクティブなフローチャートが表示されて、それらを制御できます。

cacheCleanupInterval

説明

cacheCleanupInterval プロパティーは、フローチャート・ステータス・キャッシュの自動クリーンアップ間隔を秒単位で指定します。

Campaign バージョン 7.0 より前のバージョンでは、このプロパティーは使用できません。

既定値

600 (10 分)

cacheRunCompleteTime

説明

cacheRunCompleteTime プロパティーは、完了済み実行タスクがキャッシュに入れられて、「監視」ページに表示される期間を分単位で指定します。

Campaign バージョン 7.0 より前のバージョンでは、このプロパティーは使用できません。

既定値

4320

monitorEnabled

説明

monitorEnabled プロパティーは、監視機能を有効にするかどうかを指定します。

Campaign バージョン 7.0 より前のバージョンでは、このプロパティーは使用できません。

既定値

はい

serverURL

説明

「キャンペーン」>「監視」>「serverURL」プロパティーは、操作監視サーバーの URL を指定します。これは必須設定で、操作監視サーバー URL が既定以外の場合には、値を変更してください。

Campaign が Secure Sockets Layer (SSL) 通信を使用するように構成されている場合には、HTTPS を使用するようにこのプロパティーの値を設定します。例えば、次のようにします。 serverURL=https://*host:SSL_port*/Campaign/OperationMonitor ここで、それぞれの意味は次のとおりです。

- *host* は、Web アプリケーションがインストールされているマシンの名前または IP アドレスです。
- *SSL_Port* は Web アプリケーションの SSL ポートです。

URL の https に注意してください。

既定値

http://localhost:7001/Campaign/OperationMonitor

monitorEnabledForInteract

説明

「はい」に設定すると、Campaign JMX コネクター・サーバーが Interact で使用可能になります。Campaign には JMX セキュリティーはありません。

「いいえ」に設定すると、Campaign JMX コネクター・サーバーに接続できません。

この JMX 監視は、Interact コンタクトとレスポンスの履歴モジュール専用です。

既定値

False

有効な値

True | False

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

プロトコル (protocol)

説明

monitorEnabledForInteract が「はい」に設定されている場合、Campaign JMX コネクター・サーバーのリスニング・プロトコルです。

この JMX 監視は、Interact コンタクトとレスポンスの履歴モジュール専用です。

既定値

JMXMP

有効な値

JMXMP | RMI

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

ポート

説明

monitorEnabledForInteract が「はい」に設定されている場合、Campaign JMX コネクター・サーバーのリスニング・ポートです。

この JMX 監視は、Interact コンタクトとレスポンスの履歴モジュール専用です。

既定値

2004

有効な値

1025 から 65535 までの整数。

使用可能性

このプロパティーは、Interact をインストールしてある場合のみ適用可能です。

付録 D. クライアント・サイドでのリアルタイム・オファーのカスタマイズ

低レベルの Java コードまたは SOAP 呼び出しを Interact サーバーに実装せずに、リアルタイム・オファーのカスタマイズを提供する場合があります。例えば、Javascript コンテンツだけが有効な拡張プログラミングである Web ページを最初に訪問者が読み込む場合や、HTML コンテンツだけが有効な E メール・メッセージを訪問者が開く場合があります。 IBM Unica Interact にはいくつかのコネクターが用意されており、それらのコネクターを使用することで、クライアント・サイドで読み込まれる Web コンテンツのみを制御する場合や、Interact へのインターフェースを単純化する場合の、リアルタイム・オファーを管理できます。

Interact インストール済み環境には、クライアント・サイドで開始されるオファーのカスタマイズ用の以下の 2 つのコネクターが含まれています。

- 『Interact Message Connector について』. Message Connector を使用することで、例えば、E メール・メッセージや他の電子メディアの Web コンテンツにイメージ・タグとリンク・タグを組み込み、Interact サーバーを呼び出して、ページ読み込み時に提示されるオファーおよびクリックスルー・ランディング・ページを示すことができます。
- 244 ページの『Interact Web Connector について』. Web Connector (JS Connector ともいう) を使用することで、Web ページでクライアント・サイドの JavaScript を使用し、ページ読み込みオファー提示とクリックスルー・ランディング・ページを通じて、オファーのアービトレーション、提示、およびコンタクト/レスポンス履歴を管理できます。

Interact Message Connector について

Interact Message Connector を使用することで、E メール・メッセージや他の電子メディアで IBM Unica Interact を呼び出し、オープン時、および顧客が指定サイトにリンクされているメッセージをクリックスルーしたときに、カスタマイズされたオファーを提示することができます。これは 2 つのキー・タグを使用して行われます。1 つは、オープン時にカスタマイズされたオファーを読み込むイメージ・タグ (IMG) で、もう 1 つはクリックスルーに関する情報を取り込み、特定のランディング・ページに顧客をリダイレクトするリンク・タグ (A) です。

例

以下の例には、マーケティング・スポット (E メール・メッセージなど) に含まれる可能性がある HTML コードがいくつか示されています。このマーケティング・スポットには、IMG タグ URL (文書が開いたときに情報を Interact サーバーに渡し、返された適切なオファー・イメージを取得する) と、A タグ URL (クリックスルー時に Interact サーバーに渡す情報を決定する) の両方が含まれます。

```
<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>
```

次の例では、IMG タグが A タグで囲まれているため、以下のような動作になります。

1. E メール・メッセージを開くと、Message Connector は IMG タグ内のエンコードされた情報 (このメッセージの msgID と linkID、およびユーザー ID、所得水準、所得タイプを含む顧客パラメーター) を含む要求を受け取ります。
2. この情報は、API 呼び出しを通じて、Interact ランタイム・サーバーに渡されます。
3. ランタイム・サーバーはオファーを Message Connector に戻します。Message Connector は、オファー・イメージの URL を取得し、その URL (追加パラメーターを含む) を指定して、そのオファー URL にイメージ要求をリダイレクトします。
4. オファーはイメージとして顧客に示されます。

その時点で、顧客はそのイメージをクリックして、なんらかの方法でオファーに応える可能性があります。A タグとその指定された HREF 属性 (宛先 URL を指定する) を使用してクリックスルーした場合、そのオファーの URL にリンクされたランディング・ページの別の要求が Message Connector に送られます。その後、顧客のブラウザーはオファーに構成されているランディング・ページにリダイレクトされます。

クリックスルー A タグは厳密には必要ではないことに注意してください。オファーは、顧客が印刷するクーポンなどのイメージのみで構成される場合があります。

Message Connector のインストール

Message Connector のインストール、配置、および実行に必要なファイルは、IBM Unica Interact ランタイム・サーバーのインストール済み環境に自動的に含まれています。セクションでは、Message Connector の使用準備に必要なステップを要約します。

Message Connector のインストールと配置には以下のタスクが含まれます。

- オプションで、Message Connector のデフォルト設定を構成する (『Message Connector の構成』を参照)。
- Message Connector トランザクション・データの保管に必要なデータベース・テーブルを作成する (239 ページの『Message Connector テーブルの作成』を参照)。
- Message Connector Web アプリケーションをインストールする (240 ページの『Message Connector の配置および実行』を参照)。
- オープン時およびクリックスルー時に Message Connector オファーを呼び出すために必要なマーケティング・スポット (E メールまたは Web ページなど) に IMG タグと A タグを作成する (241 ページの『Message Connector リンクの作成』を参照)。

Message Connector の構成

Message Connector を配置する前に、特定の環境に合うように、インストール済み環境に含まれる構成ファイルをカスタマイズする必要があります。その場合、Interact

ランタイム・サーバー上の Message Connector ディレクトリー (<Interact_home>/msgconnector/config/MessageConnectorConfig.xml など) にある MessageConnectorConfig.xml という XML ファイルを変更できます。

MessageConnectorConfig.xml ファイルには、必須の構成設定とオプションの構成設定がいくつか含まれています。使用する設定は、特定のインストール済み環境に応じてカスタマイズする必要があります。構成を変更する場合は、以下のステップに従ってください。

1. Web アプリケーション・サーバー上に Message Connector が配置されており、実行されている場合は、Message Connector を配置解除してから作業を続けてください。
2. Interact ランタイム・サーバーで、任意のテキスト・エディターまたは XML エディターで MessageConnectorConfig.xml ファイルを開きます。
3. 必要に応じて、構成設定を変更し、以下の*必須* 設定をインストール済み環境に適したものにします。
   - <interactUrl>。Message Connector ページ・タグを接続する必要があり、Message Connector を実行する Interact ランタイム・サーバーの URL。
   - <imageErrorLink>。オファー・イメージ要求の処理中にエラーが発生した場合の、Message Connector によるリダイレクト先の URL。
   - <landingPageErrorLink>。オファー・ランディング・ページ要求の処理中にエラーが発生した場合の、Message Connector によるリダイレクト先の URL。
   - <audienceLevels>。1 つ以上のオーディエンス・レベル設定セットが含まれており、Message Connector リンクで何も指定されなかった場合にデフォルトのオーディエンス・レベルを指定する、構成ファイルのセクション。少なくとも 1 つのオーディエンス・レベルを構成する必要があります。

   すべての構成設定のより詳しい情報については、『Message Connector の構成設定』を参照してください。
4. 構成変更が完了したら、MessageConnectorConfig.xml ファイルを保存して閉じます。
5. Message Connector の設定と配置を続行します。

Message Connector の構成設定:

Message Connector を構成する場合、Interact ランタイム・サーバー上の Message Connector ディレクトリー (通常は <Interact_home>/msgconnector/config/MessageConnectorConfig.xml) にある MessageConnectorConfig.xml という XML ファイルを変更できます。この XML ファイル内の各構成についてはここで説明します。Message Connector の配置および実行後にこのファイルを変更する場合は、必

ず、Message Connector の配置解除および再配置を行うか、ファイルの変更が完了してからアプリケーション・サーバーを再始動して設定を再ロードしてください。

一般設定

以下の表には、MessageConnectorConfig.xml ファイルの generalSettings セクションに含まれるオプション設定と必須設定のリストが含まれています。

表 20. Message Connector の一般設定

要素	説明	既定値
<interactURL>	Message Connector ページ・タグからの呼び出しを処理する Interact ランタイム・サーバー (Message Connector を実行するランタイム・サーバーなど) の URL。この要素は必須です。	http://localhost:7001/interact
<defaultDateTimeFormat>	デフォルトの日付形式。	MM/dd/yyyy
<log4jConfigFileLocation>	Log4j プロパティー・ファイルの場所。*$MESSAGE_CONNECTOR_HOME* 環境変数が設定されている場合は、その変数が基準になります。それ以外の場合、この値は Message Connector Web アプリケーションのルート・パスが基準になります。	config/ MessageConnectorLog4j.properties

デフォルトのパラメーター値

以下の表には、MessageConnectorConfig.xml ファイルの defaultParameterValues セクションに含まれるオプション設定と必須設定のリストが含まれています。

表 21. Message Connector のメッセージ・パラメーター設定

要素	説明	既定値
<interactiveChannel>	デフォルトのインタラクティブ・チャネルの名前。	
<interactionPoint>	デフォルトのインタラクション・ポイントの名前。	
<debugFlag>	デバッグを有効にするかどうかを決定します。許可される値は true および false です。	false
<contactEventName>	通知されるコンタクト・イベントのデフォルト名。	
<acceptEventName>	通知される承認イベントのデフォルト名。	
<imageUrlAttribute>	オファー・イメージの URL を含むデフォルトのオファー属性名 (Message Connector リンクに何も指定されていない場合)。	
<landingPageUrlAttribute>	クリックスルー・ランディング・ページのデフォルトの URL (Message Connector リンクに何も指定されていない場合)。	

動作設定

以下の表には、MessageConnectorConfig.xml ファイルの behaviorSettings セクションに含まれるオプション設定と必須設定のリストが含まれています。

表 22. Message Connector の動作設定

要素	説明	既定値
<imageErrorLink>	オファー・イメージ要求の処理中にエラーが発生した場合の、コネクターによるリダイレクト先の URL。この設定は必須です。	/images/default.jpg
<landingPageErrorLink>	クリックスルー・ランディング・ページ要求の処理中にエラーが発生した場合の、コネクターによるリダイレクト先の URL。この設定は必須です。	/jsp/default.jsp
<alwaysUseExistingOffer>	キャッシュされたオファーを、既に有効期限が切れている場合でも返す必要があるかどうかを決定します。許可される値は true および false です。	false
<offerExpireAction>	元のオファーが検出されたが、既に有効期限が切れている場合に実行するアクション。許可される値は以下のとおりです。 • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

ストレージ設定

以下の表には、MessageConnectorConfig.xml ファイルの storageSettings セクションに含まれるオプション設定と必須設定のリストが含まれています。

表 23. Message Connector のストレージ設定

要素	説明	既定値
<persistenceMode>	キャッシュ内の新規エントリーをデータベースで保持する場合。許可される値は WRITE-BEHIND (データを最初にキャッシュに書き込み、後でデータベースに更新する場合) および WRITE-THROUGH (データをキャッシュとデータベースに同時に書き込む場合) です。	WRITE-THROUGH
<maxCacheSize>	メモリー・キャッシュ内のエントリーの最大数。	5000
<maxPersistenceBatchSize>	データベースでエントリーを保持する際の最大バッチ・サイズ。	200

表 23. *Message Connector* のストレージ設定 (続き)

要素	説明	既定値
<macCachePersistInterval>	エントリーがデータベースで保持されるまで、キャッシュに入れられる最大時間 (秒単位)。	3
<maxElementOnDisk>	ディスク・キャッシュ内のエントリーの最大数。	5000
<cacheEntryTimeToExpireInSeconds>	有効期限が切れるまで、ディスク・キャッシュで保持するエントリーの最長時間。	60000
<jdbcSettings>	特定の情報を含む XML ファイルのセクション (JDBC 接続が使用されている場合)。これは <dataSourceSettings> セクションと同時に使用することはできません。	デフォルトでは、ローカル・サーバーに構成されている SQL Server データベースに接続するように構成されていますが、このセクションを有効にする場合は、ログインするための実際の JDBC 設定と資格情報を指定する必要があります。
<dataSourceSettings>	特定の情報を含む XML ファイルのセクション (データ・ソース接続が使用されている場合)。これは <jdbcSettings> セクションと同時に使用することはできません。	デフォルトでは、ローカル Web アプリケーション・サーバーに定義されている InteractDS データ・ソースに接続するように構成されています。

オーディエンス・レベル

以下の表には、MessageConnectorConfig.xml ファイルの audienceLevels セクションに含まれるオプション設定と必須設定のリストが含まれています。

audienceLevels 要素は、Message Connector リンクに何も指定されていない場合に使用するデフォルトのオーディエンス・レベルを指定するために、オプションで使用されることに注意してください。以下に例を示します。

```
<audienceLevels default="Customer">
```

この例では、デフォルトの属性値は、このセクションに定義されている audienceLevel の名前と一致します。この構成ファイルには、少なくとも 1 つのオーディエンス・レベルを定義する必要があります。

表 24. *Message Connector* のオーディエンス・レベル設定

要素	要素	説明	既定値
<audienceLevel>		オーディエンス・レベル構成を含む要素。名前属性 (<audienceLevel name="Customer"> など) を指定します。	
	<messageLogTable>	ログ・テーブルの名前。この値は必須です。	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<field>	この audienceLevel の 1 つ以上のオーディエンス ID 項目の定義。	

表 24. *Message Connector* のオーディエンス・レベル設定 (続き)

要素	要素	説明	既定値
	<name>	Interact ランタイムに指定されている、オーディエンス ID 項目の名前。	
	<httpParameterName>	このオーディエンス ID 項目の対応するパラメーター名。	
	<dbColumnName>	このオーディエンス ID 項目の、データベース内の対応する列名。	
	<type>	Interact ランタイムに指定されている、オーディエンス ID 項目のタイプ。string または numeric の値を指定できます。	

Message Connector テーブルの作成

IBM Unica Interact Message Connector を配置する前に、まず、Interact ランタイム・データの保管先のデータベースにテーブルを作成する必要があります。定義されたオーディエンス・レベルごとに 1 つのテーブルを作成します。Interact は、オーディエンス・レベルごとに作成されたテーブルを使用して、Message Connector のトランザクションに関する情報を記録します。

必要なテーブルを作成するには、データベース・クライアントを使用して、該当するデータベースまたはスキーマに対して Message Connector SQL スクリプトを実行します。サポートされるデータベースの SQL スクリプトは、Interact ランタイム・サーバーのインストール時に自動的にインストールされます。Interact ランタイム・テーブルを含む、データベースへの接続の詳細については、「*IBM Unica Interact インストール・ガイド*」にある完成したワークシートを参照してください。

1. データベース・クライアントを起動し、Interact ランタイム・テーブルが現在保管されているデータベースに接続します。
2. <*Interact_home*>/msgconnector/scripts/ddl ディレクトリー (ここで、<*Interact_home*> は、C:¥Unica¥Interact または /Unica/Interact など、Interact ランタイムをインストールしたディレクトリーです) の適切なスクリプトを実行します以下の表に、Message Connector テーブルを手動で作成するために使用できる SQL スクリプトの例をリストします。

表 25. *Message Connector* テーブルを作成するスクリプト

データ・ソース・タイプ	スクリプト名
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

これらは例として提供されているスクリプトであることに注意してください。オーディエンス ID 値によって異なる命名規則または構造を使用する可能性がある

ため、スクリプトを実行する前に変更する必要があります。一般的には、オーディエンス・レベルにそれぞれ専用の 1 つのテーブルを作成するのがベスト・プラクティスです。

テーブルは以下の情報を含めるために作成されます。

表 26. SQL スクリプト例で作成される情報

列名	説明
LogId	この項目のプライマリー・キー。
MessageId	メッセージング・インスタンスごとの固有 ID。
LinkId	電子メディア (E メール・メッセージなど) 内の各リンクの固有 ID。
OfferImageUrl	返されたオファーに関するイメージの URL。
OfferLandingPageUrl	返されたオファーに関するランディング・ページの URL。
TreatmentCode	返されたオファーの処理コード。
OfferExpirationDate	返されたオファーの有効期限の日時。
OfferContactDate	オファーが顧客に返された日時。
AudienceId	電子メディアのオーディエンス ID。

この表については、以下の点に注意してください。

- オーディエンス・レベルに応じて、AudienceId 列がオーディエンス・キーのコンポーネントごとに 1 つになります。
- この表の固有キーは、MessageId、LinkId、および AudienceId を組み合わせて形成されています。

スクリプトの実行が終了した時点で、Message Connector に必要なテーブルが作成されています。

これで、Message Connector Web アプリケーションの配置準備が完了しました。

Message Connector の配置および実行

IBM Unica Interact Message Connector は、サポートされている Web アプリケーション・サーバー上にスタンドアロン Web アプリケーションとして配置されます。

Message Connector を配置する前に、以下のタスクが完了していることを確認してください。

- IBM Unica Interact ランタイム・サーバーをインストールしている必要があります。配置可能な Message Connector アプリケーションはランタイム・サーバーと共に自動的にインストールされ、Interact ホーム・ディレクトリーからの配置準備は整っています。
- インストール済み環境で提供される SQL スクリプトを実行し、Message Connector が使用する Interact ランタイム・データベースで必要なテーブルを作成する必要もあります (239 ページの『Message Connector テーブルの作成』を参照)。

他の IBM Unica アプリケーションを実行する前に Web アプリケーション・サーバーに配置する場合と同様に、オファー配信で使用できるように Message Connector アプリケーションを配置する必要があります。

1. 必要な権限で Web アプリケーション・サーバー管理インターフェースに接続し、アプリケーションを配置します。
2. Web アプリケーション・サーバーの指示に従って、*<Interact_home>*/msgconnector/MessageConnector.war というファイルを展開して実行します。*<Interact_home>* は、Interact ランタイム・サーバーのインストール先の実際のディレクトリー (C:¥Unica¥Interact、または /Unica/Interact など) に置き換えてください。

これで Message Connector を使用できます。Message Connector がオファーを提供するために使用する基本データ (インタラクティブ・チャネルと戦略、フローチャート、オファーなど) を作成するように Interact インストール済み環境を構成したら、Message Connector が受け入れる電子メディアにリンクを作成できます。

Message Connector リンクの作成

Message Connector を使用して、エンド・ユーザーが (E メール・メッセージを開くなどして) 電子メディアと対話する場合はカスタム・オファー・イメージを提供し、エンド・ユーザーがオファーをクリックスルーする場合にはカスタム・ランディング・ページを提供するには、メッセージに埋め込むリンクを作成する必要があります。このセクションでは、それらのリンクの HTML タグ付けの概要を示します。

エンド・ユーザーへの出力メッセージを生成するために使用するシステムに関係なく、Interact ランタイム・サーバーに渡す情報を含む、(属性として HTML タグに指定される) 適切な項目を含めるために HTML タグ付けを行う必要があります。以下のステップに従って、Message Connector メッセージに最低限必要な情報を構成してください。

ここに示す手順は特に Message Connector リンクを含むメッセージに関するものですが、リンクを Web ページまたはその他の電子メディアに追加する場合も同じステップと構成を使用できます。

1. 最低でも以下のパラメーターを指定して、メッセージに表示する IMG リンクを作成します。
   - msgID。このメッセージの固有 ID を示します。
   - linkID。メッセージ内のリンクの固有 ID を示します。
   - audienceID。メッセージの受信者が属するオーディエンスの ID。

     オーディエンス ID が複合 ID である場合は、これらのすべてのコンポーネントをリンクに含める必要があることに注意してください。

   オプション・パラメーターを含めることもできます。このパラメーターには、オーディエンス・レベル、インタラクティブ・チャネル名、インタラクション・ポイント名、イメージのロケーション URL、および Message Connector では特に使用されないユーザー独自のカスタム・パラメーターが含まれます。
2. オプションで、IMG リンクを囲む A リンクを作成します。これにより、ユーザーがイメージをクリックしたときに、ブラウザーはユーザーに対するオファーを

含むページを読み込みます。 A リンクには上記の 3 つのパラメーター (msgID、linkID、および audienceID) に加えて、オプション・パラメーター (オーディエンス・レベル、インタラクティブ・チャネル名、およびインタラクティブ・ポイント名) と Message Connector では特に使用されないカスタム・パラメーターも含める必要があります。 A リンクには Message Connector IMG リンクが含まれる可能性がありますが、必要に応じて、ページに単独で示すこともできることに注意してください。リンクに IMG リンクが含まれている場合は、IMG リンクにそれを囲む A リンクと同じパラメーター・セット (オプション・パラメーターまたはカスタム・パラメーターを含む) を含める必要があります。

3. リンクが正しく定義されると、E メール・メッセージが生成され、送信されます。

使用可能なパラメーターおよびサンプル・リンクについて詳しくは、『「IMG」タグおよび「A」タグの HTTP 要求パラメーター』を参照してください。

「IMG」タグおよび「A」タグの HTTP 要求パラメーター

エンド・ユーザーが Message Connector でエンコードされた IMG タグを含む E メールを開いたため、あるいは、エンド・ユーザーが A タグをクリックスルーしたために、Message Connector が要求を受け取った場合、その要求に含まれるパラメーターは解析され、適切なオファー・データが返されます。このセクションでは、要求 URL (IMG タグ (E メールのオープン時にタグ付きイメージが表示されたときに自動で読み込まれる) または A タグ (E メールを表示しているユーザーが指定されたサイトにリンクされているメッセージをクリックスルーしたときに読み込まれる)) に含めることができるパラメーターのリストを示します。

パラメーター

Message Connector は要求を受け取ると、その要求に含まれるパラメーターを解析します。これらのパラメーターには、次のいずれかまたはすべてが含まれます。

パラメーター名	説明	必須かどうか	既定値
msgId	E メール・E メールまたは Web ページの固有 ID。	はい	なし。これは、タグを含む E メール・メッセージまたは Web ページの固有インスタンスを作成するシステムによって提供されます。
linkId	この E メールまたは Web ページ内のリンクの固有 ID。	はい	なし。これは、タグを含む E メール・メッセージまたは Web ページの固有インスタンスを作成するシステムによって提供されます。
audienceLevel	この通信の受信者が属するオーディエンス・レベル。	いいえ	MessageConnectorConfig.xml ファイルにある audienceLevels 要素にデフォルトとして指定される audienceLevel。
ic	ターゲット・インタラクティブ・チャネル (IC) の名前	いいえ	MessageConnectorConfig.xml ファイルの defaultParameterValues セクションにある interactiveChannel 要素の値 (デフォルトは「interactiveChannel」)。
ip	該当するインタラクション・ポイント (IP) の名前	いいえ	MessageConnectorConfig.xml ファイルの defaultParameterValues セクションにある interactionPoint 要素の値 (デフォルトは「headBanner」)。

パラメーター名	説明	必須かどうか	既定値
offerImageUrl	メッセージ内にある IMG URL のターゲット・オファー・イメージの URL。	いいえ	なし。
offerImageUrlAttr	ターゲット・オファー・イメージの URL を含む、オファー属性の名前	いいえ	MessageConnectorConfig.xml ファイルの defaultParameterValues セクションにある imageUrlAttribute 要素の値。
offerLandingPageUrl	ターゲット・オファーに対応するランディング・ページの URL。	いいえ	なし。
offerLandingPageUrlAttr	ターゲット・オファーに対応するランディング・ページの URL を含む、オファー属性の名前。	いいえ	MessageConnectorConfig.xml ファイルの defaultParameterValues セクションにある landingPageUrlAttribute 要素の値。
contactEvent	コンタクト・イベントの名前。	いいえ	MessageConnectorConfig.xml ファイルの defaultParameterValues セクションにある contactEventName 要素の値 (デフォルトは「contact」)。
responseEvent	承認イベントの名前。	いいえ	MessageConnectorConfig.xml ファイルの defaultParameterValues セクションにある acceptEventName 要素の値 (デフォルトは「accept」)。
debug	デバッグ・フラグ。このパラメーターは、トラブルシューティングのために IBM Unica テクニカル・サポートから指示を受けた場合にのみ、「true」に設定します。	いいえ	MessageConnectorConfig.xml ファイルの defaultParameterValues セクションにある debugFlag 要素の値 (デフォルトは「false」)。
<*audience id*>	このユーザーのオーディエンス ID。このパラメーターの名前は構成ファイルに定義されています。	はい	なし。

Message Connector が認識できない (つまり、上記リストに表示されていない) パラメーターを受け取ると、そのパラメーターは以下の考えられる 2 つの方法のいずれかで処理されます。

- 認識できないパラメーター (例えば、attribute="attrValue" などの「attribute」) が指定されており、「Type」という単語が付加された同じ名前の一致するパラメーター (例えば、attributeType="string" などの「attributeType」)がある場合、Message Connector は一致する Interact パラメーターを作成し、それを Interact ランタイムに渡します。

  Type パラメーターの値は以下のいずれかを指定できます。

  – string
  – numeric
  – datetime

  タイプ「datetime」のパラメーターについて、Message Connector は「Pattern」という単語が付加された同じ名前のパラメーター (「attributePattern」など) も検索します (値の形式が有効な日付/時刻の場合)。例えば、パラメーター attributePattern="MM/dd/yyyy" を指定できます。

「datetime」のパラメーター・タイプを指定しても、一致する日付パターンを指定しないと、Interact サーバー上の Message Connector 構成ファイル (<installation_directory>/msgconnector/config/MessageConnectorConfig.xml にあります) に指定されている値が使用されます。

- 認識できないパラメーターが指定されており、一致する Type 値がない場合、Message Connector はそのパラメーターをターゲット・リダイレクト URL に渡します。

認識できないすべてのパラメーターについて、Message Connector はそれらのパラメーターの処理も保存も行わずに、Interact ランタイム・サーバーに渡します。

Message Connector コードの例

以下の A タグには、E メール・メッセージに表示される可能性のある一連の Message Connector リンクの例が含まれています。

```
<a href="http://www.example.com/MessageConnector/offerClickthru.jsp?msgId=234
   &linkId=1&userid=1&referral=xyz">
 <img src="http://www.example.com/MessageConnector/offerImage.jsp?msgId=234&linkId=1
      &userid=1&incomeCode=3&incomeType=numeric"/>
</a>
```

この例の IMG タグの場合、E メール・メッセージが開かれたときに自動的に読み込まれます。指定されたページからイメージを取得することで、メッセージは固有のメッセージ ID (msgID)、固有のリンク ID (linkID)、および固有のユーザー ID (userid) のパラメーターを、渡す必要がある 2 つの追加パラメーター (incomeCode および incomeType) と共に Interact ランタイムに渡します。

A タグには、E メール・メッセージのオファー・イメージをクリック可能なリンクに変える HREF (ハイパーテキスト参照) 属性が示されます。メッセージのビューアーがイメージを見てすぐにランディング・ページにクリックスルーした場合、固有のメッセージ ID (msgId)、リンク ID (linkId)、およびユーザー ID (userid) が、ターゲット・リダイレクト URL に渡される 1 つの追加パラメーター (referral) と共にサーバーにパススルーされます。

Interact Web Connector について

Interact Web Connector (JavaScript Connector (つまり、JS Connector) ともいう) は、JavaScript コードによる Interact Java API 呼び出しを可能にする、Interact ランタイム・サーバーにおけるサービスを提供します。これにより、Web ページで、Web 開発言語 (Java、PHP、JSP、など) に依存することなく、埋め込み JavaScript コードのみを使用して、リアルタイム・オファーをカスタマイズするために Interact を呼び出すことができます。例えば、Interact で推奨されるオファーを提供する、Web サイトの各ページに JavaScript コードの小さなスニペットを埋め込むことができるため、ページ読み込みごとに Interact API が呼び出され、サイト訪問者の読み込みページにベスト・オファーが確実に表示されます。

サーバー・サイドでページの表示を (PHP や他のサーバー・ベースのスクリプトなどを使用して) プログラム制御できない可能性があるが、訪問者の Web ブラウザーによって実行される、JavaScript コードをページ・コンテンツにまだ埋め込めるページに訪問者へのオファーを表示する場合に、Interact Web Connector を使用します。

ヒント: Interact Web Connector ファイルは、Interact ランタイム・サーバー (*<Interact_home>*/jsconnector ディレクトリー) に自動的にインストールされます。 *<Interact_home>*/jsconnector ディレクトリーには、Web Connector 機能に関する重要な注意事項と詳細を含む ReadMe.txt、および独自のソリューションを開発する際のベースとして使用する Web Connector ソース・コードとサンプル・ファイルがあります。ここで問題を解決するための情報が見つからない場合は、jsconnector ディレクトリーにある詳細情報を参照してください。

ランタイム・サーバーへの Web Connector のインストール

Web Connector のインスタンスは、IBM Unica Interact ランタイム・サーバーと共に自動的にインストールされ、デフォルトで使用可能に設定されます。ただし、Web Connector を構成して使用する前に変更する必要がある設定がいくつかあります。

ランタイム・サーバーにインストールされている Web Connector を使用する前に変更する必要がある設定は、Web アプリケーション・サーバーの構成に追加されます。そのため、以下のステップを完了してから Web アプリケーション・サーバーを再始動する必要があります。

1. Interact ランタイム・サーバーがインストールされている Web アプリケーション・サーバーについて、以下の Java プロパティーを設定します。

   -DUI_JSCONNECTOR_ENABLE_INPROCESS=true

   -DUI_JSCONNECTOR_HOME=*<jsconnectorHome>*

   <jsconnectorHome> を、ランタイム・サーバー上の jsconnector ディレクトリーへのパス (<Interact_installation_directory>/jsconnector) に置き換えます。

   例えば、Windows インストール済み環境では、これは C:¥Unica¥Interact¥jsconnector のようになります。UNIX システムでは、この値には /Unica/Interact/jsconnector などと入力します。

   Java プロパティーの設定方法は、ご使用の Web アプリケーション・サーバーによって異なります。例えば、WebLogic では、以下の例のように startWebLogic.sh ファイルまたは startWebLogic.cmd ファイルを編集して、JAVA_OPTIONS 設定を更新します。

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
   jsconnector"
   ```

   WebSphere Application Server では、管理コンソールの Java 仮想マシン・パネルでこのプロパティーを設定します。

   Java プロパティーの設定方法について詳しくは、ご使用の Web アプリケーション・サーバーの資料を参照してください。

2. この時点で Web アプリケーション・サーバーを始動するか、既に実行されている場合は再始動して、新しい Java プロパティーが使用されていることを確認します。

Web アプリケーション・サーバーの始動プロセスが完了したら、ランタイム・サーバーへの Web Connector のインストールは終了です。次のステップは、http://<*host*>:<*port*>/interact/jsp/WebConnector.jsp (ここで、<*host*> は Interact ランタイム・サーバー名で、<*port*> は Web アプリケーション・サーバーで指定されている、Web Connector がリッスンするポートです) の構成 Web ページへの接続です。

別個の Web アプリケーションとしての Web Connector のインストール

Web Connector のインスタンスは、IBM Unica Interact ランタイム・サーバーと共に自動的にインストールされ、デフォルトで使用可能に設定されます。ただし、この Web Connector を独自の Web アプリケーションとして (例えば、別のシステム上の Web アプリケーション・サーバーに) 配置し、リモート Interact ランタイム・サーバーと通信するように構成できます。

以下の手順は、リモート Interact ランタイム・サーバーへのアクセス権を持つ別個の Web アプリケーションとして Web Connector を配置するプロセスを示したものです。

Web Connector を配置する前に、IBM Unica Interact ランタイム・サーバーをインストールしておく必要があり、また、Interact ランタイム・サーバーへのネットワーク・アクセスが可能な (ファイアウォールでブロックされていない) 別のシステム上に Web アプリケーション・サーバーがなければなりません。

1. Web Connector ファイルを含む jsconnector ディレクトリーを、Interact ランタイム・サーバーから、Web アプリケーション・サーバー (WebSphere Application Server など) が既に構成され、実行されているシステムにコピーします。jsconnector ディレクトリーは、Interact インストール・ディレクトリー (C:¥Unica¥Interact または /Unica/Interact など) にあります。
2. Web Connector インスタンスを配置するシステムで、任意のテキスト・エディターまたは XML エディターを使用して jsconnector/jsconnector.xml ファイルを構成し、interactURL 属性を変更します。

   デフォルトでは、これは http://localhost:7001/interact に設定されていますが、リモート Interact ランタイム・サーバーの URL (http://runtime.example.com:7011/interact など) と一致するように変更する必要があります。

   Web Connector を配置したら、Web インターフェースを使用して、jsconnector.xml ファイルの残りの設定をカスタマイズできます。詳しくは、247 ページの『Web Connector の構成』を参照してください。
3. Web Connector を配置する Web アプリケーション・サーバーについて、以下の Java プロパティーを設定します。

   -DUI_JSCONNECTOR_HOME=<*jsconnectorHome*>

   <*jsconnectorHome*> を、Web アプリケーション・サーバーに jsconnector ディレクトリーをコピーした場所への実際のパスに置き換えます。

例えば、Windows インストール済み環境では、これは C:¥Unica¥Interact¥jsconnector のようになります。UNIX システムでは、この値には /Unica/Interact/jsconnector などと入力します。

Java プロパティーの設定方法は、ご使用の Web アプリケーション・サーバーによって異なります。例えば、WebLogic では、以下の例のように startWebLogic.sh ファイルまたは startWebLogic.cmd ファイルを編集して、JAVA_OPTIONS 設定を更新します。

```
JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
jsconnector"
```

WebSphere Application Server では、管理コンソールの Java 仮想マシン・パネルでこのプロパティーを設定します。

Java プロパティーの設定方法について詳しくは、ご使用の Web アプリケーション・サーバーの資料を参照してください。

4. このステップで Web アプリケーション・サーバーを始動するか、既に実行されている場合は再始動して、新しい Java プロパティーが使用されていることを確認します。

   Web アプリケーション・サーバーの始動プロセスが完了するまで待ってから、作業を続行してください。
5. 必要な権限で Web アプリケーション・サーバー管理インターフェースに接続し、アプリケーションを配置します。
6. Web アプリケーション・サーバーの指示に従って、以下のファイルを配置して実行します。 jsConnector/jsConnector.war

これで Web Connector が Web アプリケーションに配置されました。完全に構成した Interact サーバーを稼働させたら、次のステップで http:// *<host>*: *<port>*/interact/jsp/WebConnector.jsp (ここで、*<host>* は上記のステップで Web Connector を配置した Web アプリケーション・サーバーを実行しているシステムで、*<port>* は Web アプリケーション・サーバーで指定されている、Web Connector がリッスンしているポートです) の Web Connector 構成 Web ページに接続します。

Web Connector の構成

Interact Web Connector の構成設定は jsconnector.xml というファイルに保管されます。このファイルは、Web Connector の配置先のシステム (Interact ランタイム・サーバー自体、または Web アプリケーション・サーバーを実行している別のシステムなど) に保管されます。 jsconnector.xml ファイルは、任意のテキスト・エディターまたは XML エディターを使用して直接編集できますが、Web ブラウザーから Web Connector の「構成」ページを使用すれば、ほとんどすべての使用可能な構成設定をより簡単に構成できます。

Web インターフェースを使用して Web Connector を構成する前に、Web Connector を提供する Web アプリケーションをインストールして配置する必要があります。Interact ランタイム・サーバーには、Interact をインストールして配置する際に、Web Connector のインスタンスが自動的にインストールされます。他の Web アプ

リケーション・サーバーには、 246 ページの『別個の Web アプリケーションとしての Web Connector のインストール』の説明に従って、Web Connector Web アプリケーションをインストールして配置する必要があります。

1. サポートされている Web ブラウザーを開き、以下のような URL を開きます。

   `http://<host>:<port>/interact/jsp/WebConnector.jsp`

   - *<host>* を、ランタイム・サーバーのホスト名または Web Connector の別のインスタンスを配置したサーバーの名前など、Web Connector を実行するサーバーに置き換えます。
   - *<port>* を、Web Connector Web アプリケーションが接続をリッスンするポート番号 (通常は、Web アプリケーション・サーバーのデフォルト・ポートと一致します) に置き換えます。

2. 表示される「構成」ページで、以下のセクションに入力します。

表 27. Web Connector の構成設定の概要:

セクション	設定
基本設定	「基本設定」ページを使用して、タグ付きページをロールアウトするサイト用に Web Connector の全体的な動作を構成します。これらの設定には、サイトの基本 URL、Interact で使用する必要があるサイト訪問者に関する情報、および Web Connector コードでタグ付けする予定のすべてのページに適用する類似設定が含まれます。 詳しくは、 250 ページの『Web Connector 構成の基本オプション』を参照してください。
HTML 表示タイプ (HTML Display Types)	「HTML 表示タイプ (HTML Display Types)」ページを使用して、ページのインタラクション・ポイントごとに指定する HTML コードを決定します。各インタラクション・ポイントで使用するカスケーディング・スタイル・シート (CSS) コード、HTML コード、および Javascript コードのいくつかの組み合わせを含むデフォルト・テンプレート (.flt ファイル) のリストから選択できます。提供されているテンプレートをそのまま使用することも、必要に応じてカスタマイズすることも、独自のテンプレートを作成することもできます。 このページの構成設定は、`jsconnector.xml` 構成ファイルの `interactionPoints` セクションに対応しています。 詳しくは、 251 ページの『Web Connector 構成の HTML 表示タイプ』を参照してください。
拡張ページ (Enhanced Pages)	「拡張ページ (Enhanced Pages)」を使用して、ページ固有の設定を URL パターンにマップします。例えば、ページ・マッピングを設定して、そのマッピングに定義した特定のページ読み込みイベントとインタラクション・ポイントを示す一般的なウェルカム・ページを、「index.htm」というテキストを含む URL で表示させることができます。 このページの構成設定は、`jsconnector.xml` 構成ファイルの `pageMapping` セクションに対応しています。 詳しくは、 253 ページの『Web Connector 構成の拡張ページ』を参照してください。

3. 「基本設定」ページで、サイト全体の設定がインストール済み環境で有効であることを確認し、オプションでデバッグ・モード (問題のトラブルシューティングを行わない場合は推奨されません)、NetInsight Page Tag 統合、およびほとんどのインタラクション・ポイントのデフォルト設定を指定してから、「構成」の下にある「HTML 表示タイプ (HTML Display Types)」リンクをクリックします。
4. 「HTML 表示タイプ (HTML Display types)」ページで、以下のステップに従って、顧客 Web ページにインタラクション・ポイントを定義する表示テンプレートを追加または変更します。

   デフォルトでは、表示テンプレート (.flt ファイル) は *<jsconnector_home>*/conf/html に保管されます。

   a. 開始点として使用、または確認する .flt ファイルをリストから選択するか、「タイプの追加 (Add a Type)」をクリックして、使用する新しい空白のインタラクション・ポイント・テンプレートを作成します。

      テンプレートのコンテンツに関する情報 (ある場合) は、テンプレート・リストの横に表示されます。

   b. オプションで、「**この表示タイプのファイル名 (File name for this display type)**」項目のテンプレート名を変更します。新しいテンプレートの場合は、CHANGE_ME.flt をもっと分かりやすい名前になるように更新してください。

      ここでテンプレートの名前を変更すると、Web Connector は、次回のテンプレートの保存時に、その名前の新規ファイルを作成します。テンプレートは、テキスト本文を変更し、他の項目に移動したときに保存されます。

   c. 必要に応じて、組み込むスタイル・シート (CSS)、JavaScript、および HTML コードなどの、HTML スニペット情報を変更または入力します。実行時に Interact パラメーターで置き換えられる変数を組み込むこともできることに注意してください。例えば、${offer.HighlightTitle} は、インタラクション・ポイントの指定された場所でオファー・タイトルで自動的に置き換えられます。

      CSS、JavaScript、または HTML コードの各ブロックのフォーマット方法を示す場合は、HTML スニペット項目の下に表示される例を使用してください。
5. 必要に応じて「拡張ページ (Enhanced Pages)」ページを使用して、ページ上で特定の URL パターンを処理する方法を決定するページ・マッピングを設定します。
6. 構成プロパティーの設定が終了したら、「**変更のロールアウト (Roll Out the Changes)**」をクリックします。 「**変更のロールアウト (Roll Out the Changes)**」をクリックすると、以下のアクションが実行されます。
   - IBM Unica Interact Web Connector ページ・タグを表示します。これには、Web Connector ページからコピーして、Web ページに挿入できる JavaScript コードが含まれます。
   - Interact サーバー上の既存の Web Connector 構成ファイル (Web Connector のインストール先のサーバーにある jsconnector.xml ファイル) をバックアップし、定義された設定で新しい構成ファイルを作成します。

     バックアップ構成ファイルは、jsconnector.xml.20111113.214933.750-0500 などの *<jsconnector_home>*/conf/archive/jsconnector.xml.*<date>*.*<time>*

(ここで、date ストリングは 20111113 で、タイム・ゾーン・インジケーターを含む time ストリングは 214933.750-0500 です) に保管されます。

これで、Web Connector の構成が完了しました。

構成を変更する場合は、上記の最初のステップに戻り、新しい値で再度実行するか、任意のテキスト・エディターまたは XML エディターで構成ファイル (<Interact_home>/jsconnector/conf/jsconnector.xml) を開き、必要に応じて変更できます。

Web Connector 構成の基本オプション

Web Connector 構成ページの「基本設定」ページを使用して、タグ付きページをロールアウトするサイト用に Web Connector の全体的な動作を構成します。これらの設定には、サイトの基本 URL、Interact で使用する必要があるサイト訪問者に関する情報、および Web Connector コードでタグ付けする予定のすべてのページに適用する類似設定が含まれます。

サイト全体の設定

サイト全体の設定構成オプションは、構成する Web Connector のインストール済み環境全体の動作に影響するグローバル設定です。以下の値を指定できます。

表 28. Web Connector インストール済み環境のサイト全体の設定

設定	説明	jsconnector.xml の同等の設定
Interact API URL	Interact ランタイム・サーバーの基本 URL。 **注:** この設定は、Web Connector が Interact ランタイム・サーバー内で実行されていない (つまり、別個に配置された) 場合にのみ使用されます。	<*interactURL*>
Web Connector URL	クリックスルー URL の生成に使用される基本 URL。	<*jsConnectorURL*>
ターゲット Web サイトのインタラクティブ・チャネル名	このページ・マッピングを表す、Interact サーバーに定義したインタラクティブ・チャネルの名前。	<*interactiveChannel*>
訪問者のオーディエンス・レベル	インバウンド訪問者の Campaign オーディエンス・レベル。Interact ランタイムの API 呼び出しで使用されます。	<*audienceLevel*>
プロファイル・テーブルのオーディエンス ID 項目名	Interact の API 呼び出しで使用されるオーディエンス ID 項目の名前。複数項目のオーディエンス ID では現在サポートされていないことに注意してください。	<*audienceIdField*>
オーディエンス ID 項目のデータ型	Interact の API 呼び出しで使用されるオーディエンス ID 項目のデータ型 (「numeric」または「string」)。	<*audienceIdFieldType*>
セッション ID を表す Cookie 名	セッション ID を含む Cookie の名前。	<*sessionIdCookie*>

表 28. *Web Connector インストール済み環境のサイト全体の設定 (続き)*

設定	説明	jsconnector.xml の同等の設定
訪問者 ID を表す Cookie 名	訪問者 ID を含む Cookie の名前。	<*visitorIdCookie*>

オプション機能

オプション機能の構成オプションは、構成する Web Connector のインストール済み環境のオプションのグローバル設定です。以下の値を指定できます。

表 29. *Web Connector インストール済み環境のオプションのサイト全体の設定*

設定	説明	jsconnector.xml の同等の設定
デバッグ・モードを有効にする	特別なデバッグ・モードを使用するかどうかを (yes または no の回答で) 指定します。この機能を有効にすると、Web Connector から返されるコンテンツに、発生した特定のページ・マッピングを顧客に知らせる「アラート」の Javascript 呼び出しが含まれます。顧客は、アラートを受け取るために <*authorizedDebugClients*> 設定で指定されたファイルへのエントリーが必要です。	<*enableDebugMode*>
クライアントのデバッグが許可されたホストのファイル	デバッグ・モードに適したホストまたは IP (インターネット・プロトコル) アドレスのリストを含むファイルへのパス。クライアントのホスト名または IP アドレスは、収集されるデバッグ情報用に指定されたファイルに表示される必要があります。	<*authorizedDebugClients*>
NetInsight ページ・タグの統合を有効にする	Web Connector がページ・コンテンツの末尾に指定された IBM Unica NetInsight タグを付加する必要があるかどうかを (yes または no の回答で) 指定します。	<enableNetInsightTagging>
NetInsight タグの HTML テンプレート・ファイル	NetInsight タグの呼び出しを統合するために使用される HTML/Javascript テンプレート。一般的には、別のテンプレートを提供するように指示されない限り、デフォルト設定を受け入れる必要があります。	<netInsightTag>

Web Connector 構成の HTML 表示タイプ

「HTML 表示タイプ (HTML Display Types)」ページを使用して、ページのインタラクション・ポイントごとに指定する HTML コードを決定します。各インタラクション・ポイントで使用するカスケーディング・スタイル・シート (CSS) コード、HTML コード、および JavaScript コードのいくつかの組み合わせを含むデフォルト・テンプレート (.flt ファイル) のリストから選択できます。提供されているテン

プレートをそのまま使用することも、必要に応じてカスタマイズすることも、独自のテンプレートを作成することもできます。

注: このページの構成設定は、jsconnector.xml 構成ファイルの interactionPoints セクションに対応しています。

インタラクション・ポイントには、オファー属性を自動的にドロップできるプレースホルダー (ゾーン) も含めることができます。例えば、インタラクション時にそのオファーに割り当てられた処理コードで置き換えられる ${offer.TREATMENT_CODE} を含めることができます。

このページに表示されるテンプレートは、Web Connector サーバーの <Interact_home>/jsconnector/conf/html ディレクトリーに保管されているファイルから自動的に読み込まれます。ここで作成する新規テンプレートもすべてそのディレクトリーに保管されます。

「HTML 表示タイプ (HTML Display Types)」ページを使用して既存のテンプレートのいずれかを表示または変更するには、リストから .flt ファイルを選択します。

「HTML 表示タイプ (HTML Display Types)」ページで新しいテンプレートを作成するには、「**タイプの追加 (Add a Type)**」をクリックします。

テンプレートを作成または変更するために選択したメソッドに関係なく、テンプレート・リストの横に以下の情報が表示されます。

設定	説明	jsconnector.xml の同等の設定
この表示タイプのファイル名	編集するテンプレートに割り当てられている名前。この名前は、Web Connector が稼働しているオペレーティング・システムで有効でなければなりません。例えば、オペレーティング・システムが Microsoft Windows の場合、名前にスラッシュ (/) を使用することはできません。 新しいテンプレートを作成する場合、この項目が CHANGE_ME.flt に事前に設定されます。作業を続行する前に、これを分かりやすい値に変更する必要があります。	*<htmlSnippet>*

設定	説明	jsconnector.xml の同等の設定
HTML スニペット	Web Connector が Web ページ上のインタラクション・ポイントに挿入する必要がある特定のコンテンツ。このスニペットには、HTML コード、CSS フォーマット情報、またはページで実行する JavaScript を含めることができます。 以下の例の場合、これら 3 つのタイプのコンテンツはそれぞれ BEGIN コードと END コードで囲む必要があります。 • ${BEGIN_HTML} <*ご使用の HTML のコンテンツ*> ${END_HTML} • ${BEGIN_CSS} <*ご使用のインタラクション・ポイント固有のスタイル・シート情報*> ${END_CSS} • ${BEGIN_JAVASCRIPT} <*ご使用のインタラクション・ポイント固有の JavaScript コード*> ${END_JAVASCRIPT} ページの読み込み時に自動的に置き換えられる、以下のような定義済み特殊コードをいくつか入力することもできます。 • ${logAsAccept} : 2 つのパラメーター (ターゲット URL、およびオファー承認の識別に使用される TreatmentCode) を取り、それをクリックスルー URL で置き換えるマクロ。 • ${offer.AbsoluteLandingPageURL} • ${offer.OFFER_CODE} • ${offer.TREATMENT_CODE} • ${offer.TextVersion} • $offer.AbsoluteBannerURL} ここにリストされている各オファー・コードは、Interact から返されるオファーを作成するためにマーケティング担当者が使用した、IBM Unica Campaign のオファー・テンプレートに定義されているオファー属性を表します。 Web Connector は、ページ・テンプレートでのコードの設定に役立つ可能性のある多くの追加オプションを提供する、FreeMarker と呼ばれるテンプレート・エンジンを使用することに注意してください。詳しくは、http://freemarker.org/docs/index.htmlを参照してください。	HTML スニペットは jsconnector.xml とは別の独自のファイルにあるため、同等の設定はありません。
特殊コードの例	HTML、CSS、または JAVASCRIPT などのブロックと、特定のオファー・メタデータを参照するために挿入できるドロップ可能なゾーンを識別するコードを含む、特殊コードのタイプ例が含まれます。	同等の設定はありません。

このページの変更内容は、別の Web Connector 構成ページに移動したときに自動的に保存されます。

Web Connector 構成の拡張ページ

「拡張ページ (Enhanced Pages)」を使用して、ページ固有の設定を URL パターンにマップします。例えば、ページ・マッピングを設定して、そのマッピングに定義

した特定のページ読み込みイベントとインタラクション・ポイントを示す一般的なウェルカム・ページを、「index.htm」というテキストを含む受信 URL で表示させることができます。

注: このページの構成設定は、jsconnector.xml 構成ファイルの pageMapping セクションに対応しています。

「拡張ページ (Enhanced Pages)」ページを使用して新しいページ・マッピングを作成するには、「**ページの追加**」リンクをクリックし、マッピングに必要な情報を入力します。

ページ情報

ページ・マッピング用の「ページ情報」構成オプションでは、このマッピングのトリガーとして機能する URL パターンと、このページ・マッピングが Interact によって処理される方法に関する追加設定をいくつか定義します。

設定	説明	jsconnector.xml の同等の設定
URL (次の値を含む)	これは、着信ページ要求における Web Connector のウォッチ対象となる URL パターンです。例えば、要求 URL に「mortgage.htm」が含まれている場合は、それを住宅ローン情報ページと照合できます。	<*urlPattern*>
このページまたはページ・セットのわかりやすい名前	このページ・マッピングの目的を示す、独自の参照の分かりやすい名前 (「住宅ローン情報ページ」など)。	<*friendlyName*>
JavaScript で使用するために JSON データとしてもオファーを返す	Web Connector で、ページ・コンテンツの末尾に JavaScript Object Notation (http://www.json.org/) 形式で未加工のオファー・データを組み込むかどうかを示すドロップダウン・リスト。	<*enableRawDataReturn*>

このページまたはページ・セットへのアクセス時 (オンロード) に発生するイベント

ページ・マッピング用の一連の構成オプションは、このマッピングのトリガーとして機能する URL パターンと、このページ・マッピングが Interact によって処理される方法に関する追加設定をいくつか定義します。

注: このセクションの構成設定は、jsconnector.xml の <pageLoadEvents> セクションに対応しています。

設定	説明	jsconnector.xml の同等の設定
個々のイベント	このページまたはページ・セットで使用可能なイベントのリスト。このリスト内のイベントは Interact に定義したイベントです。ページの読み込み時に発生させる 1 つ以上のイベントを選択します。 Interact API 呼び出しのシーケンスは以下のとおりです。 1. `startSession` 2. 個々のページ読み込みイベントごとの `postEvent` (Interact に個々のイベントを定義した場合) 3. インタラクション・ポイントごとに以下を指定します。 • `getOffers` • `postEvent(ContactEvent)`	<*event*>

このページまたはページ・セットのインタラクション・ポイント (オファーの表示場所)

ページ・マッピング用の一連の構成オプションにより、Interact ページに表示するインタラクション・ポイントの選択が可能になります。

注: このセクションの構成設定は、`jsconnector.xml` の `<pageMapping>` | `<page>` | `<interactionPoints>` セクションに対応しています。

設定	説明	jsconnector.xml の同等の設定
インタラクション・ポイント名チェック・ボックス	構成ファイルに定義された各インタラクション・ポイントは、このページ・セクションに表示されます。インタラクション・ポイントの名前の横にあるチェック・ボックスを選択すると、インタラクション・ポイントで使用可能ないくつかのオプションが表示されます。	<*interactionPoint*>
HTML 要素 ID (Interact によって内部 HTML が設定されます)	このインタラクション・ポイントのコンテンツを受け取る必要がある HTML 要素の名前。例えば、ページに `<div id="welcomebanner">` を指定した場合、この項目には `welcomebanner` (ID 値) を入力します。	<*htmlElementId*>

設定	説明	jsconnector.xml の同等の設定
HTML 表示タイプ	このインタラクション・ポイントで使用する HTML 表示タイプ (HTML スニピット、またはそれ以前に Web Connector 構成ページで定義した .flt ファイル) を選択できるドロップダウン・リスト。	<*htmlSnippet*>
提示するオファーの最大数 (これがカルーセルまたはフリップブックの場合)	このインタラクション・ポイントについて、Web Connector が Interact サーバーから取得する必要のあるオファーの最大数。この項目はオプションであり、ページを再ロードせずに提示されるオファーを定期的に更新するインタラクション・ポイントにのみ適用されます (複数のオファーを 1 つずつ使用できるように取得するカルーセル・シナリオの場合)。	<*maxNumberOfOffers*>
オファーの提示時に発生するイベント	このインタラクション・ポイントについて、通知されるコンタクト・イベントの名前。	<*contactEvent*>
オファーの承認時に発生するイベント	オファー・リンクがクリックされたときに、このインタラクション・ポイントについて、通知される承認イベントの名前。	<*acceptEvent*>
オファーの拒否時に発生するイベント	このインタラクション・ポイントについて、通知される拒否イベントの名前。 **注:** 現時点では、この機能はまだ使用されていません。	<*rejectEvent*>

Web Connector の構成オプション

一般的には、Web Connector のグラフィカル・インターフェースを使用して、Web Connector 設定を構成できます。指定したすべての設定は、`jsconnector/conf` ディレクトリーにある jsconnector.xml と呼ばれるファイルにも保管されます。ここでは、jsconnector.xml 構成ファイルに保存される各パラメーターについて説明します。

パラメーターとその説明

以下のパラメーターは `jsconnector.xml` ファイルに保管され、Web Connector インタラクションで使用されます。これらの設定の変更方法には次の 2 つがあります。

- Web Connector アプリケーションの配置および始動後に自動的に有効になる Web Connector 構成 Web ページを使用する。構成 Web ページを使用するには、Web ブラウザーを使用して `http://<host>:<port>/interact/jsp/WebConnector.jsp` などの URL を開きます。

  管理 Web ページで行った変更の内容は、Web Connector の配置先のサーバーにある `jsconnector.xml` ファイルに保管されます。

- 任意のテキスト・エディターまたは XML エディターを使用して、jsconnector.xml ファイルを直接編集する。この方法を使用する前に、XML タグと値の編集に習熟していることを確認してください。

  注: jsconnector.xml ファイルを手動で編集する場合、Web Connector の管理ページ (http://*<host>*:*<port>*/interact/jsp/jsconnector.jsp にあります) を開き、「**構成の再ロード**」をクリックすることで、これらの設定をいつでも再ロードできます。

以下の表では、jsconnector.xml ファイルに表示されている、設定可能な構成オプションについて説明します。

表 30. Web Connector の構成オプション

パラメーター・グループ	パラメーター	説明
defaultPageBehavior		
	friendlyName	Web Connector の Web 構成ページに表示する URL パターンの人間が読み取れる ID。
	interactURL	Interact ランタイム・サーバーの基本 URL。注: このパラメーターは、Web Connector (jsconnector) サービスが配置済み Web アプリケーションとして実行されている場合にのみ、設定する必要があります。Web Connector が Interact ランタイム・サーバーの一部として自動的に実行されている場合、このパラメーターを設定する必要はありません。
	jsConnectorURL	http://host:port/jsconnector/clickThru などの、クリックスルー URL の生成に使用される基本 URL。
	interactiveChannel	このページ・マッピングを表す、インタラクティブ・チャネルの名前。
	sessionIdCookie	Interact の API 呼び出しで使用されるセッション ID を含む Cookie の名前。
	visitorIdCookie	オーディエンス ID を含む Cookie の名前。
	audienceLevel	Interact ランタイムの API 呼び出しで使用される、インバウンド訪問者のキャンペーン・オーディエンス・レベル。
	audienceIdField	Interact ランタイムの API 呼び出しで使用される audienceId 項目の名前。 **注:** 注: 複数項目のオーディエンス ID では現在サポートされていません。
	audienceIdFieldType	Interact ランタイムの API 呼び出しで使用される、オーディエンス ID 項目 [numeric | string] のデータ型

表 30. Web Connector の構成オプション (続き)

パラメーター・グループ	パラメーター	説明
	audienceLevelCookie	オーディエンス・レベルを含む Cookie の名前。これはオプションです。このパラメーターを設定しない場合、システムは audienceLevel に定義されたものを使用します。
	relyOnExistingSession	Interact ランタイムの API 呼び出しで使用されます。通常、このパラメーターは「true」に設定されます。
	enableInteractAPIDebug	ログ・ファイルへのデバッグ出力を有効にするために、Interact ランタイムの API 呼び出しで使用されます。
	pageLoadEvents	この特定のページを読み込んだ場合に通知されるイベント。このタグ内には、<event>event1</event> などの形式で 1 つ以上のイベントを指定します。
	interactionPointValues	このカテゴリーのすべてのアイテムは、IP 固有のカテゴリーで指定されていない値に代わるデフォルト値として機能します。
	interactionPointValuescontactEvent	この特定のインタラクション・ポイントについて、通知されるコンタクト・イベントのデフォルト名。
	interactionPointValuesacceptEvent	この特定のインタラクション・ポイントについて、通知される承認イベントのデフォルト名。
	interactionPointValuesrejectEvent	この特定のインタラクション・ポイントについて、通知される拒否イベントのデフォルト名。(注: 現時点では、この機能は使用されていません。)
	interactionPointValueshtmlSnippet	このインタラクション・ポイントについて、提供される HTML テンプレートのデフォルト名。
	interactionPointValuesmaxNumberOfOffers	このインタラクション・ポイントについて、Interact から取得されるオファーのデフォルト最大数。
	interactionPointValueshtmlElementId	このインタラクション・ポイントについて、コンテンツを受け取る HTML 要素のデフォルト名。
	interactionPoints	このカテゴリーにはインタラクション・ポイントごとの構成が含まれます。欠落プロパティーについては、システムは interactionPointValues カテゴリーで構成された内容に依存します。
	interactionPointname	インタラクション・ポイント (IP) の名前。
	interactionPointcontactEvent	この特定の IP の、通知されるコンタクト・イベントの名前。

表 30. Web Connector の構成オプション (続き)

パラメーター・グループ	パラメーター	説明
	interactionPointacceptEvent	この特定の IP の、通知される承認イベントの名前。
	interactionPointrejectEvent	この特定の IP の、通知される拒否イベントの名前。(この機能はまだ使用されていないことに注意してください。)
	interactionPointhtmlSnippet	この IP の、提供される HTML テンプレートの名前。
	interactionPointmaxNumberOfOffers	この IP の、Interact から取得されるオファーの最大数
	interactionPointhtmlElementId	このインタラクション・ポイントの、コンテンツを受け取る HTML エレメントの名前。
	enableDebugMode	特別なデバッグ・モードを有効にするためのブール・フラグ (許容値: true または false)。これを true に設定すると、Web Connector から返されるコンテンツに、発生した特定のページ・マッピングを顧客に知らせる「アラート」の JavaScript 呼び出しが含まれます。顧客は、アラートを生成するために、authorizedDebugClients ファイルへのエントリーが必要です。
	authorizedDebugClients	デバッグ・モードに適したホスト名またはインターネット・プロトコル (IP) アドレスを含む、特別なデバッグ・モードで使用されるファイル。
	enableRawDataReturn	Web Connector がコンテンツの末尾に JSON 形式の未加工オファー・データを付加するかどうかを決定するブール・フラグ (許容値: true または false)。
	enableNetInsightTagging	Web Connector がコンテンツの末尾に NetInsight タグを付加するかどうかを決定するブール・フラグ (許容値: true または false)。
	apiSequence	pageTag の呼び出し時の Web Connector による API 呼び出しのシーケンスを示す、APISequence インターフェースの実装を表します。デフォルトでは、この実装で StartSession、pageLoadEvents、getOffers、および logContact というシーケンスが使用されます。最後の 2 つは各インタラクション・ポイントに固有のものです。

表 30. Web Connector の構成オプション (続き)

パラメーター・グループ	パラメーター	説明
	clickThruApiSequence	clickThru の呼び出し時の Web Connector による API 呼び出しのシーケンスを示す、APISequence インターフェースの実装を表します。デフォルトでは、この実装で StartSession および logAccept というシーケンスが使用されます。
	netInsightTag	NetInsight タグの呼び出しを統合するために使用される HTML および JavaScript テンプレートを表します。通常、このオプションを変更する必要はありません。

Web Connector 管理ページの使用

Web Connector には、特定の URL パターンで使用される可能性のある構成の管理およびテストの際に役立ついくつかのツールを提供する管理ページが含まれています。この管理ページを使用して、変更済みの構成を再ロードすることもできます。

管理ページについて

サポートされている Web ブラウザーを使用して、http://*host*:*port*/interact/jsp/jsconnector.jsp を開くことができます。ここで、*host:port* は Web Connector が稼働しているホストの名前および接続をリッスンしているポート (runtime.example.com:7001 など) です。

管理ページは、以下のいずれかの方法で使用できます。

表 31. Web Connector 管理ページのオプション

オプション	目的
構成の再ロード	「**構成の再ロード**」リンクをクリックして、ディスク上に保存された構成変更をメモリーに再ロードします。これは、構成 Web ページを使用せずに、 Web Connector の jsconnector.xml 構成ファイルを直接変更した場合に必要です。
構成の表示 (View Config)	「**構成の表示 (View Config)**」項目に入力した URL パターンに基づいて、Web Connector の構成を表示します。ページの URL を入力して「**構成の表示 (View Config)**」をクリックすると、システムがそのパターン・マッピングに基づいて使用する構成が Web Connector から返されます。一致するものが見つからない場合は、デフォルト構成が返されます。これは、特定のページで正しい構成が使用されているかどうかをテストする場合に便利です。

表 31. Web Connector 管理ページのオプション (続き)

オプション	目的
ページ・タグの実行 (Execute Page Tag)	このページの項目に入力して「**ページ・タグの実行 (Execute Page Tag)**」をクリックすると、URL パターンに基づく pageTag 結果が Web Connector から返されます。これにより、ページ・タグの呼び出しがシミュレートされます。 このツールから pageTag を呼び出す場合と、実際の Web サイトを使用する場合の違いは、この管理ページを使用したときにエラーまたは例外が表示されることです。実際の Web サイトの場合、例外は返されず、Web Connector ログ・ファイルにのみ表示されます。

Web Connector のサンプル・ページ

例として、testPage.html と呼ばれるファイルが Interact Web Connector に組み込まれています。ここには、ページ内で多くの Web Connector 機能がタグ付けされる方法が示されています。便宜上、そのサンプル・ページがここでも示されています。

Web Connector のサンプル HTML ページ

```
<?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: Unica Interact Web Connector Page Tag -->
<!-- Copyright 2011, IBM Corporation All rights reserved. -->
<script language="javascript" type="text/javascript">
//<![CDATA[
   var unicaWebConnectorBaseURL = "http://localhost:7001/interact/pageTag";
   var unicaURLData = "ok=Y";
   try {
    unicaURLData += "&url=" + escape(location.href)
   } catch (err) {}
   try {
    unicaURLData += "&title=" + escape(document.title)
   } catch (err) {}
   try {
    unicaURLData += "&referrer=" + escape(document.referrer)
   } catch (err) {}
   try {
    unicaURLData += "&cookie=" + escape(document.cookie)
   } catch (err) {}
   try {
    unicaURLData += "&browser=" + escape(navigator.userAgent)
   } catch (err) {}
   try {
    unicaURLData += "&screensize=" +
    escape(screen.width + "x" + screen.height)
   } catch (err) {}
   try {
    if (affiliateSitesForUnicaTag) {
        var unica_asv = "";
        document.write("<style id=\"unica_asht1\" type=\"text/css\"> "
     + "p#unica_ashtp a {border:1px #000000 solid; height:100px "
     + "!important;width:100px "
     + "!important; display:block !important; overflow:hidden "
     + "!important;} p#unica_ashtp a:visited {height:999px !important;"
     + "width:999px !important;} <\/style>");
        var unica_ase = document.getElementById("unica_asht1");
        for (var unica_as in affiliateSitesForUnicaTag) {
         var unica_asArr = affiliateSitesForUnicaTag[unica_as];
         var unica_ashbv = false;
         for (var unica_asIndex = 0; unica_asIndex <
     unica_asArr.length && unica_ashbv == false;
     unica_asIndex++)
   {
          var unica_asURL = unica_asArr[unica_asIndex];
          document.write("<p id=\"unica_ashtp\" style=\"position:absolute; "
     + "top:0;left:-10000px;height:20px;width:20px;overflow:hidden; \
   margin:0;padding:0;visibility:visible;\"> \
   <a href=\"" + unica_asURL + "\">" + unica_as + " <\/a><\/p>");
          var unica_ae = document.getElementById("unica_ashtp").childNodes[0];
          if (unica_ae.currentStyle) {
           if (parseFloat(unica_ae.currentStyle["width"]) > 900)
       unica_ashbv = true
          } else if (window.getComputedStyle) {
           if (parseFloat(document.defaultView.getComputedStyle
       (unica_ae, null).getPropertyValue("width")) > 900)
    unica_ashbv = true
          }
```

```
          unica_ae.parentNode.parentNode.removeChild(unica_ae.parentNode)
         }
         if (unica_ashbv == true) {
          unica_asv += (unica_asv == "" ? "" : ";") + unica_as
         }
        }
        unica_ase.parentNode.removeChild(unica_ase);
        unicaURLData += "&affiliates=" + escape(unica_asv)
    }
   } catch (err) {}
   document.write("<script language='javascript' "
     + " type='text/javascript' src='" + unicaWebConnectorBaseURL + "?"
  + unicaURLData + "'><\/script>");
  //]]&gt;
</script>
<style type="text/css">
/*<![CDATA[*/
   .unicainteractoffer {display:none !important;}
   /*]]&gt;*/
</style>
   <title>Sample Interact Web Connector Page</title>
   </head>
   <body>
   <!-- END: Unica Interact Web Connector Page Tag -->
   <!--
################################################################################
end of pageTag paste
################################################################################
-->
   </body>
</html>
```

付録 E. Interact と Intelligent Offer の統合製品の推奨

IBM Unica Interact は IBM Coremetrics Intelligent Offer との統合により、Interact 駆動の製品推奨を提供できます。両製品ともオファーする製品推奨を提供できますが、それぞれ別の方法を使用して行います。Intelligent Offer は、訪問者の Web 上の動作 (協調フィルタリング) を使用して、訪問者と推奨されたオファーの間の相関を作成します。Interact は、ビュー・レベルのオファーよりも、顧客の過去の動作、属性、履歴に基づいており、どのオファーが顧客の動作プロファイルに (デモグラフィックや顧客に関するその他の情報に基づいて) 最も合致するかを学習します。オファーの承認率は、自習を通して予測モデルを作成するのに役立ちます。両製品を最大限に活用すると、Interact は個人プロファイルを使用してオファーを定義できます。これはカテゴリー ID を Intelligent Offer に渡し、人気 (「群衆の知恵」) に基づいて推奨製品を取得し、選択されたオファーの一部として訪問者に表示することができます。これによって、より良い推奨を顧客に提供できるので、いずれかの製品を単独で使用する場合と比べ、結果としてより多くのクリックスルーとより良い成果が得られます。

次のセクションでは、この統合がどのように動作するのか、および提供されるサンプル・アプリケーションを使用して独自のカスタム・オファー統合を作成する方法について説明します。

Interact と Intelligent Offer の統合の概要

このセクションでは、IBM Unica Interact と IBM Coremetrics Intelligent Offer を統合すると、どのように Interact 駆動の製品推奨を提供できるかについて説明します。また、統合を行うプロセスとメカニズムも説明します。

IBM Unica Interact と IBM Coremetrics Intelligent Offer の統合は、Representational state transfer (REST) アプリケーション・プログラミング・インターフェース (API) を介して行われます。これは、Intelligent Offer インストールにより利用できます。適切なカテゴリー ID を指定して REST API 呼び出しを行うと、Interact は推奨製品を取得でき、これを訪問者が見るカスタマイズされたページに表示するオファー情報に組み込むことができます。

訪問者が Web ページの URL (Interact インストール済み環境に含まれるサンプル JSP ページなど) を参照すると、そのページが Interact を呼び出してオファーを取り出します。Interact 内で適正なパラメーターによってオファーが構成されたことを前提とすると、最も簡単なケースでは次のようなステップが生じます。

1. ページのロジックが訪問者のカスタマー ID を識別します。
2. Interact への API 呼び出しが行われ、その顧客用のオファーを生成するために必要な情報が渡されます。
3. 返されるオファーにより、少なくとも 3 つの属性 (オファーのイメージの URL、顧客がクリックスルーしたときのランディング・ページの URL、および推奨する製品を判別するために使用されるカテゴリー ID) を持つ Web ページが提供されます。

4. このカテゴリー ID を使用して Intelligent Offer が呼び出され、推奨製品が取得されます。この製品セットは JSON (JavaScript Object Notation) 形式であり、そのカテゴリーの製品のベストセラー・ランキング順になっています。
5. 次いで、訪問者のブラウザーにオファーと製品が表示されます。

この統合は、オファー推奨と製品推奨を組み合わせるのに役立ちます。例えば、1 つの Web ページ上に 2 つのインタラクション・ポイント、すなわち 1 つはオファー、もう 1 つはそのオファーに一致する推奨を表示させることが考えられます。これを行うには、Web ページから Interact を呼び出し、リアルタイムのセグメント化を行うことにより、最良のオファー (例えば 10% オフの小型装置すべて) を判別します。ページが Interact からオファーを受け取ると、そのオファーにはカテゴリー ID (このサンプルでは小型装置のカテゴリー ID) が含まれています。そして、ページは API 呼び出しを使用して小型装置のカテゴリー ID を Intelligent Offer に渡し、レスポンスとして、そのカテゴリーの人気に基づいた最良の製品推奨を受け取ることになります。

さらに単純なサンプルとしては、Web ページから Interact への呼び出しは単に顧客プロファイルに一致するカテゴリー (例えば高級カトラリー) を見つけるだけというものも考えられます。それから、取得したカテゴリー ID を Intelligent Offer に渡して、カトラリー製品の推奨を受け取ることになります。

統合の前提条件

Intelligent Offer - Interact 統合を使用するには、まずこのセクションに記載する前提条件を満たしていることを確認する必要があります。

以下の前提条件が満たされていることを確認します。

- 「*管理者ガイド*」およびオンライン・ヘルプで説明されている Interact API の使用法を十分理解しておくこと。
- Intelligent Offer 開発者向け資料で説明されている Intelligent Offer REST API を十分理解しておくこと。
- HTML、JavaScript、CSS、および JSON (JavaScript Object Notation) の基礎知識があること。

  要求された製品情報を Intelligent Offer REST API が JSON 形式のデータとして返すので、JSON の知識は重要です。
- Web ページのサーバー・サイド・コーディングを十分理解しておくこと。Interact に付属のデモンストレーション・アプリケーションでは JSP を使用しているためです (ただし JSP は必須ではありません)。
- 有効な Intelligent Offer アカウント、および Interact に製品推奨 (指定したカテゴリーのベストセラー製品や特に人気のある製品) を取得させるよう計画しているカテゴリー ID のリストを用意しておくこと。
- Intelligent Offer REST API リンク (使用する Intelligent Offer 環境の URL) を把握しておくこと。

  使用例については、Interact インストールに同梱されたサンプル・アプリケーションを参照してください。また、詳細については、268 ページの『統合サンプル・プロジェクトの使用』にあるサンプル・コードを参照してください。

Intelligent Offer 統合のためのオファーの構成

Web ページから IBM Coremetrics Intelligent Offer を呼び出して推奨製品を取得できるようにするには、まず、IBM Unica Interact オファーを、Intelligent Offer に渡すために必要な情報を持つように構成する必要があります。

Intelligent Offer とリンクするようにオファーをセットアップするには、まず、以下の状態に整っていることを確認します。

- 使用する Interact ランタイム・サーバーが正しくセットアップされ、稼働中であることを確認します。
- ランタイム・サーバーが Intelligent Offer サーバーとの接続を確立できることを確認します。このとき、使用しているファイアウォールが標準 Web 接続 (ポート 80) の発信の確立を妨げていないことも確認します。

オファーを Intelligent Offer との統合用にセットアップするには、次の手順を実行します。

1. Interact のオファーを作成または編集します。

   オファーの作成と変更について詳しくは、「*IBM Unica Interact*ユーザー・ガイド 」および IBM Unica Campaignの資料を参照してください。
2. オファーに、他の設定と共に以下のオファー属性が含まれていることを確認してください。
   - オファーのイメージにリンクする URL (Uniform Resource Locator)。
   - オファーのランディング・ページにリンクする URL。
   - このオファーに関連付けられた Intelligent Offer カテゴリー ID。

     カテゴリー ID は、Intelligent Offer 構成から手動で取得できます。Interact は直接カテゴリー ID 値を取得できません。

   Interact インストールに含まれているデモンストレーション Web アプリケーションでは、これらのオファー属性は `ImageURL`、`ClickThruURL`、および `CategoryID` と呼ばれています。これらの名前は、オファーが期待する値と Web アプリケーションとが一致している限り、意味がわかりやすい任意のものにすることができます。

   例えば、これらの属性を含む「10PercentOff」というオファーを定義することが考えられます (カテゴリー ID (Intelligent Offer 構成から取得したもの) が「`PROD1161127`」、オファーのクリックスルーの URL が `http://www.example.com/success`、オファー用に表示するイメージの URL が `http://localhost:7001/sampleIO/img/10PercentOffer.jpg` (この場合、Interact ランタイム・サーバーに対してローカルとなる URL))。
3. このオファーを含むようにインタラクティブ・チャネルの処理ルールを定義し、通常どおりインタラクティブ・チャネルを配置します。

これで、Intelligent Offer 統合に必要な情報を持つようにオファーが定義できました。残りの作業は Intelligent Offer から Interact に製品推奨を提供できるようにすることです。これは、適切な API 呼び出しを行うように Web ページを構成することによって達成できます。

統合されたページを訪問者に提示できるように Web アプリケーションを構成するとき、WEB-INF/lib ディレクトリーに次のファイルが含まれていることを確認してください。

- *Interact_Home*/lib/interact_client.jar。Web ページから Interact API への呼び出しを処理するために必要です。
- *Interact_Home*/lib/JSON4J_Apache.jar。Intelligent Offer REST API 呼び出しから返されるデータの処理に必要です。この呼び出しは JSON 形式のデータを返します。

顧客にオファーを提供する方法について詳しくは、『統合サンプル・プロジェクトの使用』を参照してください。

統合サンプル・プロジェクトの使用

すべての Interact ランタイムのインストール済み環境には、Intelligent Offer - Interact 統合プロセスを例示するサンプル・プロジェクトが含まれています。サンプル・プロジェクトは Web ページを作成する完全なエンドツーエンド・デモンストレーションを提供しています。Web ページはカテゴリー ID を含むオファーを呼び出し、次にそれがIntelligent Offer に渡されて、それによりページのインタラクション・ポイントに表示する推奨製品リストが取得されます。

概説

付属のサンプル・プロジェクトは、統合プロセスのテストを行う場合はそのまま使用することができ、またこれを独自のカスタム・ページを開発するための開始点として使用することもできます。サンプル・プロジェクトは以下のファイルにあります。

Interact_home/samples/IntelligentOfferIntegration/MySampleStore.jsp

このファイルは完全に作動する統合プロセスのサンプルを備えているだけでなく、豊富なコメントが含まれていて、使用するインストール済み環境で稼働させるために Interact に何をセットアップすべきか、.jsp ファイル内のどこをカスタマイズすべきか、また、適切にページを配置するにはどうしたらよいかが示されています。

MySampleStore.jsp

便宜のため、MySampleStore.jsp ファイルをここに示します。このサンプルは、Interact の将来のリリースで更新される可能性があります。そのため、必要なサンプルはすべて、開始点として使用するのはご自分のインストール済み環境に含まれているファイルにしてください。

```
<!--
# ********************************************************************************
#  Licensed Materials - Property of IBM
#  Unica Interact
#  (c) Copyright IBM Corporation 2001, 2011.
#  US Government Users Restricted Rights - Use, duplication or disclosure
#  restricted by GSA ADP Schedule Contract with IBM Corp.
# ********************************************************************************

-->


<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page import="java.net.URL,
```

```
    java.net.URLConnection,
    java.io.InputStreamReader,
    java.io.BufferedReader,
    com.unicacorp.interact.api.*,
    com.unicacorp.interact.api.jsoverhttp.*,
    org.apache.commons.json.JSONObject,
    org.apache.commons.json.JSONArray" %>

<%

 /*****************************************************************************
 * This sample jsp program demonstrates integration of Interact and IntelligentOffer.
 *
 * 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 IntelligentOffer 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 : IntelligentOffer 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 Intelligent Offers environment specific properties here...
 *****************************************************************************/

 final String ioURL="http://recs.coremetrics.com/iorequest/restapi";
 final String zoneID="ProdRZ1";
 final String cID="90007517";



 /*****************************************************************************
   *****************************************************************************/

 StringBuilder interactErrorMsg = new StringBuilder();
 StringBuilder intelligentOfferErrorMsg = new StringBuilder();

 // get the customerID if passed in as a parameter
 String cid = request.getParameter("cid");
 if(cid != null)
 {
```

```
 customerId = Integer.parseInt(cid);
}

// call Interact to get offer
Offer offer=getInteractOffer(interactURL,sessionId,interactiveChannel,audienceLevel,
       audienceColumnName,ip,customerId,debug,relyOnExistingSession,interactErrorMsg);

// get specific attributes from the offer (img url, clickthru url, & category id)
String offerImgURL=null;
String offerClickThru=null;
String categoryId="";

if(offer != null)
{
 for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
    {
        if(offerAttribute.getName().equalsIgnoreCase("ImageURL"))
        {
         offerImgURL=offerAttribute.getValueAsString();
        }
        else if(offerAttribute.getName().equalsIgnoreCase("ClickThruURL"))
        {
         offerClickThru=offerAttribute.getValueAsString();
        }
        else if(offerAttribute.getName().equalsIgnoreCase("CategoryID"))
        {
         categoryId=offerAttribute.getValueAsString();
        }
    }
}

// call IO to get products
JSONObject products=getProductsFromIntelligentOffer(ioURL, cID, zoneID, categoryId,
       intelligentOfferErrorMsg);

%>

<html>
 <head>
  <title>My Favorite Store</title>

  <script language="javascript" type="text/javascript">
   var unicacarousel=(function(){var g=false;var h;var j=0;var k=0;var l=0;var m=40;
         var n=new Array(0,2,6,20,40,60,80,88,94,97,99,100);var o=function(a){var b=a.parentNode;
         h=b.getElementsByTagName("UL")[0];var c=h.getElementsByTagName("LI");j=c[0].offsetWidth;
         k=c.length;l=Math.round((b.offsetWidth/j));unicacarousel.recenter()};var p=function(a)
         {var b=parseFloat(h.style.left);if(isNaN(b))b=0;for(var i=0;i<n.length;i++)
         {setTimeout("unicacarousel.updateposition("+(b+(a*(n[i]/100)))+");",((i*m)+50))}
         setTimeout("unicacarousel.recenter();",((i*m)+50))};return{gotonext:function(a,b)
         {if(!g){o(a);g=true;p((-1*b*j))}},gotoprev:function(a,b){if(!g){o(a);g=true;p((b*j))}},
         updateposition:function(a){h.style.left=a+"px"},recenter:function(){var a=parseFloat(h.style.left);
         if(isNaN(a))a=0;var b=j*Math.round(((l-k)/2));var c=Math.abs(Math.round((b-a)/j));
         if(a<b){var d=h.getElementsByTagName("LI");var e=new Array();
         for(var i=0;i<c;i++){e[e.length]=d[i]}for(var i=0;i<e.length;i++)
         {h.insertBefore(e[i],null)}unicacarousel.updateposition(b)}else
         if(a>b){var d=h.getElementsByTagName("LI");var e=new Array();
         for(var i=0;i<c;i++){e[e.length]=d[d.length-c+i]}var f=d[0];
         for(var i=0;i<e.length;i++){h.insertBefore(e[i],f)}unicacarousel.updateposition(b)}g=false}})();
  </script>

  <style type="text/css">
 .unicaofferblock_container {width:250px; position:relative; display:block;
                             text-decoration:none; color:#000000; cursor: pointer;}
 .unicaofferblock_container .unicateaserimage {margin:0px 0.5em 0.25em 0px; float:left;}
 .unicaofferblock_container .unicabackgroundimage {position:absolute; top:0px; left:0px;}
 .unicaofferblock_container .unicabackgroundimagecontent {width:360px; height:108px;
                     padding:58px 4px 4px 20px; position:relative; top:0px;}
 .unicaofferblock_container h4 {margin:0px; padding:0px; font-size:14px;}

 .unicacarousel {width:588px; position:relative; top:0px;}
 .unicacarousel_sizer {width:522px; height:349px; margin:0px 33px; padding:0;
                       overflow:hidden; position:relative;}
 .unicacarousel_rotater {height:348px; width:1000px; margin:0 !important;
                         padding:0; list-style:none; position:absolute; top:0px;
                         left:0px;}
 .unicacarousel li {width:167px; height:349px; float:left; padding:0 4px;
                    margin:0px !important; list-style:none !important;
                    text-indent:0px !important;}
 .unicacarousel_gotoprev, .unicacarousel_gotonext {width:18px; height:61px;
```

```
                    top:43px; background:url(../img/carouselarrows.png) no-repeat;
                   position:absolute; z-index:2; text-align:center; cursor:pointer;
                   display:block; overflow:hidden; text-indent:-9999px;
                   font-size:0px; margin:0px !important;}
 .unicacarousel_gotoprev {background-position:0px 0; left:0;}
 .unicacarousel_gotonext {background-position:-18px 0; right:0;}

  </style>


 </head>

 <body>

  <b>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
 *   Intelligent Offer
 *****************************************************************************/

 /*****************************************************************************
 * Call IntelligentOffer 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());
    }
%>
```

IBM Unica 技術サポートへの連絡

ドキュメンテーションを参照しても解決できない問題があるなら、指定されているサポート窓口を通じて IBM Unica 技術サポートに電話することができます。このセクションの情報を使用するなら、首尾よく効率的に問題を解決することができます。

サポート窓口が指定されていない場合は、IBM Unica 管理者にお問い合わせください。

収集する情報

IBM Unica 技術サポートに連絡する前に、以下の情報を収集しておいてください。

- 問題の性質の要旨。
- 問題発生時に表示されるエラー・メッセージの詳細な記録。
- 問題を再現するための詳しい手順。
- 関連するログ・ファイル、セッション・ファイル、構成ファイル、およびデータ・ファイル。
- 「システム情報」の説明に従って入手した、製品およびシステム環境に関する情報。

システム情報

IBM Unica 技術サポートに電話すると、実際の環境に関する情報について尋ねられることがあります。

問題が発生してもログインは可能である場合、情報の大部分は「バージョン情報」ページで入手できます。そのページには、インストールされている IBM Unica のアプリケーションに関する情報が表示されます。

「バージョン情報」ページは、**「ヘルプ」>「バージョン情報」**を選択することにより表示できます。「バージョン情報」ページを表示できない場合、どの IBM Unica アプリケーションについても、そのインストール・ディレクトリーの下にある `version.txt` ファイルを表示することにより、各アプリケーションのバージョン番号を入手できます。

IBM Unica 技術サポートの連絡先情報

IBM Unica 技術サポートとの連絡を取る方法については、IBM Unica 製品技術サポートの Web サイト (http://www.unica.com/about/product-technical-support.htm) を参照してください。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
170 Tracer Lane
Waltham, MA 02451
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することが

できます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

IBM、IBM ロゴ、および ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、『www.ibm.com/legal/copytrade.shtml』 をご覧ください。

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21