V 10 R 1
2017 年 11 月

IBM Interact 管理员指南

注意

在使用本信息及其支持的产品前，请先阅读第 345 页的『声明』中的信息。

此版本适用于 IBM Interact V10.1.0 及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

第 1 章 管理 IBM Interact 1

Interact 关键概念 1
受众级别 1
设计环境 2
事件 2
交互式渠道 3
交互式流程图 3
交互点 3
商品 3
概要文件 4
运行时环境 4
运行时会话 4
接触点 4
处理规则 4
Interact 体系结构 5
Interact 网络注意事项 6
Interact 服务器端口和网络安全 6
登录到 IBM Marketing Software 8

第 2 章 配置用户 11

配置运行时环境用户 11
配置设计环境用户 11
设计环境许可权示例 12

第 3 章 管理 Interact 数据源 15

Interact 数据源 15
数据库和应用程序 15
Campaign 系统表 16
运行时表 17
测试运行表 18
覆盖用于动态创建表的缺省数据类型 19
覆盖缺省数据类型 19
动态创建的表的缺省数据类型 20
概要文件数据库 20
学习表 22
跨会话响应跟踪的联系历史记录 22
运行数据库脚本以启用 功能 23
关于联系和响应历史记录跟踪 23
联系和响应类型 23
其他响应类型 24
运行时环境登台表与 Campaign 历史记录表之间的映射 26
为联系和响应历史记录模块配置 JMX 监视 . . . 31
关于跨会话响应跟踪 31
跨会话响应跟踪数据源配置 32
为跨会话响应跟踪配置联系和响应历史记录表 . . 32
启用跨会话响应跟踪 34
跨会话响应商品匹配 35
将数据库装入实用程序与运行时环境配合使用 . . . 37
对运行时环境启用数据库装入实用程序 38
事件模式 ETL 过程 38
运行独立 ETL 过程 39
停止独立 ETL 过程 40

第 4 章 商品服务 43

商品资格 43
生成候选商品的列表 43
计算市场营销分数 44
影响学习 45
禁止商品 45
启用商品禁止 45
商品禁止表 46
全局商品和个别分配 46
定义缺省单元代码 46
定义处理规则中未使用的商品 47
关于全局商品表 47
分配全局商品 47
全局商品表 48
关于分数覆盖表 49
配置分数覆盖 50
分数覆盖表 50
Interact 内置学习概述 52
Interact 学习模块 52
启用学习模块 53
学习属性 54
定义学习属性 55
定义动态学习属性 55
将运行时环境配置为识别外部学习模块 56

第 5 章 了解 Interact API. 59

Interact API 数据流 59
简单交互规划示例 63
设计 Interact API 集成 66
要考虑的问题 67

第 6 章 管理 IBM Interact API 69

语言环境和 Interact API 69
关于 JMX 监视 69
将 Interact 配置为使用 RMI 协议进行 JMX 监视 69
将 Interact 配置为使用 JMXMP 协议进行 JMX 监视 70
将 Interact 配置为使用 jconsole 脚本进行 JMX 监视 70
JMX 属性 71
JMX 操作 80

第 7 章 IBM Interact Java、SOAP 和 REST API 的类和方法 81

Interact API 类 81
基于 HTTP 的 Java 序列化先决条件 81
SOAP 先决条件 82
REST 先决条件 82

API JavaDoc 83
API 示例 83
处理会话数据 83
关于 InteractAPI 类 84
endSession 84
executeBatch 85
getInstance 86
getOffers 87
getOffersForMultipleInteractionPoints 88
getProfile 90
getVersion 91
postEvent 92
setAudience 94
setDebug 95
startSession 96
保留参数 100
关于 AdvisoryMessage 类 101
getDetailMessage 102
getMessage 102
getMessageCode 102
getStatusLevel 103
关于 AdvisoryMessageCode 类 103
咨询消息代码 103
关于 BatchResponse 类 105
getBatchStatusCode 105
getResponses 106
关于 Command 接口 106
setAudienceID 107
setAudienceLevel 107
setDebug 108
setEvent 109
setEventParameters 109
setGetOfferRequests 110
setInteractiveChannel 111
setInteractionPoint 111
setMethodIdentifier 112
setNumberRequested 113
setRelyOnExistingSession 113
关于 NameValuePair 接口 113
getName 113
getValueAsDate 114
getValueAsNumeric 114
getValueAsString 115
getValueDataType 115
setName 116
setValueAsDate 116
setValueAsNumeric 116
setValueAsString 117
setValueDataType 117
关于 Offer 类 118
getAdditionalAttributes 118
getDescription 119
getOfferCode 119
getOfferName 119
getScore 120
getTreatmentCode 120
关于 OfferList 类 121
getDefaultString 121
getRecommendedOffers 121
关于 Response 类 122
getAdvisoryMessages 122
getApiVersion 122
getOfferList 123
getAllOfferLists 123
getProfileRecord 124
getSessionID 124
getStatusCode 125

第 8 章 IBM Interact JavaScript API 的类和方法 127

JavaScript 先决条件 127
处理会话数据 127
使用回调参数 128
关于 InteractAPI 类 128
startSession 128
getOffers 133
getOffersForMultipleInteractionPoints 133
setAudience 135
getProfile 136
endSession 136
setDebug 137
getVersion 137
executeBatch 138
JavaScript API 示例 138
示例响应 JavaScript 对象 onSuccesss 146

第 9 章 关于 ExternalCallout API . . . 147

IAffiniumExternalCallout 接口 147
添加 Web Service 以与 EXTERNALCALLOUT 宏配合使用 147
getNumberOfArguments 148
getValue 148
initialize 149
shutdown 149
外部调出 API 示例 150
IInteractProfileDataService 接口 151
添加数据源以与 Profile Data Services 配合使用 151
IParameterizableCallout 接口 152
initialize 152
shutdown 152
ITriggeredMessageAction 接口 153
getName 153
setName 153
IChannelSelector 接口 153
selectChannels 153
IDispatcher 接口 154
dispatch 154
IGateway 接口 155
deliver 155
validate 155

第 10 章 IBM Interact 实用程序. . . . 157
运行部署实用程序 (runDeployment.sh/.bat) . . . 157

第 11 章 关于学习 API 161
将运行时环境配置为识别外部学习模块 162
ILearning 接口 162
initialize 162
logEvent 163
optimizeRecommendList. 163
reinitialize 164
shutdown 165
IAudienceID 接口 165
getAudienceLevel 165
getComponentNames. 165
getComponentValue 166
IClientArgs 166
getValue 166
IInteractSession. 166
getAudienceId 166
getSessionData 166
IInteractSessionData 接口 167
getDataType. 167
getParameterNames 167
getValue 167
setValue 167
ILearningAttribute. 168
getName 168
ILearningConfig 168
ILearningContext 168
getLearningContext 168
getResponseCode 169
IOffer 169
getCreateDate 169
getEffectiveDateFlag 169
getExpirationDateFlag 169
getOfferAttributes 170
getOfferCode 170
getOfferDescription 170
getOfferID 170
getOfferName 170
getUpdateDate 170
IOfferAttributes 171
getParameterNames 171
getValue 171
IOfferCode 接口 171
getPartCount 171
getParts 171
LearningException. 172
IScoreOverride 172
getOfferCode 172
getParameterNames 172
getValue 173
ISelectionMethod 173
ITreatment 接口 173
getCellCode 173
getCellId 173
getCellName 173
getLearningScore 174
getMarketerScore 174
getOffer 174
getOverrideValues. 174
getPredicate 175
getPredicateScore 175
getScore 175
getTreatmentCode 175
setActualValueUsed 175
学习 API 示例 176

第 12 章 IBM Interact WSDL 179

第 13 章 Interact 运行时环境配置属性 187
Interact | general 187
Interact | general | learningTablesDataSource 187
Interact | general | prodUserDataSource. . . 189
Interact | general | systemTablesDataSource 190
Interact | general | testRunDataSource . . . 194
Interact | general | contactAndResponseHistoryDataSource. . . . 195
Interact | general | idsByType 196
Interact | flowchart 197
Interact | flowchart | ExternalCallouts | [ExternalCalloutName] 199
Interact | flowchart | ExternalCallouts | [ExternalCalloutName] | Parameter Data | [parameterName] 199
Interact | monitoring 200
Interact | monitoring | activitySubscribers . . 200
Interact | profile 202
Interact | profile | Audience Levels | [AudienceLevelName] 203
Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL . 206
Interact | profile | Audience Levels | [AudienceLevelName | Profile Data Services | [DataSource]. 208
Interact | offerserving 209
Interact | offerserving | Built-in Learning Config. 211
Interact | offerserving | Built-in Learning Config | Parameter Data | [parameterName] . 213
Interact | offerserving | External Learning Config. 214
Interact | offerserving | External Learning Config | Parameter Data | [parameterName] . 215
Interact | offerserving | Constraints 215
Interact | services 216
Interact | services | contactHist. 216
Interact | services | contactHist | cache. . . 217
Interact | services | contactHist | fileCache 217
Interact | services | defaultedStats. 217
Interact | services | defaultedStats | cache 218
Interact | services | eligOpsStats 218

Interact | services | eligOpsStats | cache . . 219
Interact | services | eventActivity 219
Interact | services | eventActivity | cache 219
Interact | services | eventPattern 220
Interact | services | eventPattern | userEventCache 221
Interact | services | eventPattern | advancedPatterns 221
Interact | services | customLogger. 223
Interact | services | customLogger | cache 224
Interact | services | responseHist 224
Interact | services | responseHist | cache . . 225
Interact | services | response Hist | responseTypeCodes 225
Interact | services | responseHist | fileCache 226
Interact | services | crossSessionResponse . . 226
Interact | services | crossSessionResponse | cache 227
Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode 228
Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode 229
Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode 230
Interact | services | threadManagement | contactAndResponseHist 230
Interact | services | threadManagement | allOtherServices 231
Interact | services | threadManagement | flushCacheToDB 232
Interact | services | threadManagement | eventHandling 233
Interact | services | configurationMonitor . . 234
Interact | cacheManagement 235
Interact | cacheManagement | Cache Managers. 235
Interact | caches 239
Interact | triggeredMessage 245
Interact | triggeredMessage | offerSelection 246
Interact | triggeredMessage | dispatchers . . 246
Interact | triggeredMessage | gateways | <gatewayName> 248
Interact | triggeredMessage | channels . . . 249
Interact | activityOrchestrator. 251
Interact | activityOrchestrator | receivers . . 251
Interact | activityOrchestrator | gateways . . 252
Interact | ETL | patternStateETL 252
Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS. . . . 253
Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS 255
Interact | ETL | patternStateETL | <patternStateETLName> | Report 256

第 14 章 Interact 设计环境配置属性 259

Campaign | partitions | partition[n] | reports 259
Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking. 261
Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource] . . 265
Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings 265
Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings 266
Campaign | partitions | partition[n] | Interact | report 267
Campaign | partitions | partition[n] | Interact | learning 267
Campaign | partitions | partition[n] | Interact | learning | learningAttributes | [learningAttribute]. 270
Campaign | partitions | partition[n] | Interact | deployment 270
Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] 271
Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL] 271
Campaign | partitions | partition[n] | Interact | flowchart. 271
Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers . . 272
Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL. . . 273
Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride . . 273
Campaign | partitions | partition[n] | server | internal 273
Campaign | monitoring 277
Campaign | partitions | partition[n] | Interact | outboundChannels 279
Campaign | partitions | partition[n] | Interact | outboundChannels | Parameter Data 279
Campaign | partitions | partition[n] | Interact | Simulator. 279

第 15 章 客户机端的实时商品个性化 281

关于 Interact 消息连接器 281
安装消息连接器. 282
创建消息连接器链接 287
关于 Interact Web 连接器 290
在运行时服务器上安装 Web 连接器 290
作为单独 Web 应用程序来安装 Web 连接器 291
配置 Web 连接器 292
使用 Web 连接器的"管理"页面 302
Web 连接器页面样本. 303

第 16 章 Interact 与 Digital Recommendations 集成 307
Interact 与 Digital Recommendations 集成的概述 307
集成先决条件 308
配置商品以进行 Digital Recommendations 集成 308
使用集成样本项目 309

第 17 章 Interact 与 Digital Data Exchange 集成 315
先决条件 315
通过 IBM Digital Data Exchange 将 IBM Interact 与 Web 站点集成 315
Digital Data Exchange 中的 Interact 标记 . . . 316
结束会话 317
获取商品 317
装入库. 317
发布事件 318
设置受众 318
启动会话 318
示例标记设置 319
验证集成配置 323

第 18 章 配置触发式消息的网关 325
使用 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 325
使用 IBM Interact Outbound Gateway for IBM Universal Behavior Exchange 332
使用 IBM Interact Outbound Gateway for IBM Mobile Push Notification 336
使用 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 338
为网关集成添加分派器 338
为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 添加网关 . 339
为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 添加渠道处理程序. 341
为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 添加出站渠道 341
为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 配置交易邮件 341

在与 IBM 技术支持联系之前 343

声明 345
商标 346
隐私策略和使用条款注意事项 347

第 1 章 管理 IBM Interact

管理 Interact 时，应该配置并维护用户和角色、数据源以及可选产品功能部件。还应该监视并维护设计时环境和运行时环境。特定于产品的应用程序编程接口 (API) 可供您使用。

管理 Interact 包括若干任务。这些任务可能包括但不限于：

- 维护用户和角色
- 维护数据源
- 配置 Interact 可选商品服务功能
- 监视和维护运行时环境性能

在您开始管理 Interact 之前，可以熟悉一些有关 Interact 的工作方式的关键概念，以使您的任务更加轻松。后面各节描述了与 Interact 相关联的管理任务。

管理指南的第二部分描述了 Interact 提供的 API：

- Interact API
- ExternalCallout API
- 学习 API

Interact 关键概念

IBM® Interact 是交互式引擎，用于将个性化市场营销商品的目标指向各种受众。

本部分描述了在处理 Interact 之前应了解的某些关键概念。

受众级别

受众级别是营销活动可以为其目标的标识集合。您可以定受众级别以将营销活动的适当受众集合作为目标。

例如，一组营销活动可使用受众级别"家庭"、"潜在客户"、"客户"和"帐户"。其中每个级别表示可用于营销活动的市场营销数据的特定视图。

通常，受众级别按分层组织。使用以上示例：

- 家庭位于层次结构的顶部，每个家庭可以包含多个客户以及一个或多个潜在客户。
- 客户在层次结构中家庭的下一层，且每个客户可具有多个帐户。
- 帐户在分层结构的底层。

企业到企业环境中存在其他更复杂的受众层次结构示例，其中，可能需要存在企业、公司、部门、组、个人、帐户等等受众级别。

这些受众级别可能在彼此间具有不同关系，例如，一对一、多对一或多对多。通过定义受众级别，可以在 Campaign 中表示这些概念，从而用户可管理这些不同的受众之间的关系，以定位目标。例如，尽管每个家庭可能有多个潜在客户，可能想要对每个家庭将邮件限制为发送到一个潜在客户。

设计环境

使用设计环境来配置各种 Interact 组件，并将这些组件部署到运行时环境。

您将在设计环境中完成大多数 Interact 配置。在设计环境中定义事件、交互点、智能细分市场和处理规则。配置这些组件之后，将它们部署到运行时环境。

设计环境中安装了 Campaign Web 应用程序。

事件

事件是访问者执行的操作，会触发运行时环境中的操作。事件的示例可以是：将访问者放入细分市场、呈现商品或记录数据。

将首先在交互式渠道中创建事件，然后使用 `postEvent` 方法通过对 Interact API 的调用来触发。事件可以导致 Interact 设计环境中定义的下列一个或多个操作：

- **触发重新细分**：通过使用访问者会话中的当前数据，运行时环境将再次运行与交互式渠道相关联的当前受众级别的所有交互式流程图。

  设计交互时，除非指定特定的流程图，否则重新细分操作将再次运行与此交互式渠道相关联的前受众级别的所有交互式流程图，并且所有商品请求都将一直等待，直至所有流程图完成为止。在单个访问中过度重新细分可能会对接触点的绩效产生客户可见的影响。

  将重要的新数据添加至运行时会话对象之后，例如，来自 Interact API 发出的请求（如更改受众）的新数据或者客户操作（如将新项目添加至愿望列表或购物车），将客户放置在新细分市场中。
- **记录商品联系人**：运行时环境会对推荐的商品进行标记，以便数据库服务将这些商品记录到联系人历史记录中。

  对于 Web 集成，在您请求商品的调用中记录商品联系人，从而最大程度地减少接触点与运行时服务器之间的请求数。

  如果接触点不返回 Interact 向访问者呈现的商品的处理代码，那么运行时环境会记录最近推荐商品的列表。
- **记录商品接受**：运行时环境会对所选商品进行标记，以便数据库服务将这些商品记录到响应历史记录中。
- **记录商品拒绝**：运行时环境会对所选商品进行标记，以便数据库服务将这些商品记录到响应历史记录中。
- **触发用户表达式**：表达式操作是您可使用 Interact 宏进行定义的操作，包括函数、变量和运算符（包含 EXTERNALCALLOUT）。可以将表达式的返回值分配给任何概要文件属性。

  当您单击"触发用户表达式"旁边的"编辑"图标时，将显示标准"用户表达式"对话框，并且您可使用此对话框来指定受众级别、要将结果分配至的可选字段名以及表达式本身的定义。
- **触发器事件**：您可以使用"触发器事件"操作来输入要通过此操作触发的事件名称。如果您输入已定义的事件，那么当运行此操作时将触发该事件。如果您输入的事件名称不存在，此操作将导致使用指定的操作来创建该事件。

还可使用事件来触发 postEvent 方法定义的操作，包括将数据记录到表、在学习中包含数据或触发个别流程图。

可以将事件组织到类别中，以便于在设计环境中进行参考。类别在运行时环境中不具有功能上的作用。

交互式渠道

在 Interact 中使用交互式渠道以与交互式市场营销涉及的所有对象、数据和服务器资源协调。

交互式渠道为 Campaign 中接触点的表示，其中接口的方法为交互式对话框。此软件表示法用于协调交互式市场营销中涉及的所有对象、数据和服务器资源。

交互式渠道是一个工具，用于定义交互点和事件。还可以从此交互式渠道的"分析"选项卡访问交互式渠道的报告。

交互式渠道还包含生产运行时和登台服务器分配。如果您仅具有一组生产运行时和登台服务器，您可以创建多个交互式渠道以组织您的事件和交互点，或者按面向客户的系统来划分事件和交互点。

交互式流程图

使用交互式流程图将客户划分到细分市场，并为细分市场分配概要文件。

交互式流程图与 Campaign 批处理流程图相关，但又存在一些不同。交互式流程图与批处理流程图的主要功能相同：将客户划分到称为"细分"的组中。但是，对于交互式流程图，这些组是智能细分市场。行为事件或系统事件指示需要对访问者进行重新细分时，Interact 使用这些交互式流程图来将概要文件分配给细分市场。

交互式流程图包含批处理流程图过程的子集，还包含少量特定于交互式流程图的进程。

注：仅可在 Campaign 会话中创建交互式流程图。

交互点

交互点是指您希望在接触点中呈现商品的位置。

当运行时环境没有其他合格内容要呈示时，交互点会包含缺省填充内容。交互点可组织为区域。

商品

一个商品表示单条市场营销消息，可通过各种方式交付。

在 Campaign 中，您会创建可在一个或多个营销活动中使用的商品。

可在以下情况下重复使用商品：

- 在不同营销活动中
- 在不同时间点
- 对于不同人员组（单元）
- 通过商品的参数化字段作为不同"版本"提供

将商品分配到向访问者展示的接触点中的交互点。

概要文件

概要文件是一组由运行时环境使用的客户数据。此数据可以是客户数据库中可用客户数据的子集、实时收集的数据或者这两者的组合。

此客户数据用于以下目的：

- 在实时交互场景中将客户分配到一个或多个智能细分市场。

  需要针对要根据其进行细分的各个受众级别的概要文件数据集合。例如，如果要按位置来进行细分市场划分，那么您可能仅包含所具有的所有地址信息中的客户邮政编码。
- 个性化商品
- 作为属性来跟踪以进行学习

  例如，可以配置 Interact 来监视访问者的婚姻状态以及每种状态的访问者接受特定商品的数量。然后，运行时环境可使用此信息来优化商品选择。

对于运行时环境，此数据为只读。

运行时环境

运行时环境连接到接触点并执行交互。运行时环境可包含连接到某一接触点的一个或多个运行时服务器。

运行时环境将部署自设计环境的信息与 Interact API 组合使用以向您的接触点呈现商品。

运行时会话

运行时会话存在于运行时服务器上，以供每个访问者访问接触点时使用。此会话会保存访问者的所有数据，运行时环境使用这些数据来将访问者分配到细分市场并建议商品。

使用 `startSession` 调用时创建运行时会话。

接触点

接触点为可在其中与客户交互的应用程序或位置。接触点可为客户在其中发起联系（"入站"交互）的渠道或联系客户（"出站"交互）的渠道。

常见示例是 Web 站点和呼叫中心应用程序。使用 Interact API，可以将 Interact 与接触点集成，以基于客户在接触点中的操作向客户呈示商品。接触点也称为面向客户的系统 (CFS)。

处理规则

处理规则将商品分配给智能细分市场。这些分配由您在处理规则中与商品相关联的自定义区域进行进一步限制。

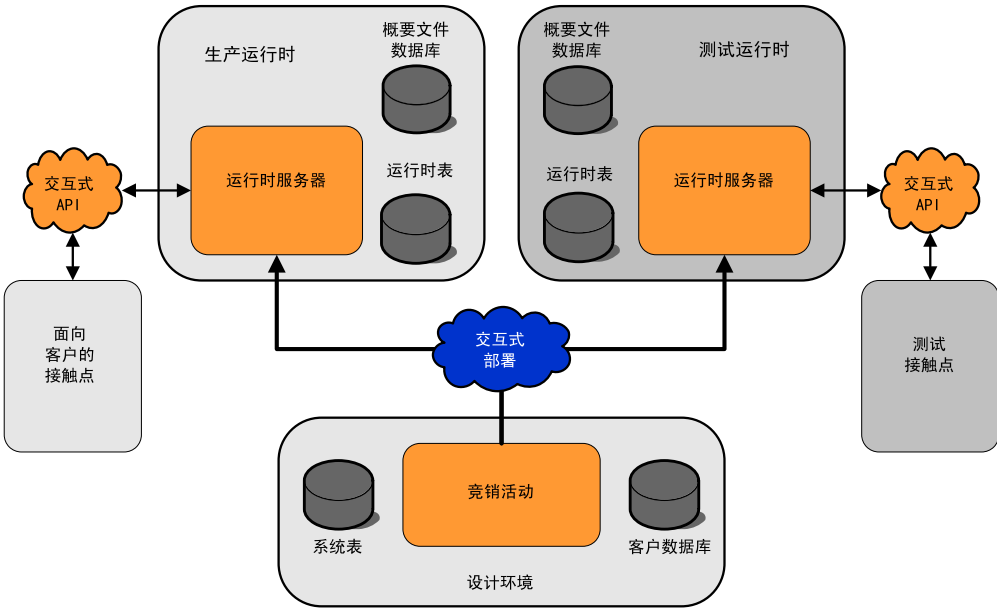
例如，您在"登录"区域中具有一组您分配了智能细分市场的商品，但在"购买后"区域中针对同一细分市场具有不同的一组商品。处理规则在营销活动的"交互策略"选项卡上定义。

每个处理规则还具有市场营销分数。如果客户分配到多个细分市场中，且因此存在多个适用的商品，那么市场营销分数可帮助定义 Interact 建议哪些商品。学习模型、商品禁止列表、全局和单个商品分配可影响运行时环境建议的哪些商品。

Interact 体系结构

Interact 环境至少包含两个主要组件：设计环境和运行时环境。您可能还具有可选的测试运行时服务器。

下图显示高级体系结构概述。



您可在设计环境中执行大部分 Interact 配置。设计环境是与 Campaign Web 应用程序一起安装的并且引用 Campaign 系统表和您的客户数据库。您使用设计环境来定义要与 API 配合使用的交互点和事件。

在您设计并配置了您希望运行时环境处理客户交互的方式之后，您要将数据部署到登台服务器组以进行测试，或部署到生产运行时服务器组以进行实时客户交互。

Interact API 提供了接触点与运行时服务器之间的连接。您引用在设计环境中通过 Interact API 创建的对象（交互点和事件）并将其用于从运行时服务器中请求信息。

Interact 网络注意事项

Interact 的产品安装至少跨两台机器。在具有多个 Interact 运行时服务器和分布式数据库的大容量生产环境中，您的安装可能跨数十台机器。

为了获得最佳性能，需要考虑几个网络拓扑需求。

- 如果您的 Interact API 实现在同一调用中启动和结束会话，例如：

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

  您不需要在负载均衡器与 Interact 运行时服务器之间启用会话持久性（粘性会话）。您可以针对本地高速缓存类型来配置 Interact 运行时服务器会话管理。
- 如果您的 Interact API 实现使用多个调用来启动和结束会话，例如：

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

  并且您对 Interact 运行时服务器使用负载均衡器，那么您应为负载均衡器启用某些类型的持久性（也称为粘性会话）。如果这不可用，或者如果您未在使用负载均衡器，请为分布式 cacheType 配置 Interact 服务器会话管理。如果您在使用分布式高速缓存，那么所有 Interact 运行时服务器必须都能够通过多点广播进行通信。您可能需要调整网络，以便使用同一多点广播 IP 地址和端口的 Interact 服务器之间的通信不会妨碍系统性能。具有粘性会话的负载均衡器比使用分布式高速缓存具有更好的性能。
- 多个服务器组之间的分布式高速缓存不受支持。
- 为获得最佳性能，请将您的运行时环境 Interact 服务器、Marketing Platform、负载均衡器和接触点保持统一地理位置中。设计时和运行时可以在不同地理位置中，但是预计部署将变慢。
- 在 Interact 生产服务器组与其关联的接触点之间具有快速网络连接（至少 1Gb）。
- 设计时需要对应运行时的 http 或 https 访问以完成部署任务。必须将任何防火墙或其他网络应用程序配置为允许部署。如果您具有大量部署，那么可能需要延长设计环境与运行时环境之间的 HTTP 超时长度。
- 联系和响应历史记录模块需要访问设计时数据库（Campaign 系统表）以及访问运行时数据库（Interact 运行时表）。您必须相应配置您的数据库和网络才能进行此数据传输。

在测试或登台安装中，您可以在同一台机器上安装 Interact 设计时和运行时。不建议对生产环境使用此方案。

Interact 服务器端口和网络安全

将 Interact 配置为保护服务器端口。

Interact 运行时端口

根据您的配置，可以关闭其中一些端口，或者并非所有 Interact 安装都需要这些端口。

用于 HTTP 的 Interact 应用程序服务器端口
处理 Interact 请求的缺省端口。

用于 HTTPS 的 Interact 应用程序服务器端口
处理 Interact 请求的缺省 SSL 端口。

Interact systemTablesDataSource 端口
请参阅 Marketing Platform 中数据源的 JDBC 配置。

Interact learningTablesDataSource 端口
请参阅 Marketing Platform 中数据源的 JDBC 配置。

Interact contactAndResponseHistoryDataSource 端口
请参阅 Marketing Platform 中数据源的 JDBC 配置。

Interact prodUserDataSource 端口
请参阅 Marketing Platform 中数据源的 JDBC 配置。

Interact testRunDataSource 端口
请参阅 Marketing Platform 中数据源的 JDBC 配置。

ETL 通信端口
在配置属性的 **Interact | ETL | patternStateETL | communicationPort** 中配置此端口。

EHCache 多点广播端口
已分发高速缓存方式时，在配置属性的 **Interact | cacheManagement | Cache | Managers | EHCache | Parameter Data | multicastPort** 中配置此端口。

ExtremeScale 目录端口
在配置属性的 **Interact | Cache Managers | Extreme Scale | Parameter Data | catalogURLs** 中配置此端口。

Interact JMX 监视端口
在配置属性下的 **Interact | monitoring | port** 中配置此端口，或运行 `-Dinteract.jmx.monitoring.port=portNumber`。

Interact Web 连接器端口

通常，此端口与 Interact 服务器端口相同，但可在 `jsconnector.xml` 中进行修改。

有关用于任何 Interact 集成产品的端口，请参阅这些产品的文档。

典型 Interact 功能不需要 JMX 监视。但是，将使用 JMX 监视进行诊断和监视。

可在 Interact 配置中禁用 JMX 端口访问，或者限制为通过防火墙配置通过特定 IP 地址访问 JMS 端口。由于最近在第三方 Apache Commons Library 中发现 JMX 漏洞，因此建议执行此操作。

对于 IBM WebSphere Application Server (WAS) Community Edition 3.0.0.3 和其他产品中使用的 Apache Geronimo 3.x 中 3.0.1 之前的 JMX 远程功能，未正确实现 RMI 类装入器，这会使远程攻击者通过使用 JMX 连接器发送手动创建的序列化对象来执行任意代码。请参阅 http://www-01.ibm.com/support/docview.wss?uid=swg21643282。

Interact 设计端口

根据您的配置，可以关闭其中一些端口，或者并非所有 Interact 安装都需要这些端口。

用于 HTTP 的 Campaign 应用程序服务器端口
处理 Interact 请求的缺省端口。

用于 HTTPS 的 Campaign 应用程序服务器端口
处理 Interact 请求的缺省 SSL 端口。

Campaign 侦听器端口
Campaign 在内部用于从 Web 客户机接受连接的端口。

其他 Campaign 设计端口
请参阅 Campaign 文档，以获取有关这些端口的更多信息。

Campaign JMX 连接器端口
在配置属性的 **Campaign | monitoring | port** 中配置此端口，以仅监视联系回应历史记录。

Campaign 运行中监视服务器端口
在配置属性的 **Campaign | monitoring | serverURL** 中配置此端口。

登录到 IBM Marketing Software

可使用此过程来登录到 IBM Marketing Software。

开始之前

需要下列各项。

- 内部网（网络）连接，用于访问 IBM Marketing Software 服务器。
- 您的计算机上安装了受支持的浏览器。
- 拥有用于登录 IBM Marketing Software 的用户名和密码。
- 拥有在您的网络上访问 IBM Marketing Software 所需的 URL。

URL 为：

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

其中，

host 是安装了 Marketing Platform 的机器。

domain.com 是主机所在的域。

port 是 Marketing Platform 应用程序服务器正在侦听的端口号。

注：以下过程假设您使用具有 Marketing Platform 的 Admin 访问权的帐户登录。

过程

使用您的浏览器访问 IBM Marketing Software URL。

- 如果 IBM Marketing Software 配置为与 Windows Active Directory 或 Web 访问控制平台集成，并且您已登录该系统，那么您将会看到缺省仪表板页面。这表示您已登录。

- 如果您看到登录屏幕，请使用缺省管理员凭证来登录。在单分区环境中，请使用 asm_admin 并使用 password 作为密码。在多分区环境中，请使用 platform_admin 并使用 password 作为密码。

  提示会要求您更改密码。您可以输入现有的密码，但为了提高安全性，您应该选择新的密码。
- 如果 IBM Marketing Software 配置为使用 SSL，那么您第一次登录时系统会提示您接受数字安全证书。单击**是**接受证书。

如果登录成功，那么 IBM Marketing Software 会显示缺省仪表板页面。

结果

借助分配给 Marketing Platform 管理员帐户的缺省许可权，可使用**设置**菜单下面列示的选项来管理用户帐户和安全。要执行 IBM Marketing Software 仪表板的最高级别管理任务，必须以 **platform_admin** 的身份登录。

第 2 章 配置用户

Interact 要求您配置两组用户，即运行时环境用户和设计环境用户。

- **运行时用户**是在配置为使用运行时服务器的 Marketing Platform 中创建。
- **设计时用户**是 Campaign 用户。配置设计团队的各个成员的安全性（与针对 Campaign 的操作相同）。

配置运行时环境用户

安装 Interact 之后，必须至少配置一个 Interact 用户：运行时环境用户。将在 Marketing Platform 中创建运行时用户。

关于此任务

运行时环境用户提供对运行时表的访问权。运行时环境用户是您用来部署交互式渠道的用户名和密码。运行时服务器对数据库凭证使用 Web 应用程序服务器 JDBC 认证。您不必对运行时环境用户添加任何运行时环境数据源。

LDAP 用户和任何 Platform 用户可以部署交互式渠道。部署交互式渠道不需要 InteractAdminRole。

创建运行时用户时：

- 如果对于每个运行时服务器有多个不同的 Marketing Platform 实例，那么必须在每个实例上创建相同的用户和密码。属于同一服务器组的所有运行时服务器必须共享用户凭证。
- 如果使用数据库装入实用程序，那么必须使用配置属性中 `Interact > general > systemTablesDataSource` 下运行时环境的登录凭证，将运行时表定义为数据源。
- 如果使用 JMXMP 协议对 JMX 监视启用安全性，那么可能需要一个单独的用户以用于 JMX 监视安全性。

要了解用于创建运行时用户的步骤，请参阅 Marketing Platform 文档。

配置设计环境用户

设计环境用户是 Campaign 用户。使用与配置 Campaign 角色许可权相同的方式来配置设计环境用户。

关于此任务

部分设计环境用户还需要某些 Campaign 许可权，如定制宏。

创建设计环境用户时：

- 如果任何 Campaign 用户具有编辑交互式流程图的许可权，请授予他们对测试运行表数据源的访问权。
- 如果已安装并配置 Interact，那么以下额外的选项可用于缺省全局策略和新策略。
-

类别	许可权
营销活动	• 查看营销活动交互策略 - 能够查看但无法编辑营销活动中的"交互策略"选项卡。 • 编辑营销活动交互策略 - 能够对"交互策略"选项卡（包括处理规则）进行更改。 • 删除营销活动交互策略 - 能够从营销活动除去"交互策略"选项卡。如果交互式渠道部署中已经包含了交互策略，那么该"交互策略"选项卡的删除受限。 • 添加营销活动交互策略 - 能够在营销活动中创建新的"交互策略"选项卡。 • 启动营销活动交互策略部署 - 能够标记"交互策略"选项卡以进行部署或取消部署。
交互式渠道	• 部署交互式渠道 - 能够将交互式渠道部署至 Interact 运行时环境。 • 编辑交互式渠道 - 能够对交互式渠道的"摘要"选项卡进行更改。 • 删除交互式渠道 - 能够除去交互式渠道。如果已部署交互式渠道，那么该交互式渠道的删除受限。 • 查看交互式渠道 - 能够查看但无法编辑交互式渠道。 • 添加交互式渠道 - 能够创建新的交互式渠道。 • 查看交互式渠道报告 - 能够查看交互式渠道的"分析"选项卡。 • 添加交互式渠道子对象 - 能够添加交互点、区域、事件和类别。
会话	• 查看交互式流程图 - 能够查看会话中的交互式流程图。 • 添加交互式流程图 - 能够在会话中创建新的交互式流程图。 • 编辑交互式流程图 - 能够对交互式流程图进行更改。 • 删除交互式流程图 - 能够除去交互式流程图。如果已部署指定了此交互式流程图的交互式渠道，那么该交互式流程图的删除受限。 • 复制交互式流程图 - 能够复制交互式流程图。 • 测试运行交互式流程图 - 能够启动交互式流程图的测试运行。 • 审阅交互式流程图 - 能够查看交互式流程图并打开流程以查看设置，但无法进行更改。 • 部署交互式流程图 - 能够标记交互式流程图以进行部署或取消部署。

设计环境许可权示例

此示例列示了已授予两个不同角色的许可权，一个角色用于创建交互式流程图的用户，另一个角色用于定义交互策略的用户。

交互式流程图角色

此表显示了已授予交互式流程图角色的许可权：

类别	许可权
定制宏	用户角色具有以下许可权： • 添加定制宏 • 编辑定制宏 • 使用定制宏
派生字段	用户角色具有以下许可权： • 添加派生字段 • 编辑派生字段 • 使用派生字段
流程图模板	用户角色具有以下许可权： • 粘贴模板
细分市场模板	用户角色具有以下许可权： • 添加细分市场 • 编辑细分市场
会话	用户角色具有以下许可权： • 查看会话摘要 • 查看交互式流程图 • 添加交互式流程图 • 编辑交互式流程图 • 复制交互式流程图 • 测试运行交互式流程图 • 部署交互式流程图

交互策略角色

此表显示了已授予交互策略角色的许可权：

类别	许可权
营销活动	用户角色具有以下许可权： • 查看营销活动摘要 • 管理营销活动目标单元 • 查看营销活动交互策略 • 编辑营销活动交互策略 • 添加营销活动交互策略 • 启动营销活动交互策略部署
商品	用户角色具有以下许可权： • 查看商品摘要
细分市场模板	用户角色具有以下许可权： • 查看细分市场摘要

类别	许可权
会话	用户角色具有以下许可权： • 审阅交互式流程图

第 3 章 管理 Interact 数据源

Interact 需要若干数据源才能正常运行。某些数据源包含 Interact 运行所需的信息，其他数据源包含您的数据。

以下各节描述了 Interact 数据源，其中包含正确配置这些数据源所需的信息以及一些维护建议。

Interact 数据源

Interact 需要有几组数据才能运行。从数据源检索并存储的数据组以及您设置的数据源取决于启用的 Interact 功能。

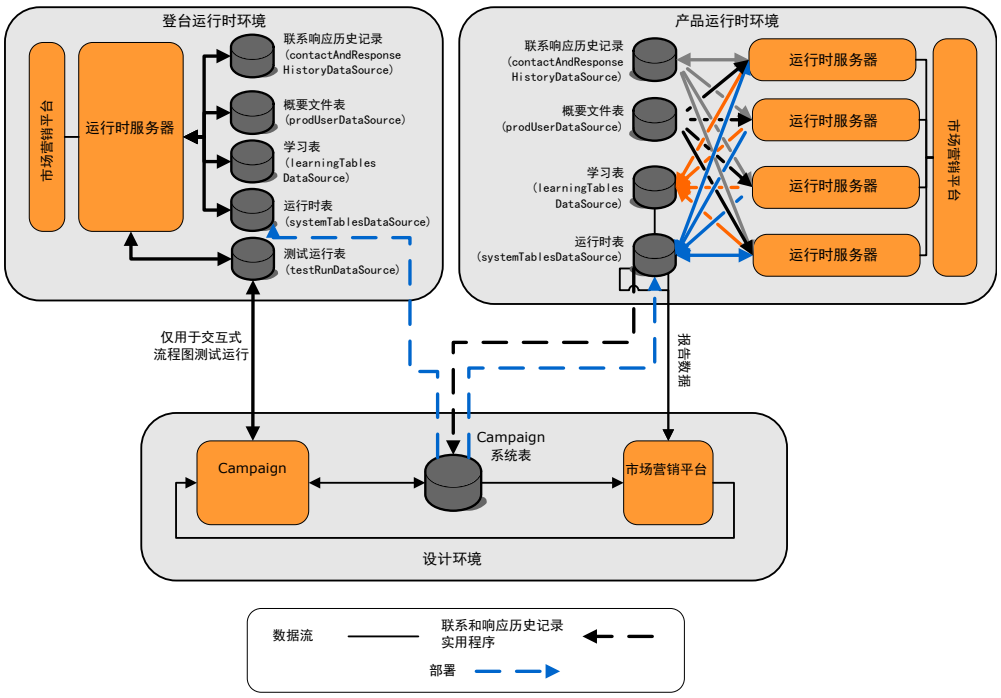
- **Campaign 系统表**。除了 Campaign 的所有数据之外，Campaign 系统表还包含您在设计环境中创建的 Interact 组件（如处理规则和交互式渠道）的数据。设计环境和 Campaign 系统表使用同一个物理数据库和模式。
- **运行时表** (systemTablesDataSource)。此数据源包含来自设计环境、联系和响应历史记录的登台表以及运行时统计信息中的部署数据。
- **概要文件表** (prodUserDataSource)。此数据源包含任何客户数据，不仅仅包括交互式流程图要将访问者正确放入智能细分市场而需要的实时收集信息。如果完全依赖实时数据，那么您不需要概要文件表。如果您在使用概要文件表，那么必须为交互式渠道使用的每个受众级别提供至少一个概要文件表。

  概要文件表还可以包含用于扩充商品供应的表，包括用于商品禁止、分数覆盖以及全局和个人商品分配的表。
- **测试运行表** (testRunDataSource)。此数据源包含交互式流程图要将访问者放入到智能细分市场而需要的所有数据的样本，包括用于模拟在交互期间实时收集的对象的数据。只有针对设计环境被指定为测试运行服务器组的服务器组才需要这些表。
- **学习表** (learningTablesDataSource)。此数据源包含内置学习实用程序所收集的所有数据。这些表可以包含用于定义动态属性的表。如果您未在使用学习，或者在使用您创建的外部学习实用程序，那么不需要学习表。
- **跨会话响应的联系和响应历史记录** (contactAndResponseHistoryDataSource)。此数据源包含 Campaign 联系历史记录表或其副本。如果您未在使用跨会话响应功能，那么不需要配置这些联系历史记录表。

数据库和应用程序

您创建并由 Interact 使用的数据源可能也可用于与其他 IBM Marketing Software 应用程序交换或共享数据。

下图显示了 Interact 数据源以及它们与 IBM Marketing Software 应用程序之间的关系。



- Campaign 以及测试运行服务器组均访问测试运行表。
- 测试运行表仅用于交互式流程图测试运行。
- 在使用运行时服务器来测试部署时（包括 Interact API），运行时服务器将对数据使用概要文件表。
- 如果您配置联系和响应历史记录模块，那么此模块将使用后台的抽取、变换、装入 (ETL) 过程将数据从运行时登台表移动到 Campaign 联系和响应历史记录表。
- 报告功能从学习表、运行时表和 Campaign 系统表中查询数据以在 Campaign 中显示报告。

您应将测试运行时环境配置为与生产运行时环境使用一组不同的表。通过登台与生产之间的单独表，您可以将测试结果与实际结果隔离。请注意，联系和响应历史记录模块始终将数据插入到实际 Campaign 联系和响应历史记录表（Campaign 不具备测试联系和响应历史记录表）。如果您具有单独学习表以用于测试运行时环境，并且您希望在报告中看到结果，那么您需要一个对测试环境运行学习报告的单独 IBM Cognos® BI 实例。

Campaign 系统表

在安装 Interact 设计环境时，您还将在 Campaign 系统表中创建新的特定于 Interact 的表。您创建的表取决于启用的 Interact 功能。

如果启用联系和响应历史记录模块，此模块会将联系和响应历史记录从运行时表中的登台表复制到 Campaign 系统表中的联系和响应历史记录表。缺省表是 UA_ContactHistory、UA_DtlContactHist 和 UA_ResponseHistory，但联系和响应历史记录模块将使用在 Campaign 中为联系和响应历史记录表映射的任何表。

如果使用全局商品表和分数覆盖表来分配商品，并且如果使用交互式渠道的处理规则中未包含的商品，那么可能需要填充 Campaign 系统表中的 UACI_ICBatchOffers 表。

运行时表

如果有一个以上的受众级别，那么必须针对每个受众级别来为联系和响应历史记录数据创建登台表。

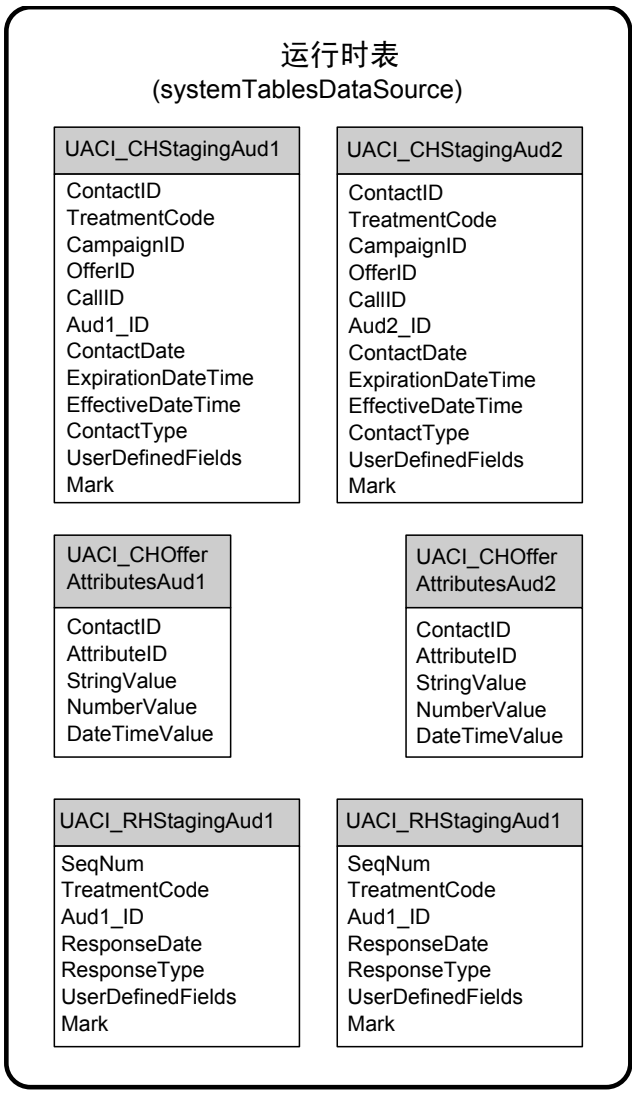
SQL 脚本将为缺省受众级别创建以下表：

- UACI_CHStaging
- UACI_CHOfferAttrib
- UACI_RHStaging

必须针对运行时表中每个受众级别来创建以上三个表的副本。

如果您的 Campaign 联系和响应历史记录表具有用户定义的字段，那么必须在 UACI_CHStaging 和 UACI_RHStaging 表中创建相同字段名称和类型。您可通过在会话数据中创建相同名称的名称/值对而在运行时期间填充这些字段。例如，您的联系和响应历史记录表包含字段 catalogID。必须将 catalogID 字段同时添加到 UACI_CHStaging 和 UACI_RHStaging 表。随后，Interact API 通过将事件参数定义为名为 catalogID 的名称/值对来填充此字段。可以由概要文件表、临时数据、学习或 Interact API 来提供会话数据。

下图显示了受众 Aud1 和 Aud2 的样本表。此图不包含运行时数据库中的所有表。



表中的所有字段都是必填字段。可以修改 `CustomerID` 和 `UserDefinedFields` 以匹配您的 Campaign 联系和响应历史记录表。

测试运行表

测试运行表仅用于交互式流程图的测试运行。交互式流程图的测试运行应测试您的细分市场逻辑。对于您的 Interact 安装，您只需要为其配置一个测试运行数据库。测试运行表不需要位于独立数据库中。例如，您可以将您的客户数据表用于 Campaign。

与测试运行表相关联的数据库用户必须具有 CREATE 特权才能添加测试运行结果表。

测试运行数据库必须包含在交互式渠道中映射的所有表。

这些表应包含某些数据，这些数据用于运行您要在交互式流程图中测试的方案。例如，如果您的交互式流程图的某种逻辑用于根据在语音邮件系统中选择的选项来将人员在细分市场中排序，那么您应为每个可能的选择至少提供一行。如果您在创建一个交互来处理您的 Web 站点上的表单，那么您应包含可表示数据缺失或格式错误的行，例如，将 `name@domaincom` 用于电子邮件地址的值。

每个测试运行表必须至少包含相应受众级别的标识的列表，以及一个表示您期望使用的实时数据的列。由于测试运行无法访问实时数据，因此您必须为每一段期望的实时数据提供样本数据。例如，如果您希望使用您可以实时收集的数据（如最近一次访问的 Web 页面的名称（存储在属性 lastPageVisited 中），或者购物车中商品的数量（存储在属性 shoppingCartItemCount 中）），那么您必须使用相同名称创建列，然后使用样本数据来填充列。这使您可以测试运行您的流程图逻辑中某些具有行为性质或上下文性质的分支。

未针对处理大量数据组来优化交互式流程图的测试运行。您可以限制用于交互进程中测试运行的行数。但是，这始终导致会选择行的第一个集合。为确保会选择行的不同集合，请使用测试运行表的不同视图。

要测试运行时中交互式流程图的吞吐量性能，您必须创建一个测试运行时环境，包括用于测试环境的概要文件表。

实际上，您可能需要三组表来进行测试：测试运行表（用于交互式流程图的测试运行）、测试概要文件表（用于测试服务器组）和一组生产概要文件表。

覆盖用于动态创建表的缺省数据类型

Interact 运行时环境在以下两种情况下动态创建表：流程图的测试运行期间；运行快照进程期间（此进程向尚不存在的表中进行写入）。要创建这些表，Interact 依赖于每种受支持数据库类型的硬编码数据类型。

您可以覆盖缺省数据类型，方法是在 testRunDataSource 或 prodUserDataSource 中创建名为 uaci_column_types 的备用数据类型的表。这一附加表使 Interact 能够适应硬编码数据类型未涵盖的极少数情况。

在定义 uaci_column_types 表时，Interact 将列的元数据用作要用于任何表生成的数据类型。如果未定义 uaci_column_types 表，或者如果在尝试读取此表时遇到了任何异常，那么将使用缺省数据类型。

在启动时，运行时系统首先检查 testRunDataSource 来获取 uaci_column_types 表。如果 testRunDataSource 中不存在 uaci_column_types 表，或者如果 prodUserDataSource 是不同数据库类型，那么 Interact 将检查 prodUserDataSource 来获取此表。

覆盖缺省数据类型

使用此过程来覆盖动态创建的表的缺省数据类型。

关于此任务

无论何时更改 uaci_column_types 表，都必须重新启动运行时服务器。作出更改计划，以使重新启动服务器对操作的影响最小。

过程

1. 在 TestRunDataSource 或 ProdUserDataSource 中创建一个具有以下属性的表：

   表名：uaci_column_types

   列名：

- uaci_float
- uaci_number
- uaci_datetime
- uaci_string

使用您的数据库所支持的相应数据类型来定义每一列。

2. 重新启动运行时服务器以使 Interact 识别新表。

动态创建的表的缺省数据类型

对于 Interact 运行时系统使用的每种受支持数据库，存在缺省情况下对浮点、数字、日期/时间和字符串列使用的硬编码数据类型。

表 *1.* 动态创建的表的缺省数据类型

数据库	缺省数据类型
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

概要文件数据库

概要文件数据库的内容完全取决于您配置交互式流程图和 Interact API 所需要的数据。Interact 要求或建议每个数据库都包含特定表或数据。

概要文件数据库必须包含以下内容：

- 交互式渠道中映射的所有表。

  这些表必须包含要在生产中运行交互式流程图而必需的所有数据。这些表应正确建立序列化的简化索引。由于访问维数据需要耗费性能成本，因此应尽可能地使用非规范化模式。至少，您应对受众级别标识字段上的概要文件表建立索引。如果有从维度表中检索的其他字段，那么应该对这些字段相应地建立索引以减少数据库访存时间。概要文件表的受众标识必须与 Campaign 中定义的受众标识匹配。

- 如果将 enableScoreOverrideLookup 配置属性设置为 true，那么您必须至少为一个受众级别包含分数覆盖表。您将通过 scoreOverrideTable 属性来定义分数覆盖表名称。

  分数覆盖表包含个别的客户与商品的配对。可以通过针对概要文件数据库运行 aci_usrtab SQL 脚本来创建样本分数覆盖表 UACI_ScoreOverride。您还应在此表的"受众标识"列上建立索引。

  如果您将 enableScoreOverrideLookup 属性设置为 false，那么不需要包含分数覆盖表。
- 如果将 enableDefaultOfferLookup 配置属性设置为 true，那么必须包含全局商品表 (UACI_DefaultOffers)。可以通过针对概要文件数据库运行 aci_usrtab SQL 脚本来创建全局商品表。

  全局商品表可以包含受众与商品的配对。
- 如果将 enableOfferSuppressionLookup 属性设置为 true，那么必须至少为一个受众级别包含商品禁止表。您将通过 offerSuppressionTable 属性来定义商品禁止表名称。

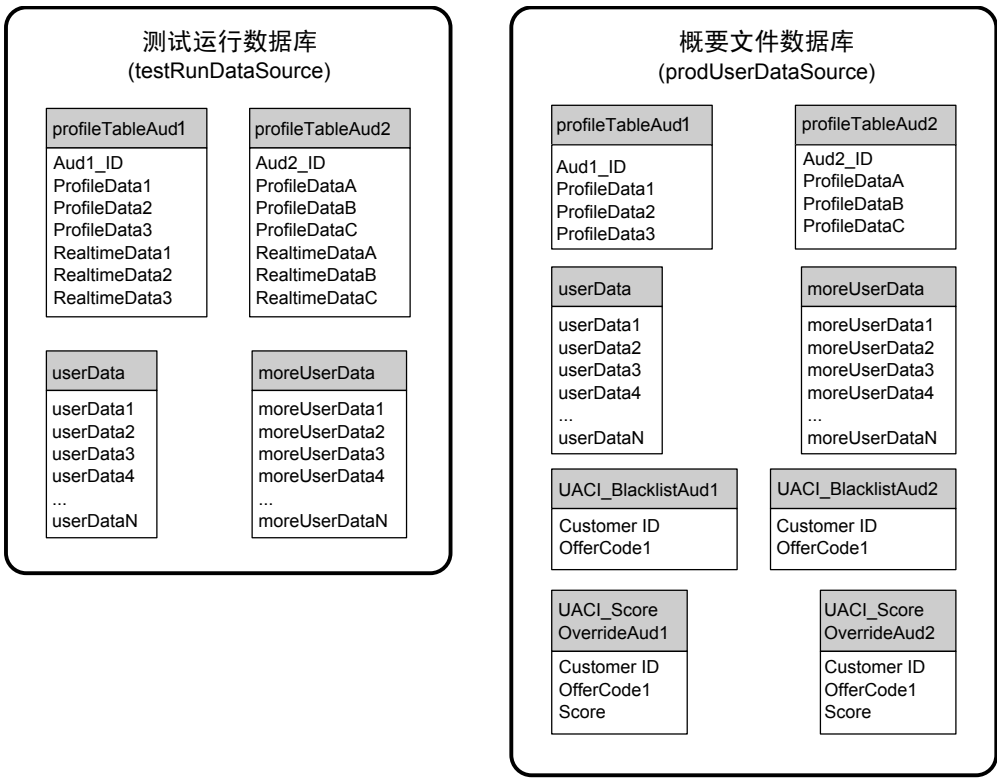
  商品禁止表可以针对为某一受众成员禁止的每个商品包含一行，尽管并非所有受众成员都需要一个条目。可以通过针对概要文件数据库运行 aci_usrtab SQL 脚本来创建样本商品禁止表 UACI_BlackList。

  如果您将 enableOfferSuppressionLookup 属性设置为 false，那么不需要包含商品禁止表。

其中任何一个表中具有大量数据都可能会降低性能。为了获得最佳结果，请在受众级别列上为用于运行时的具有大量数据的表建立相应索引。

以上引用的所有配置属性都位于 **Interact > 概要文件**或 **Interact> 概要文件 > 受众级别 > *AudienceLevel*** 类别中。aci_usrtab SQL 脚本位于您的运行时环境安装目录的 ddl 目录中。

下图显示了受众级别 Aud1 和 Aud2 的测试运行和概要文件数据库的示例表。



学习表

如果您在使用 Interact 内置学习，那么必须配置学习表。这些表包含内置学习功能要了解的所有数据。

如果您在使用动态学习属性，那么必须填充 UACI_AttributeList 表。

学习涉及到写入到中间登台表以及将来自登台表的信息聚集到学习表。Interact > offerserving > Built-in Learning Config 类别中的 insertRawStatsIntervalInMinutes 和 aggregateStatsIntervalInMinutes 配置属性确定了填充学习表的频率。

insertRawStatsIntervalInMinutes 属性确定将每个客户和商品的接受和联系信息从内存移动到登台表 UACI_OfferStatsTX 和 UACI_OfferTxAll 的频率。登台表中存储的信息将进行聚集并按照 aggregateStatsIntervalInMinutes 配置属性确定的固定时间间隔移动到 UACI_OfferStats 和 UACI_OfferStatsAll 表。

Interact 内置学习使用此数据来计算商品的最终分数。

跨会话响应跟踪的联系历史记录

如果您启用跨会话响应功能，那么运行时环境需要对 Campaign 联系历史记录表的只读访问权。您可以将运行时环境配置为查看 Campaign 系统表，也可以创建 Campaign 联系历史记录表的副本。如果您创建表的副本，那么必须管理使副本保持为最新的过程。联系和响应历史记录模块将不会更新联系历史记录表的副本。

您必须针对这些联系历史记录表运行 aci_crhtab SQL 脚本以添加跨会话响应跟踪功能所必需的表。

运行数据库脚本以启用 功能

要使用 中提供的可选功能，请针对数据库运行数据库脚本，以创建表或更新现有表。

您的 安装（设计时环境和运行时环境）包括功能 **ddl** 脚本。**ddl** 脚本将必需的列添加到您的表中。

要启用任何可选功能，请针对指示的数据库或表运行相应的脚本。

dbType 是数据库类型，例如，sqlsvr（对于 Microsoft SQL Server）、ora（对于 Oracle）或 db2（对于 IBM DB2）。

使用下表来针对数据库运行数据库脚本，以创建表或更新现有表：

表 2. 数据库脚本

功能名称	功能脚本	运行对象	更改
全局商品、商品禁止和分数覆盖	*Interact_Home*\ddl\acifeatures\（运行时环境安装目录）中的 **aci_usrtab_***dbType***.sql**	您的概要文件数据库 (userProdDataSource)	创建 UACI_DefaultOffers、UACI_BlackList 和 UACI_ScoreOverride 表。
计分	*Interact_Home*\ddl\acifeatures\（运行时环境安装目录）中的 **aci_scoringfeature_***dbType***.sql**	您的概要文件数据库中的分数覆盖表 (userProdDataSource)	添加 LikelihoodScore 和 AdjExploreScore 列。
学习	*Interact_Home*\interactDT\ddl\acifeatures\（设计时环境安装目录）中的 **aci_lrnfeature_***dbType***.sql**	包含联系历史记录表的 Campaign 数据库	请将列 RTSelectionMethod、RTLearningMode 和 RTLearningModelID 添加到 UA_DtlContactHist 表。也请将列 RTLearningMode 和 RTLearningModelID 添加到 UA_ResponseHistory 表。可选的 报告包所提供的报告功能也需要此脚本。

关于联系和响应历史记录跟踪

您可以将运行时环境配置为在 Campaign 联系和响应历史记录表中记录联系和响应历史记录。运行时服务器将联系和响应历史记录存储在登台表中。联系和响应历史记录模块将此数据从登台表复制到 Campaign 联系和响应历史记录表。

仅当您针对设计环境在"配置"页面上将 Campaign > partitions > partition1 > Interact > interactInstalled 和 contactAndResponseHistTracking > isEnabled 属性设置为 yes 时，联系和响应历史记录模块才会起作用。

如果您在使用跨会话响应跟踪模块，那么联系和响应历史记录模块是一个单独实体。

联系和响应类型

可以使用 Interact 来记录一种联系类型和两种响应类型。还可以使用 postEvent 方法来记录更多定制响应类型。

contactAndResponseHistTracking 表属性

此表列示了在 contactAndResponseHistTracking 类别中找到的属性：

事件	联系/响应类型	配置属性
记录商品联系人	联系	contacted
记录商品接受	响应	接受
记录商品拒绝	响应	拒绝

UA_UsrResponseType 表属性

确保 Campaign 系统表中 UA_UsrResponseType 表的 CountsAsResponse 列已正确配置。UA_UsrResponseType 表中必须存在所有这些响应类型。

要使条目在 UA_UsrResponseType 表中有效，必须为该表中的所有列（其中包括 CountsAsResponse）定义值。CountsAsResponse 的有效值为：

- 0 - 未响应
- 1 - 一次响应
- 2 - 一次拒绝
-

这些响应用于进行报告。

其他响应类型

在 Interact 中，您可以在 Interact API 中使用 postEvent 方法来触发用于记录对于商品的"接受"或"拒绝"操作的事件。您还可以扩充系统以使 postEvent 调用能够记录其他响应类型，如探查、考虑、提交或履行。

所有这些响应类型都必须存在于 Campaign 系统表的 UA_UsrResponseType 表中。通过对 postEvent 方法使用特定事件参数，您可以记录其他响应类型并定义是否应在学习中包含接受。

要记录其他响应类型，必须添加以下事件参数：

- **UACIResponseTypeCode** - 表示响应类型代码的字符串。该值必须是 UA_UsrResponseType 表中的有效条目。

  要使条目在 UA_UsrResponseType 中有效，您必须定义表中的所有列，包括 CountsAsResponse。 CountsAsResponse 的有效值为 0、1 或 2。0 表示无响应，1 表示有某个响应，2 表示拒绝。这些响应用于进行报告。
- **UACILogToLearning** - 值为 1 或 0 的数字。1 指示 Interact 应将事件记录为接受学习系统或在会话中启用商品抑制。0 指示 Interact 不应将事件记录到学习系统，也不应在会话中启用商品抑制。此参数使您可以创建若干 postEvent 方法，这些方法记录不同响应类型，而不影响学习。如果不定义 UACILogToLearning，那么 Interact 假定缺省值为 0。

如果在发布接受事件时提供了 responseTypeCode，那么在接受时不会禁止该商品。无论 ResponseTypeCode 值是多少（例如，0、1 或 2），如果 logToLearningAsAccept 为 0，那么应该永远不会禁止该商品。要禁止该商品，postEvent 不应该指定

UACIResponseTypeCode 参数。如果提供了 UACIResponseTypeCode 参数，那么当您希望禁止该商品时，UACILogToLearning 的值应该为 1。

您可能希望创建若干包含"记录商品接受"操作的事件，一个事件对应于您要记录的每个响应类型；或者要创建单个事件，其中包含您用于每个 postEvent 调用（用于记录单独响应类型）的"记录商品接受"操作。

例如，为每种响应类型创建一个具有"记录商品验收"操作的事件。您在 UA_UsrResponseType 表 [Name (code)] 中定义以下定制响应：Explore (EXP)、Consider (CON) 和 (CMT)。然后，您将创建三个事件并将其命名为 LogAccept_Explore、LogAccept_Consider 和 LogAccept_Commit。所有这个三个事件都完全相同（具有"记录商品接受"操作），但是名称不同，以便使用 API 的人员能够区分它们。

或者，您也可以创建单个事件，其中包含您将用于所有定制响应类型的"记录商品接受"操作。例如，将其命名为 LogCustomResponse。

使用 API 时，事件之间没有功能上的差异，但是命名约定可能会使代码更加明确。此外，如果您为每个定制响应提供一个单独名称，那么"渠道事件活动摘要"报告将显示更加精确的信息。

首先，设置所有名称/值对

```
//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);
```

以下第一个示例表明使用个别事件。

```
//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);
```

以下第二个示例表明仅使用一个事件。

```
//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);
```

这两个示例执行完全相同的操作，但是，其中一个版本可能比另一个更容易阅读。

运行时环境登台表与 Campaign 历史记录表之间的映射

Interact 联系历史记录登台表映射到 Campaign 历史记录表。对于每个受众级别，您都必须具有其中一个运行时环境登台表。

UACI_CHStaging 联系历史记录登台表映射

此表说明 UACI_CHStaging 运行时环境登台表映射到 Campaign 联系历史记录表的方式。显示的表名称是为运行时表和 Campaign 系统表中的缺省受众创建的样本表。

表 3. 联系历史记录

UACI_CHStaging **Interact 联系历史记录登台表列名称**	**Campaign 联系历史记录表**	**表列名称**
ContactID	不适用	不适用
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 是用于将 UACI_CHOfferAtrib 表与 UACI_CHStaging 表进行连接的键。userDefinedFields 列可以包含您选择的任何数据。

UACI_CHOfferAttrib 联系历史记录登台表映射

此表说明 UACI_CHOfferAttrib 运行时环境登台表映射到 Campaign 联系历史记录表的方式。显示的表名称是为运行时表和 Campaign 系统表中的缺省受众创建的样本表。

表 4. 商品属性

UACI_CHOfferAttrib **Interact 联系历史记录登台表列名称**	**Campaign 联系历史记录表**	**表列名称**
ContactID	不适用	不适用
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

UACI_RHStaging 联系响应历史记录登台表映射

此表说明 UACI_RHStaging 运行时环境登台表映射到 Campaign 响应历史记录表的方式。显示的表名称是为运行时表和 Campaign 系统表中的缺省受众创建的样本表。

表 5. 响应历史记录

UACI_RHStaging **Interact 响应历史记录登台表列名称**	**Campaign 响应历史记录表**	**表列名称**
SeqNum	不适用	不适用
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime
ResponseType	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

SeqNum 是由联系和响应历史记录模块用于标识数据的键，但是未记录在 Campaign 响应表中。userDefinedFields 列可以包含您选择的任何数据。

登台表中的其他列

如果您将向登台表中添加列，那么联系和响应历史记录模块会将这些列写入到 UA_DtlContactHist 或 UA_ResponseHistory 表的具有相同名称的列中。

例如，如果您将列 linkFrom 添加到 UACI_CHStaging 表，那么联系和响应历史记录模块会将该数据复制到 UA_DtlContactHist 表中的 linkFrom 列。

Campaign 联系和响应历史记录表中的其他列

如果 Campaign 联系和响应历史记录表中有其他列，那么在运行联系和响应历史记录模块之前，请将匹配列添加到登台表。

您可通过创建与运行时会话数据中的名称/值对同名的列来填充登台表中额外的列。

例如，创建名称/值对 NumberItemsInWishList 和 NumberItemsInShoppingCart 并将它们添加到 UACI_RHStaging 表。当发生"记录商品接受"或"记录商品拒绝"事件时，运行时环境会填充这些字段。当发生"商品联系记录"事件时，运行时环境将填充 UACI_CHStaging 表。

使用表来包括商品的分数

您可以使用用户定义的字段来包括用于呈示商品的分数。向运行时表中的 UACI_CHStaging 表以及 Campaign 系统表中的 UA_DtlContactHist 表中添加一个名为 FinalScore 的列。如果您在使用内置学习，那么 Interact 将通过用于商品的最终分数来自动填充 FinalScore 列。

如果您在构建定制学习模块，那么您可以使用 ITreatment 接口的 setActualValueUsed 方法以及 ILearning 接口的 logEvent 方法。

如果您未在使用学习，请向运行时表中的 UACI_CHStaging 表以及 Campaign 系统表中的 UA_DtlContactHist 表中添加一个名为 Score 的列。Interact 将通过用于商品的分数来自动填充 Score 列。

在 Campaign 中创建新的历史记录表并在 Interact 中创建新的登台表

如果您要使用除 Customer 之外的受众级别，那么必须在 Campaign 中创建新的历史记录表，并在 Interact 中创建新的登台表。

例如，将在 IBM DB2 设计时数据库中使用以下样本脚本，以在 Campaign 中为 Account 类型的受众级别创建历史记录表。

```
DROP TABLE ACCT_UA_ResponseHistory;
DROP TABLE ACCT_UA_DtlContactHist;
DROP TABLE ACCT_UA_ContactHistory;
CREATE TABLE ACCT_UA_ResponseHistory (
        AccountID           varchar(30) NOT NULL,
        TreatmentInstID      bigint NOT NULL,
        ResponsePackID       bigint NOT NULL,
        ResponseDateTime     timestamp NOT NULL,
        WithinDateRangeFlg   int,
        OrigContactedFlg     int,
        BestAttrib           int,
        FractionalAttrib     float,
        DirectResponse       int,
        CustomAttrib         float,
        ResponseTypeID       bigint,
        DateID               bigint,
        TimeID               bigint,
        UserDefinedFields    char(18),
        CONSTRAINT ACCT_cRespHistory_PK
               PRIMARY KEY (AccountID, TreatmentInstID,
               ResponsePackID )
 );
 CREATE TABLE ACCT_UA_ContactHistory (
        AccountID           varchar(30) NOT NULL,
        CellID               bigint NOT NULL,
        PackageID            bigint NOT NULL,
        ContactDateTime      timestamp,
        UpdateDateTime       timestamp,
        ContactStatusID      bigint,
        DateID               bigint,
        TimeID               bigint,
        UserDefinedFields    char(18),
        CONSTRAINT ACCT_cContactHist_PK
```

```
            PRIMARY KEY (AccountID, CellID, PackageID )
);
CREATE INDEX ACCT_cContactHist_IX1 ON ACCT_UA_ContactHistory
(
      CellID );
CREATE INDEX ACCT_cContactHist_IX2 ON ACCT_UA_ContactHistory
(
      PackageID                    ,
      CellID );
CREATE TABLE ACCT_UA_DtlContactHist (
      AccountID           varchar(30) NOT NULL,
      TreatmentInstID      bigint NOT NULL,
      ContactStatusID      bigint,
      ContactDateTime      timestamp,
      UpdateDateTime       timestamp,
      UserDefinedFields    char(18),
      DateID               bigint NOT NULL,
      TimeID               bigint NOT NULL
);
CREATE INDEX ACCT_cDtlContHist_IX1 ON ACCT_UA_DtlContactHist
(
      AccountID                    ,
      TreatmentInstID );
ALTER TABLE ACCT_UA_ResponseHistory
      ADD CONSTRAINT ACCT_cRespHistory_FK2
            FOREIGN KEY (TimeID)
                        REFERENCES UA_Time  (TimeID);
ALTER TABLE ACCT_UA_ResponseHistory
      ADD CONSTRAINT ACCT_cRespHistory_FK4
            FOREIGN KEY (DateID)
                        REFERENCES UA_Calendar  (DateID);
ALTER TABLE ACCT_UA_ResponseHistory
      ADD CONSTRAINT ACCT_cRespHistory_FK3
            FOREIGN KEY (ResponseTypeID)
                        REFERENCES UA_UsrResponseType  (
            ResponseTypeID);
ALTER TABLE ACCT_UA_ResponseHistory
      ADD CONSTRAINT ACCT_cRespHistory_FK1
            FOREIGN KEY (TreatmentInstID)
                        REFERENCES UA_Treatment  (
            TreatmentInstID);
ALTER TABLE ACCT_UA_ContactHistory
      ADD CONSTRAINT ACCT_cContactHist_FK2
            FOREIGN KEY (DateID)
                        REFERENCES UA_Calendar  (DateID);
ALTER TABLE ACCT_UA_ContactHistory
      ADD CONSTRAINT ACCT_cContactHist_FK3
            FOREIGN KEY (TimeID)
                        REFERENCES UA_Time  (TimeID);
ALTER TABLE ACCT_UA_ContactHistory
      ADD CONSTRAINT ACCT_cContactHist_FK1
            FOREIGN KEY (ContactStatusID)
                        REFERENCES UA_ContactStatus  (
            ContactStatusID);
ALTER TABLE ACCT_UA_DtlContactHist
      ADD CONSTRAINT ACCT_cDtlContactH_FK3
            FOREIGN KEY (TimeID)
                        REFERENCES UA_Time  (TimeID);
ALTER TABLE ACCT_UA_DtlContactHist
      ADD CONSTRAINT ACCT_cDtlContactH_FK2
            FOREIGN KEY (DateID)
                        REFERENCES UA_Calendar  (DateID);
ALTER TABLE ACCT_UA_DtlContactHist
      ADD CONSTRAINT ACCT_cDtlContactH_FK1
            FOREIGN KEY (ContactStatusID)
                        REFERENCES UA_ContactStatus  (
```

```
              ContactStatusID);
alter table ACCT_UA_DtlContactHist add RTSelectionMethod int;
alter table ACCT_UA_ResponseHistory add RTSelectionMethod int;
```

将在运行时 IBM DB2 数据库中使用以下样本脚本，以在 Interact 中为 Account 类型的受众级别创建历史记录登台表。

```
DROP TABLE ACCT_UACI_RHStaging;
DROP TABLE ACCT_UACI_CHOfferAttrib;
DROP TABLE ACCT_UACI_CHStaging;
DROP TABLE ACCT_UACI_UserEventActivities;
DROP TABLE ACCT_UACI_EventPatternState;
CREATE TABLE ACCT_UACI_RHStaging (
       SeqNum               bigint NOT NULL,
       TreatmentCode        varchar(512),
       AccountID           varchar(30),
       ResponseDate         timestamp,
       ResponseType         int,
       ResponseTypeCode     varchar(64),
       Mark                 bigint NOT NULL
                                       DEFAULT 0,
       UserDefinedFields    char(18),
  RTSelectionMethod    int,
       CONSTRAINT iRHStaging_PK1
              PRIMARY KEY (SeqNum)
);
CREATE TABLE ACCT_UACI_CHOfferAttrib (
        ContactID            bigint NOT NULL,
        AttributeID          bigint NOT NULL,
        StringValue          varchar(512),
        NumberValue          float,
        DateTimeValue        timestamp,
        CONSTRAINT ACCT_iCHOfferAttrib_PK
               PRIMARY KEY (ContactID, AttributeID)
 );
 CREATE TABLE ACCT_UACI_CHStaging (
        ContactID            bigint NOT NULL,
        TreatmentCode        varchar(512),
        CampaignID           bigint,
        OfferID              bigint,
        CellID               bigint,
        AccountID           varchar(30),
        ContactDate          timestamp,
        ExpirationDateTime   timestamp,
        EffectiveDateTime    timestamp,
        ContactType          int,
        UserDefinedFields      char(18),
        Mark                 bigint NOT NULL DEFAULT 0,
  RTSelectionMethod    bigint,
        CONSTRAINT ACCT_iCHStaging_PK
               PRIMARY KEY (ContactID)
 );
CREATE TABLE ACCT_UACI_UserEventActivity
(
       SeqNum               bigint NOT NULL GENERATED ALWAYS AS IDENTITY,
       ICID                 bigint NOT NULL,
       ICName               varchar(64) NOT NULL,
       CategoryID           bigint NOT NULL,
       CategoryName         varchar(64) NOT NULL,
       EventID              bigint NOT NULL,
       EventName            varchar(64) NOT NULL,
       TimeID               bigint,
       DateID               bigint,
       Occurrences          bigint NOT NULL,
       AccountID varchar(30) not null,
       CONSTRAINT iUserEventActivity_PK
              PRIMARY KEY (SeqNum)
```

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

为联系和响应历史记录模块配置 JMX 监视

使用此过程来为联系和响应历史记录模块配置 JMX 监视。将支持 JMXMP 和 RMI 协议。配置 JMX 监视并不会为联系和响应历史记录模块启用安全性。请使用设计环境的 Marketing Platform 来配置 JMX 监视。

关于此任务

为了将 JMX 监视工具用于联系和响应历史记录模块，用于以下协议的缺省地址分别为：

- JMXMP 协议的缺省地址为 `service:jmx:jmxmp://CampaignServer:port/campaign`。
- RMI 协议的缺省地址为 `service:jmx:rmi:///jndi/rmi://CampaignServer:port/campaign`。

当您在 JMX 监视工具中查看数据时，结果属性首先按分区进行组织，然后按受众级别进行组织。

过程

在设计环境的 Marketing Platform 中，编辑 `Campaign > monitoring` 类别中的以下配置属性。

配置属性	设置
monitorEnabledForInteract	**True**
port	JMX 服务的端口号
协议	要使用的协议： • **JMXMP** • **RMI** 不为联系和响应历史记录模块启用安全性，即使您选择 JMXMP 协议。

关于跨会话响应跟踪

访问者可能不会总是在对您接触点的单次访问中完成交易。客户可能将某一商品添加到其购物车（您的 Web 站点上）并在两天之后完成销售。无限期保持运行时会话处于活动状态是不可行的。您可以启用跨会话响应跟踪以跟踪某一会话中的商品呈现，并将其与另一会话中的响应进行匹配。

缺省情况下，Interact 跨会话响应跟踪可以匹配处理代码或商品代码。您还可以将其配置为匹配您选择的任何定制代码。跨会话响应匹配可用数据。例如，您的 Web 站点包括一个商品，此商品具有一个在显示时生成的促销代码，可在一周内享受巨大折扣。用户可能会将商品添加到其购物车，但是在三天之后才完成购买。如果您使用 postEvent 调用来记录接受事件，那么您只能包含促销代码。因为运行时在当前会话中找不到要匹配的处理代码或商品代码，运行时会将包含可用信息的接受事件放在跨会话响应 (XSessResponse) 登台表中。CrossSessionResponse 服务定期读取 XSessResponse 表，并尝试将记录与可用的联系历史记录数据进行匹配。CrossSessionResponse 服务将促销代码与联系历史记录进行匹配，并收集所有必需数据以记录正确响应。然后，CrossSessionResponse 服务将响应写入到响应登台表，如果启用了学习，还将写入到学习表。然后，联系和响应历史记录模块将响应写入到 Campaign 联系和响应历史记录表。跨会话响应的处理是否成功，取决于已由联系历史记录 ETL 迁移到 Campaign 数据库的原始联系历史记录。

跨会话响应跟踪数据源配置

Interact 跨会话响应跟踪将运行时环境中的会话数据与 Campaign 联系和响应历史记录进行匹配。缺省情况下，跨会话响应跟踪与处理代码或商品代码相匹配。您可以配置运行时环境以匹配定制、备用代码。

- 如果您选择匹配备用代码，那么必须在 Interact 运行时表中的 UACI_TrackingType 表中定义备用代码。
- 运行时环境必须能够访问 Campaign 联系历史记录表。可以通过两种方法来实现此目的：将运行时环境配置为能够访问 Campaign 联系历史记录表；在运行时环境中创建联系历史记录表的副本。

  此访问是只读，并且不与联系和响应历史记录实用程序共享配置。

  如果您创建表的副本，那么您有责任确保联系历史记录副本中的数据是准确的。您可以使用 purgeOrphanResponseThresholdInMinutes 属性来配置 CrossSessionResponse 服务保留不匹配响应（以便匹配您刷新联系历史记录表副本中数据的频率）的时间长度。如果您在使用联系和响应历史记录模块，那么您应协调 ETL 更新以确保具有最新数据。

为跨会话响应跟踪配置联系和响应历史记录表

无论您是创建联系历史记录表的副本还是在 Campaign 系统表中使用实际表，您都必须执行以下步骤以配置联系和响应历史记录表。

开始之前

在执行这些步骤之前，必须在 Campaign 中正确地映射联系和响应历史记录表。

过程

1. 对 Campaign 系统表中的 UA_DtlContactHist 和 UA_ResponseHistory 表运行 Interact 设计环境安装目录中 interactDT/ddl/acifeatures 中的 aci_lrnfeature SQL 脚本。

   这会将 RTSelectionMethod 列添加到 UA_DtlContactHist 和 UA_ResponseHistory 表。针对每个受众级别的这些表运行 aci_lrnfeature 脚本。根据需要编辑脚本以对您的每个受众级别使用正确的表。

2. 如果您要将联系历史记录表复制到运行时环境，请立即执行此操作。

   如果您在创建 Campaign 联系历史记录表（可供运行时环境访问）的副本以支持跨会话响应跟踪，请遵循以下指南：

   - 跨会话响应跟踪需要对这些表进行只读访问。
   - 跨会话响应跟踪需要 Campaign 联系历史记录中的以下表。
     - UA_DtlContactHist（针对每个受众级别）
     - UA_Treatment

     您必须定期更新这些表中的数据以确保精确的响应跟踪。

3. 针对联系和响应历史记录数据源运行 aci_crhtab SQL 脚本（位于 Interact 运行时环境安装目录的 ddl 目录中）。

   此脚本将创建 UACI_XsessResponse 和 UACI_CRHTAB_Ver 表。

4. 为每个受众级别创建一个版本的 UACI_XsessResponse 表。

结果

为提高跨会话响应跟踪的性能，您可能希望限制联系历史记录数据的量：通过您复制联系历史记录数据的方式，或者通过在 Campaign 联系历史记录表中配置一个视图。例如，如果您的一项业务实践是所有商品的有效期都不超过 30 天，那么您应该将联系历史记录数据限制为过去 30 天。要修改联系历史记录数据的保留天数，请打开配置属性 **Campaign | partitions | partition*n*| Interact | contactAndResponseHistTracking** 并设置值 **daysBackInHistoryToLookupContact**。

在联系和响应历史记录模块运行之前，您将不会看到跨会话响应跟踪中的结果。例如，缺省 processSleepIntervalInMinutes 为 60 分钟。因此，至少要在一个小时之后跨会话响应才会显示在您的 Campaign 响应历史记录中。

UACI_TrackingType 表

UACI_TrackingType 表是运行时环境表的一部分。此表定义了与跨会话响应跟踪结合使用的跟踪代码。跟踪代码定义了运行时环境用来将运行时会话中的当前商品与联系和响应历史记录进行匹配的方法。

列	类型	描述
TrackingCodeType	int	表示跟踪代码类型的数字。用于将会话数据中的信息与联系和响应历史记录表进行匹配的 SQL 命令将引用此数字。
名称	varchar(64)	跟踪代码类型的名称。将使用 UACI_TrackingCodeType 保留参数通过 postEvent 方法将此名称传递到会话数据。
Description	varchar(512)	对跟踪代码类型的简要描述。此字段是可选的。

缺省情况下，运行时环境定义了两个跟踪代码类型，如下表中所示。对于任何备用代码，您必须定义一个唯一的 TrackingCodeType。

TrackingCodeType	名称	描述
1	处理代码	UACI 生成的处理代码

TrackingCodeType	名称	描述
2	商品代码	UAC 营销活动商品代码

UACI_XSessResponse

UACI_XSessResponse 表是运行时环境表的一部分。此表用于跨会话响应跟踪。

对于每个受众级别，此表的实例必须存在可用于 Interact 跨会话响应跟踪的联系和响应历史记录数据源中。

列	类型	描述
SeqNumber	bigint	数据行的标识。CrossSessionResponse 服务将按照 SeqNumber 顺序来处理所有记录。
ICID	bigint	交互式渠道标识
AudienceID	bigint	此受众级别的受众标识。此列的名称必须与 Campaign 中定义的受众标识相匹配。样本表包含列 CustomerID。
TrackingCode	varchar(64)	postEvent 方法的 UACIOfferTrackingCode 参数所传递的值。
TrackingCodeType	int	跟踪代码的数字表示。该值必须是 UACI_TrackingType 表中的有效条目。
OfferID	bigint	商品标识（如 Campaign 中所定义）。
ResponseType	int	此记录的响应类型。该值必须是 UA_UsrResponseType 表中的有效条目。
ResponseTypeCode	varchar(64)	此记录的响应类型代码。该值必须是 UA_UsrResponseType 表中的有效条目。
ResponseDate	datetime	响应的日期。
Mark	bigint	此字段的值可标识记录的状态。 • 1 - 正在进行 • 2 - 成功 • NULL - 重试 • -1 - 记录在数据库中的时间已经超过了 purgeOrphanResponseThresholdInMinutes 分钟。 作为数据库管理员维护此表的一部分，您可以检查此字段以了解不匹配的记录，即，值为 -1 的所有记录。值为 2 的所有记录将由 CrossSessionResponse 服务自动除去。
UsrDefinedFields	char(18)	您在使商品响应与联系和响应历史记录匹配时希望包括的任何定制字段。例如，如果您要匹配促销代码，请包括用户定义的促销代码字段。

启用跨会话响应跟踪

使用此过程来启用跨会话响应跟踪。

开始之前

您必须配置联系和响应历史记录模块才能完全利用跨会话响应跟踪。

要使用跨会话响应跟踪，必须配置运行时环境，以便对 Campaign 联系和响应历史记录表具有读访问权。您可以从设计环境中的实际 Campaign 联系和响应历史记录表中读取，也可以从运行时环境数据源中表的副本中读取。配置运行时环境以便对联系和响应历史记录表具有读访问权，这个配置过程独立于任何联系和响应历史记录模块配置。

如果您在匹配处理代码或商品代码之外的对象，那么必须将此对象添加到 UACI_TrackingType 表。

过程

1. 在可供运行时环境访问的联系和响应历史记录表中创建 XSessResponse 表。
2. 在运行时环境的 contactAndResponseHistoryDataSource 类别中定义属性。
3. 为每个受众级别定义 crossSessionResponseTable 属性。
4. 为每个受众级别创建 OverridePerAudience 类别。

跨会话响应商品匹配

缺省情况下，跨会话响应跟踪与处理代码或商品代码进行匹配。crossSessionResponse 服务使用 SQL 命令将处理代码、商品代码或定制代码从会话数据匹配到 Campaign 联系和响应历史记录表。您可以编辑这些 SQL 命令以匹配您定制的任何跟踪代码、商品代码或定制代码。

按处理代码匹配

用于按处理代码进行匹配的 SQL 必须返回此受众级别的 XSessResponse 表中的所有列以及一个名为 OfferIDMatch 的列。OfferIDMatch 列中的值必须是与 XSessResponse 记录中的处理代码对应的 offerId。

以下是与处理代码匹配的缺省生成的 SQL 命令样本。Interact 生成 SQL 以对受众级别使用正确表名称。如果 Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byTreatmentCode > SQL 属性设置为**使用系统生成的 SQL**，那么将使用此 SQL。

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

值 UACI_XsessResponse、UA_DtlContactHist、CustomerID 和 UA_ContactHistory 由 Interact 中的设置来定义。例如，UACI_XsessResponse 由 Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable 配置属性来定义。

如果您已定制了您的联系和响应历史记录表，那么可能需要修订此 SQL 以处理您的表。您将在 Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byTreatmentCode > OverrideSQL 属性中定义 SQL 覆盖。如果您提供某些覆盖 SQL，那么必须将 SQL 属性更改为**覆盖 SQL**。

按商品代码匹配

用于按商品代码进行匹配的 SQL 必须此受众级别的 XSessResponse 表中的所有列以及一个名为 TreatmentCodeMatch 的列。TreatmentCodeMatch 列中的值是与 XSessResponse 记录中的商品标识（和商品代码）一起的处理代码。

以下是与商品代码匹配的缺省生成的 SQL 命令样本。Interact 生成 SQL 以对受众级别使用正确表名称。如果 Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byOfferCode > SQL 属性设置为**使用系统生成的 SQL**，那么将使用此 SQL。

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

值 UACI_XsessResponse、UA_DtlContactHist、CustomerID 和 UA_ContactHistory 由 Interact 中的设置来定义。例如，UACI_XsessResponse 由 Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable 配置属性来定义。

如果您已定制了您的联系和响应历史记录表，那么可能需要修订此 SQL 以处理您的表。您将在 Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byOfferCode > OverrideSQL 属性中定义 SQL 覆盖。如果您提供某些覆盖 SQL，那么必须将 SQL 属性更改为**覆盖 SQL**。

按备用代码匹配

您可以定义 SQL 命令以按照您选择的某些备用代码进行匹配。例如，您可以具有单独于商品或处理代码的促销代码或产品代码。

您必须在 Interact 运行时环境表中的 UACI_TrackingType 表中定义此备用代码。

您必须在 Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byAlternateCode > OverrideSQL 属性中提供 SQL 或存储过程，并且此 SQL 或存储过程必须返回此受众级别的 XSessResponse 表中的所有列以及 TreatmentCodeMatch 和 OfferIDMatch 列。您可以选择返回 OfferIDMatch 以代替 offerCode（对于有 N 个部分的商品代码，将采用格式 offerCode1,offerCode2, ... offerCode*N*）。TreatmentCodeMatch 列和 OfferIDMatch 列（或商品代码列）中的值必须对应于 XSessResponse 记录中的 TrackingCode。

例如，以下 SQL 伪代码与 XSessResponse 表中的 AlternateCode 列匹配。

```
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>
```

其中 *<x>* 是 UACI_TrackingType 表中定义的跟踪代码。

将数据库装入实用程序与运行时环境配合使用

缺省情况下，运行时环境会将会话数据中的联系和响应历史记录数据写入到登台表。但是，在非常活动的生产系统中，对所有数据进行高速缓存（此后运行时才能将数据写入到登台表）所需的内存量可能非常大。您可以将运行时配置为使用数据库装入实用程序以提高性能。

如果启用了数据库装入实用程序，那么运行时会将数据写入到登台文件，而不是在写入到登台表之前将所有联系和响应历史记录保留在内存中。您通过 externalLoaderStagingDirectory 属性来定义包含登台文件的目录的位置。此目录包含若干子目录。第一个子目录是运行时实例目录，其中包含 contactHist 和 respHist 目录。 contactHist 和 respHist 目录包含 *audienceLevelName.uniqueID.currentState* 格式且名称唯一的子目录，这些子目录中包含登台表。

当前状态	描述
CACHE	当前写入到文件的目录的内容。
READY	已准备好进行处理的目录的内容。
RUN	当前写入到数据库的目录的内容。
PROCESSED	已写入到数据库的目录的内容。
ERROR	将目录内容写入到数据库时发生错误。
ATTN	需要注意的目录的内容。即，您可能需要采取一些手动步骤来完成将此目录的内容写入到数据库。
RERUN	已准备好写入到数据库的目录的内容。在收集了问题之后，您应将目录从 ATTN 或 ERROR 重命名为 RERUN。

您可以通过在应用程序服务器启动脚本中定义 interact.runtime.instance.name JVM 属性来定义运行时实例目录。例如，您可以将 -Dinteract.runtime.instance.name=instance2 添加到您的 Web 应用程序服务器启动脚本。如果未设置，那么缺省名称为 DefaultInteractRuntimeInstance。

samples 目录包含一些样本文件以帮助您编写您自己的数据库装入实用程序控制文件。

对运行时环境启用数据库装入实用程序

使用此过程对运行时环境启用数据库装入实用程序。

开始之前

在将运行时环境配置为使用任何命令或控制文件之前，必须首先为您的数据库装入实用程序定义这些命令或控制文件。在同一服务器组的所有运行时服务器上，这些文件必须存在于相同位置中。

Interact 在您的 Interact 运行时服务器安装中的 loaderService 目录中提供了命令和控制文件的样本。

过程

1. 确认运行时环境用户具有配置属性的 Interact > general > systemTablesDataSource 中所定义运行时表数据源的登录凭证。
2. 定义 Interact > general > systemTablesDataSource > loaderProperties 配置属性。
3. 定义 Interact > services >externalLoaderStagingDirectory 属性。
4. 修改 Interact > services > responseHist > fileCache 配置属性（如果需要）。
5. 修改 Interact > services > contactHist > fileCache 配置属性（如果需要）。
6. 重新启动运行时服务器。

事件模式 ETL 过程

要处理大量 IBM Interact 事件模式数据，以及要将该数据用于查询和报告，您可以在任何受支持的服务器上安装独立的抽取、变换和装入 (ETL) 过程以获得最佳性能。

在 Interact 中，给定受众标识的所有事件模式数据都作为单一集合存储在运行时数据库表中。受众标识和模式状态信息将作为二进制大对象 (BLOB) 进行存储。要根据事件模

式来执行任何 SQL 查询或报告，需要这个新的 ETL 过程来将对象分割成多个表存储在目标数据库中。要完成此目的，独立 ETL 过程采用 Interact 运行时数据库表中的事件模式数据，按照您指定的计划来处理该数据，并将其存储在目标数据库中，该数据在数据库中可供 SQL 查询或其他报告使用。

除了将事件模式数据移动和变换至目标数据库外，独立 ETL 过程还将目标数据库中的数据与 Interact 运行时数据库中的当前信息进行同步。例如，如果删除 Interact 运行时中的事件模式，那么将在下次 ETL 过程运行时，从目标数据库中除去该事件模式的已处理数据。事件模式状态信息也会保持最新。因此，目标数据库中存储的事件模式的相关信息全部是当前数据，不是历史信息。

运行独立 ETL 过程

在服务器上启动独立 ETL 过程时，独立 ETL 过程会继续在后台运行，直到停止为止。该过程遵循 Marketing Platform 配置属性中的指示信息来确定其操作期间的频率、数据库连接及其他详细信息。

开始之前

在运行独立 ETL 过程之前，请确保完成下列任务：

- 您必须具有"交互管理"用户角色的许可权。
- 必须将该过程安装在某个服务器上，并针对您的配置正确配置该服务器上以及 Marketing Platform 中的文件。

注：

如果要在 Microsoft Windows 上对除美国英语外的其他语言运行 ETL 过程，请在命令提示符处使用 chcp 来设置您正在使用的语言的代码页。例如，您可以使用下列任一编码：ja_jp=932、zh_cn=936、ko_kr=949 和 ru_ru=1251，并且对于 de_de、fr_fr、it_it、es_es 和 pt_br，使用 1252。为了确保字符显示正确，请先在 Windows 命令提示符中使用 chcp 命令，然后再启动 ETL 过程。

关于此任务

安装并配置独立 ETL 过程之后，您就可以开始启动该过程。

过程

1. 在安装了 ETL 过程的服务器上打开命令提示符。
2. 浏览至包含 ETL 过程的可执行文件的 <Interact_home>/PatternStateETL/bin 目录。
3. 带以下参数运行 command.bat 文件（在 Microsoft Windows 上）或 command.sh 文件（在类 UNIX 操作系统上）：
   -

     -u <username>. 此值必须是有效的 Marketing Platform 用户，并且您必须已向该用户授予对 **TargetDS** 和 **RuntimeDS** 数据源的访问权，ETL 过程将使用这两个数据源。
   -

-p <password>. 将 *<password>* 替换为与您指定的用户相匹配的密码。如果此用户的密码为空，请指定两个双引号（如 -p "" 中所示）。运行命令文件时，密码是可选项；如果在命令中省略了密码，那么在该命令运行时系统将提示您输入密码。

- -c <profileName>。将 *<profileName>* 替换为您在 Marketing Platform 中所创建的 **Interact | PatternStateETL** 配置内指定的确切名称。

  您在这里输入的名称必须与您在创建配置时于**新类别名称**字段中指定的值相匹配。

- start。start 命令是启动该过程必需的。

因此，用于启动该过程的完整命令将采用以下形式：

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

结果

独立 ETL 过程将运行，并继续在后台运行，直到您停止该过程或重新启动服务器为止。

注：

第一次运行该过程时，累积的事件模式数据可能要花大量的时间来运行。该过程以后运行时，将仅处理事件模式数据的最新集合，因此只需较少时间就能完成。

您应该知道还可以向 command.bat 或 command.sh 文件提供 help 自变量，以查看所有可用的选项，如以下示例中所示：

```
command.bat help
```

停止独立 ETL 过程

在服务器上启动独立 ETL 过程时，独立 ETL 过程会继续在后台运行，直到停止为止。

关于此任务

过程

1. 在安装了 ETL 过程的服务器上打开命令提示符。
2. 浏览至包含 ETL 过程的可执行文件的 <Interact_home>/PatternStateETL/bin 目录。
3. 带以下参数运行 command.bat 文件（在 Microsoft Windows 上）或 command.sh 文件（在类 UNIX 操作系统上）：

   - -u <username>. 此值必须是有效的 Marketing Platform 用户，并且您必须已向该用户授予对 **TargetDS** 和 **RuntimeDS** 数据源的访问权，ETL 过程将使用这两个数据源。

-p <password>. 将 *<password>* 替换为与您指定的用户相匹配的密码。如果此用户的密码为空，请指定两个双引号（如 -p "" 中所示）。运行命令文件时，密码是可选项；如果在命令中省略了密码，那么在该命令运行时系统将提示您输入密码。

- -c <profileName>。将 *<profileName>* 替换为您在 Marketing Platform 中所创建的 **Interact | PatternStateETL** 配置内指定的确切名称。

  您在这里输入的名称必须与您在创建配置时于**新类别名称**字段中指定的值相匹配。

- stop. stop 命令是停止该过程必需的。如果使用此命令，那么任何执行中的 ETL 操作将先完成，然后该过程才关闭。

  要关闭 ETL 过程而不等待任何执行中的操作完成，请使用 forcestop，而不要使用 stop。

因此，用于启动该过程的完整命令将采用以下形式：

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

结果

独立 ETL 过程会停止。

第 4 章 商品服务

能够以多种方法来配置 Interact 以增强其选择要呈现的商品的方式。以下各节详细描述了这些可选功能。

商品资格

Interact 的目的是呈现合格商品。简而言之，Interact 根据访问者、渠道和情况来呈现合格商品中的最佳者。

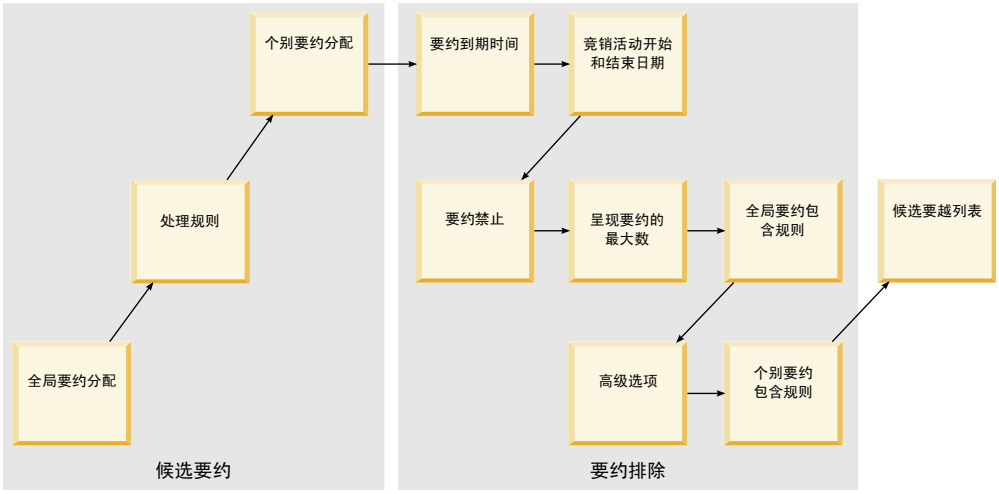
处理规则只是有关 Interact 如何为客户确定合格商品的开始。Interact 具有几个可选功能，您可以实施这几个可选功能来增强运行时环境确定所呈现的商品的方式。这些功能都不能保证某个商品将呈现给客户。这些功能将影响某个商品有资格向客户进行呈现的概率。您可以根据需要来使用其中任意数量的功能以针对您的环境实施最佳解决方案。

您可以影响商品合格性的主要有三个方面：生成候选商品的列表、确定市场营销分数和学习。

生成候选商品的列表

生成候选商品的列表有两个主要阶段。第一个阶段是生成客户可能有资格获得的所有可能商品的列表。第二个阶段是过滤掉客户不再有资格获得的所有商品。在这两个阶段中，您可以在某些环节中影响候选商品列表的生成。

此图显示了候选商品列表生成的阶段。箭头显示了优先顺序。例如，如果某一商品通过**呈现商品的最大次数**过滤器，但是未通过**全局商品包含规则**过滤器，那么运行时环境将排除此商品。



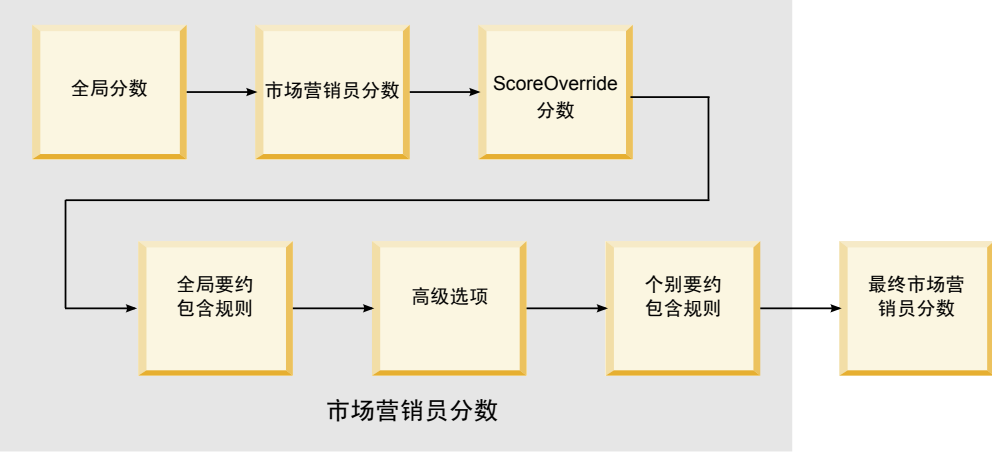
- **全局商品分配** - 您可以使用全局商品表按受众级别来定义全局商品。
- **处理规则** - 定义商品的基本方法，使用"交互策略"选项卡按细分市场和交互点进行定义。
- **个别商品分配** - 您可以使用分数覆盖表按客户来定义特定商品分配。

- **商品截止日期** - 在 Campaign 中创建商品时，您可以定义截止日期。如果某个商品的截止日期已经过去，那么运行时环境将排除此商品。
- **营销活动开始日期和结束日期** - 在 Campaign 中创建营销活动时，可以定义营销活动的开始日期和结束日期。如果营销活动的开始日期尚未出现或者营销活动的结束日期已经过去，那么运行时环境将排除商品。
- **商品禁止** - 您可以使用商品禁止表来为特定受众成员定义商品禁止。
- **呈现商品的最大次数** - 定义交互式渠道时，您应该定义每个会话中向客户呈现某一商品的最大次数。如果商品已呈现了此次数，那么运行时环境将排除商品。
- **全局商品包含规则** - 您可以使用全局商品表来定义布尔表达式，以根据受众级别来过滤商品。如果结果为 false，那么运行时环境将排除此商品。
- **高级选项** - 您可以使用处理规则中的**如果满足以下表达式，那么将此规则视为合格**高级选项，以根据细分市场级别来过滤商品。如果结果为 false，那么运行时环境将排除此商品。
- **个别商品包含规则** - 您可以使用分数覆盖表来定义一个布尔表达式，以根据客户级别来过滤商品。如果结果为 false，那么运行时环境将排除此商品。

计算市场营销分数

有许多种方法来影响（通过使用计算）或覆盖市场营销分数。

此图显示了您可以影响或覆盖市场营销分数的不同阶段。



箭头显示了优先顺序。例如，如果您定义一个表达式来确定某一处理规则在"高级选项"中的市场营销分数，并且在分数覆盖表中定义了一个表达式，那么分数覆盖表中的表达式优先。

- **全局分数** - 您可以使用全局商品表来定义每个受众级别的分数。
- **市场营销人员的分数** - 您可以使用处理规则中的滑块来定义每个细分市场的分数。
- **分数覆盖分数** - 您可以使用分数覆盖表来定义每个客户的分数。
- **全局商品包含规则** - 您可以使用全局商品表来定义用于计算每个受众级别的分数的表达式。
- **高级选项** - 您可以使用处理规则中的**将以下表达式用作营销分数**高级选项来定义一个用于计算每个细分市场的分数的表达式。
- **分数覆盖商品包含规则** - 您可以使用分数覆盖表来定义一个用于计算每个客户的分数的表达式。

影响学习

如果您在使用 Interact 内置学习模块，那么您可以影响学习输出（不仅包含标准学习配置），如学习属性或置信度级别的列表。您可以覆盖学习算法的某些组件，同时仍使用其余组件。

您可以使用缺省商品表和分数覆盖表的 `LikelihoodScore` 和 `AdjExploreScore` 列来覆盖学习。您可以使用 `aci_scoringfeature` 功能脚本将这些列添加到缺省商品表和分数覆盖表。要正确使用这些覆盖，您需要彻底了解 Interact 内置学习。

学习模块获取候选商品以及每个候选商品的市场营销分数的列表，并将其用于最终计算。商品列表与学习属性配合使用来计算客户将接受商品的可能性（接受概率）。通过使用这些概率以及呈现的历史次数在探索和利用之间进行均衡，学习算法可确定商品权重。最终，内置学习获取商品权重，将其乘以最终市场营销分数并返回最终分数。商品按照此最终分数来进行排序。

禁止商品

您可以配置运行时环境以禁止商品。

运行时环境禁止商品的方式有以下几种：

- 交互式渠道的**单次访问期间要显示商品的最大次数**元素。

  您将在创建或编辑交互式渠道时定义**单次访问期间要显示商品的最大次数**。
- 使用商品禁止表。

  您将在概要文件数据库中创建商品禁止表。
- 已超过截止日期的商品。
- 已到期营销活动中的商品。
- 由于未通过商品包含规则而被排除的商品（处理规则高级选项）。
- 已在 Interact 会话中被显式接受或拒绝的商品。如果客户显式接受或拒绝某个商品，那么将在会话期间禁止该商品。

启用商品禁止

使用此过程来启用商品禁止。

关于此任务

您可以配置 Interact 以引用已禁止商品的列表。

过程

1. 为包含受众标识和商品标识的每个受众创建一个新表 `offerSuppressionTable`。
2. 将 `enableOfferSuppressionLookup` 属性设置为 **true**。
3. 将 `Interact > profile > offerSuppressionTable` 属性设置为相应受众的商品禁止表的名称。

商品禁止表

商品禁止表使您可以为特定受众标识禁止商品。例如，如果您的受众为客户，那么您可以为客户 John Smith 禁止商品。您的生产概要文件数据库中必须存在至少一个受众级别的此表的版本。可以通过针对概要文件数据库运行 aci_usrtab SQL 脚本来创建样本商品禁止表 UACI_BlackList。aci_usrtab SQL 脚本位于您的运行时环境安装目录的 ddl 目录中。

必须为每一行定义 AudienceID 和 OfferCode1 字段。如果您的受众标识和商品代码包含多个列，那么可添加其他列。这些列必须与 Campaign 中定义的列名称匹配。例如，如果您通过字段 HHold_ID 和 MemberNum 来定义受众 Customer，那么必须向商品禁止表中添加 HHold_ID 和 MemberNum。

名称	描述
AudienceID	（必需）此列的名称必须与用于在 Campaign 中定义受众标识的列的名称相匹配。如果您的受众标识包含多个列，那么可将其添加到此表中。每行必须包含要将缺省商品分配到的受众标识，例如，customer1。
OfferCode1	（必需）您正在覆盖的商品的商品代码。如果您的商品代码由多个字段组成，那么可添加其他列，例如，OfferCode2 等等。

全局商品和个别分配

您可以配置运行时环境以分配特定商品（不仅包括在"交互策略"选项卡上配置的处理规则）。您可以为受众级别的任何成员定义全局商品，并可以为特定受众成员定义个别分配。例如，您可以为所有家庭定义一个全局商品以了解在什么情况下任何其他商品都不可用，然后为特定家庭 Smith 创建个体商品分配。

您可以按照区域、单元和商品包含规则来同时约束全局商品和个别分配。全局商品和个别分配均通过向生产概要文件数据库中的特定表添加数据来配置。

要使全局商品和个别分配能够正常运行，部署中必须存在所有引用的单元代码和商品代码。为确保必需数据可用，您必须配置缺省单元代码和 UACI_ICBatchOffers 表。

定义缺省单元代码

如果您将缺省商品表或分数覆盖表用于全局或个别商品分配，那么必须定义缺省单元代码。未在缺省商品表或分数覆盖表的特定行中定义单元代码时，将使用 DefaultCellCode。报告将使用此缺省单元代码。

关于此任务

DefaultCellCode 必须与 Campaign 中定义的单元代码格式匹配。此单元代码用于报告中出现的所有商品分配。

如果您定义唯一缺省单元代码，那么可以轻松识别由缺省商品表或分数覆盖表分配的商品。

过程

为 IndividualTreatment 类别中的每个受众级别和表类型定义 DefaultCellCode 属性。

定义处理规则中未使用的商品

如果您使用缺省商品表或分数覆盖表，那么必须确保部署中存在所有商品代码。如果您知道您在缺省商品表或分数覆盖表中使用的所有商品都用于处理规则中，那么部署中存在商品。但是，必须在 UACI_ICBatchOffers 表中定义处理规则中未使用的任何商品。

关于此任务

Campaign 系统表中存在 UACI_ICBatchOffers 表。

过程

使用缺省商品表或分数覆盖表中使用的商品代码来填充 UACI_ICBatchOffers 表。此表具有以下格式：

列名称	类型	描述
ICName	varchar(64)	与商品关联的交互式渠道的名称。如果您使用具有不同交互式渠道的同一个商品，那么必须为每个交互式渠道提供一行。
OfferCode1	varchar(64)	商品代码的第一部分。
OfferCode2	varchar(64)	商品代码的第二部分。
OfferCode3	varchar(64)	商品代码的第三部分。
OfferCode4	varchar(64)	商品代码的第四部分。
OfferCode5	varchar(64)	商品代码的第五部分。

关于全局商品表

全局商品表使您可以在受众级别定义处理。例如，您可以为受众 Household 的每个成员定义一个全局商品。

您可以为 Interact 商品服务的以下元素定义全局设置。

- 全局商品分配
- 全局市场营销人员的分数（通过数字或通过表达式）
- 用于过滤商品的布尔表达式
- 学习可能性和权重（如果您在使用 Interact 内置的学习）
- 全局学习覆盖

分配全局商品

使用此过程将运行时环境配置为针对受众级别分配全局商品，从而忽略处理规则中定义的任何内容。

过程

1. 在概要文件数据库中创建名为 UACI_DefaultOffers 的表。

   要创建具有正确列的 UACI_DefaultOffers 表，请使用 aci_usrtab DDL 文件。
2. 将 Interact > profile > enableDefaultOfferLookup 属性设置为 **true**。

全局商品表

概要文件数据库中必须存在全局商品表。可以通过针对概要文件数据库运行 aci_usrtab SQL 脚本来创建全局商品表 UACI_DefaultOffers。

aci_usrtab SQL 脚本位于您的运行时环境安装目录的 ddl 目录中。

必须为每一行定义 AudienceLevel 和 OfferCode1 字段。其他字段是可选的，用于进一步约束您的商品分配或者在受众级别影响内置的学习。

为了获得最佳性能，您应该在受众级别列上对此表创建索引。

名称	类型	描述
AudienceLevel	varchar(64)	（必需）您将缺省商品分配到的受众级别的名称，例如，customer 或 household。此名称必须与 Campaign 中定义的受众级别相匹配。
OfferCode1	varchar(64)	（必需）缺省商品的商品代码。如果您的商品代码由多个字段组成，那么可添加其他列，例如，OfferCode2 等等。 如果您在添加此商品以提供全局商品分配，那么必须将此商品添加到 UACI_ICBatchOffers 表。
分数	float	用于为此商品分配定义市场营销分数的数字。
OverrideTypeID	int	如果设置为 1，并且商品的候选列表中不存在商品，请将此商品添加到列表并将任何分数数据用于此商品。通常，使用 1 以提供全局商品分配。 如果设置为 0、*null* 或除了 1 之外的任何数字，那么仅当商品的候选列表中存在商品时，才将任何数据用于商品。在大多数情况下，处理规则或个别分配将覆盖此设置。
谓词	varchar(4000)	对于处理规则的高级选项，您可以在此列中输入表达式。可以使用与您在为处理规则编写高级选项时能够使用的相同变量和宏。此列的行为取决于 EnableStateID 列中的值。 • 如果 EnableStateID 为 2，那么此列与**如果满足以下表达式，那么将此规则视为合格**选项（在用于约束此商品分配的处理规则的高级选项中）具有相同作用。此列必须包含一个布尔表达式，并且必须解析为 true 才能包含此商品。 如果无意中定义了一个解析为数字的表达式，那么会将任何非零数字视为 true，将零视为 false。 • 如果 EnableStateID 为 3，那么此列与**将以下表达式用作营销分数**选项（在用于约束此商品的处理规则的高级选项中）具有相同作用。此列必须包含一个解析为数字的表达式。 • 如果 EnableStateID 为 1，那么 Interact 将忽略此列中的任何值。

名称	类型	描述
FinalScore	float	要覆盖最终分数的数字，此最终分数用于对返回商品的最终列表进行排序。如果您已启用了内置学习模块，那么将使用此列。您可以实施您自己的学习以使用此列。
CellCode	varchar(64)	要为其分配此缺省商品的已部署的交互式细分市场的单元代码。如果单元代码由多个字段组成，那么您可添加其他列。 如果 OverrideTypeID 为 0 或空，那么必须提供单元代码。如果不包含单元代码，那么运行时环境将忽略此行数据。 如果 OverrideTypeID 为 1，那么不需要在此列中提供单元代码。如果不提供单元代码，那么运行时环境将此受众级别和表的 DefaultCellCode 属性中定义的单元代码用于报告目的。
区域	varchar(64)	您要将此商品分配应用到的分区的名称。如果为 NULL，那么这适用于所有区域。
EnableStateID	int	此列中的值定义 Predicate 列的行为。 • **1** - 不使用 Predicate 列。 • **2** - 使用 Predicate 作为布尔值以过滤商品。这与处理规则中**如果满足以下表达式，那么将此规则视为合格**高级选项遵循相同的规则。 • **3** - 使用 Predicate 来定义市场营销人员的分数。这与处理规则中**将以下表达式用作营销分数**高级选项遵循相同的规则。 此列为 NULL 或者值不为 2 或 3 的任何行将忽略 Predicate 列。
LikelihoodScore	float	此列仅用于影响内置学习。您可以将此列与 aci_scoringfeature ddl 一起添加。
AdjExploreScore	float	此列仅用于影响内置学习。您可以将此列与 aci_scoringfeature ddl 一起添加。

关于分数覆盖表

分数覆盖表使您可以定义对某一受众标识或个别级别的处理。例如，如果您的受众级别为"访问者"，那么您可以为特定访问者创建覆盖。

您可以为 Interact 商品服务的以下元素定义覆盖。

- 个别商品分配
- 个别市场营销人员的分数（通过数字或通过表达式）
- 用于过滤商品的布尔表达式
- 学习可能性和权重（如果您在使用内置的学习）
- 个别学习覆盖

配置分数覆盖

您可以配置 Interact 以使用通过建模应用程序生成的分数，而不是使用市场营销分数。

过程

1. 针对您要为其提供覆盖的每个受众级别，创建一个分数覆盖表。

   要创建具有正确列的样本分数覆盖表，请使用 aci_usrtab DDL 文件。
2. 将 Interact > Profile > enableScoreOverrideLookup 属性设置为 **true**。
3. 将 scoreOverrideTable 属性设置为您要为其提供覆盖的每个受众级别的分数覆盖表的名称。

   不需要为每个受众级别提供分数覆盖表。

分数覆盖表

生产概要文件数据库中必须存在分数覆盖表。可以通过针对概要文件数据库运行 aci_usrtab SQL 脚本来创建样本分数覆盖表 UACI_ScoreOverride。

aci_usrtab SQL 脚本位于您的运行时环境安装目录的 ddl 目录中。

必须为每一行定义 *AudienceID*、OfferCode1 和 Score 字段。其他字段中的值是可选的，用于进一步约束您的个别商品分配或为内置学习提供分数覆盖信息。

名称	类型	描述
AudienceID	varchar(64)	（必需）此列的名称必须与用于在 Campaign 中定义受众标识的列的名称相匹配。aci_usrtab ddl 文件创建的样本表作为 CustomerID 列来创建此列。如果您的受众标识包含多个列，那么可将其添加到此表中。每行必须包含要将个别商品分配到的受众标识，例如，customer1。为了获得最佳性能，您应该对此列创建索引。
OfferCode1	varchar(64)	（必需）商品的商品代码。如果您的商品代码由多个字段组成，那么可添加其他列，例如，OfferCode2 等等。 如果您在添加此商品以提供个别商品分配，那么必须将此商品添加到 UACI_ICBatchOffers 表。
分数	float	用于为此商品分配定义市场营销分数的数字。
OverrideTypeID	int	如果设置为 0、*null*（或除了 1 之外的任何数字），那么仅当商品的候选列表中存在商品时，才将任何数据用于商品。通常，使用 0 以提供分数覆盖。您必须提供一个单元代码。 如果设置为 1，并且商品的候选列表中不存在商品，请将此商品添加到列表并将任何分数数据用于此商品。通常，使用 1 以提供个别商品分配。

名称	类型	描述
谓词	varchar(4000)	对于处理规则的高级选项，您可以在此列中输入表达式。可以使用与您在为处理规则编写高级选项时能够使用的相同变量和宏。此列的行为取决于 EnableStateID 列中的值。 • 如果 EnableStateID 为 2，那么此列与**如果满足以下表达式，那么将此规则视为合格**选项（在用于约束此商品分配的处理规则的高级选项中）具有相同作用。此列必须包含一个布尔表达式，并且必须解析为 true 才能包含此商品。 如果无意中定义了一个解析为数字的表达式，那么会将任何非零数字视为 true，将零视为 false。 • 如果 EnableStateID 为 3，那么此列与**将以下表达式用作营销分数**选项（在用于约束此商品的处理规则的高级选项中）具有相同作用。此列必须包含一个解析为数字的表达式。 • 如果 EnableStateID 为 1，那么 Interact 将忽略此列中的任何值。
FinalScore	float	要覆盖最终分数的数字，此最终分数用于对返回商品的最终列表进行排序。如果您已启用了内置学习模块，那么将使用此列。您可以实施您自己的学习以使用此列。
CellCode	varchar(64)	要为其分配此商品的交互式细分市场的单元代码。如果单元代码由多个字段组成，那么您可添加其他列。 如果 OverrideTypeID 为 0 或空，那么必须提供单元代码。如果不包含单元代码，那么运行时环境将忽略此行数据。 如果 OverrideTypeID 为 1，那么不需要在此列中提供单元代码。如果不提供单元代码，那么运行时环境将此受众级别和表的 DefaultCellCode 属性中定义的单元代码用于报告目的。
区域	varchar(64)	您要将此商品分配应用到的分区的名称。如果为 NULL，那么这适用于所有区域。
EnableStateID	int	此列中的值定义 Predicate 列的行为。 • **1** - 不使用 Predicate 列。 • **2** - 使用 Predicate 作为布尔值以过滤商品。这与处理规则中**如果满足以下表达式，那么将此规则视为合格**高级选项遵循相同的规则。 • **3** - 使用 Predicate 来定义市场营销人员的分数。这与处理规则中**将以下表达式用作营销分数**高级选项遵循相同的规则。 此列为 NULL 或者值不为 2 或 3 的任何行将忽略 Predicate 列。
LikelihoodScore	float	此列仅用于影响内置学习。您可以将此列与 aci_scoringfeature ddl 一起添加。

名称	类型	描述
AdjExploreScore	float	此列仅用于影响内置学习。您可以将此列与 aci_scoringfeature ddl 一起添加。

Interact 内置学习概述

尽管您可以尽一切可能来确保您向正确的细分市场建议正确的商品，但是您始终可以从访问者的实际选择中学习一些内容。您的访问者的实际行为应该影响您的策略。您可以获取响应历史记录并通过某些建模工具来运行，以获取您可以包含在交互式流程图中的分数。

但是，此数据不是实时的。

Interact 为您提供了两个选项以用于实时地从访问者的操作中进行学习。

- 内置学习模块 - 运行时环境具有一个基于 Naive Bayesian 的学习模块。此模块监视您选择的客户属性并使用该数据来帮助选择要呈现的商品。
- 学习 API - 运行时环境还具有一个学习 API 可供您编写自己的学习模块。

您不一定必须使用学习。缺省情况下，将禁用学习。

Interact 学习模块

Interact 学习模块监视访问者对商品的响应以及访问者属性。

学习模块方式

学习模块有两个常规方式：

- 探索 - 学习模块提供商品，以便可以收集足够的响应数据来优化采用方式期间使用的估算。在探索期间提供的商品不一定反映最佳选择。
- 采用 - 在探索阶段收集了足够数据之后，学习模块将使用概率来帮助选择要呈示的商品。

学习模块使用两个属性在探索与采用方式之间轮换。这两个属性为：

- 您使用 confidenceLevel 属性配置的置信度级别。
- 您使用 percentRandomSelection 属性配置的学习模块呈示随机商品的概率。

置信度级别属性

您将 confidenceLevel 设置为一个百分比，它表示在将学习模块对某一商品给出的分数用于仲裁之前，您对学习模块的确信度（或置信度）。首先，在学习模块没有任何可使用的数据时，学习模块完全依赖于市场营销分数。在每个商品都已呈示 minPresentCountThreshold 所定义的次数之后，学习模块进入探索方式。如果不处理大量数据，那么学习模块便不确信其计算的百分比是否正确。因此，其停留在探索方式中。

学习模块为每个商品分配权重。为了计算权重，学习模块将使用一个公式，该公式接受已配置的置信度级别、历史接受数据和当前会话数据作为输入。此公式从根本上均衡探索和采用，并返回相应权重。

随机选择属性

为确保系统不偏向于在早期阶段业绩最好的商品，Interact 会将一个随机商品呈现 percentRandomSelection% 的时间。此随机商品百分比会强制学习模块推荐最成功商品以外的商品，以确定其他商品在曝光度增加的情况下是否将更加成功。例如，如果您将 percentRandomSelection 配置为 5，那么学习模块将花费 5% 的时间来呈示随机商品，并将响应数据纳入到计算中。

您可以在"交互式渠道"窗口的"交互点"选项卡上为每个区域设置**随机百分比**，以指定随机选择所返回的商品的变化，不考虑分数。

学习模块确定商品的方式

学习模块按照以下方式来确定所呈现的商品。

1. 计算访问者选择商品的概率。
2. 使用步骤 1 中的概率来计算商品权重，并确定要进入探索方式还是采用方式。
3. 使用市场营销分数以及步骤 2 中的商品权重来计算每个商品的最终分数。
4. 按照步骤 3 中确定的分数来对商品进行排序，然后返回所请求数量的顶级商品。

例如，学习模块确定访问者有 30% 的可能性会接受商品 A，有 70% 的可能性会接受商品 B，并确定采用此信息。通过处理规则，商品 A 和商品 B 的市场营销分数分别为 75 和 55。但是，步骤 3 中的计算使商品 B 的最终分数高于商品 A 的最终分数，因此，运行时环境推荐商品 B。

权重因子属性

学习还基于 recencyWeightingFactor 属性和 recencyWeightingPeriod 属性。这些属性使您可以为更近期的数据（相对于较早的数据）增大权重。recencyWeightingFactor 是要给予近期数据的权重。recencyWeightingPeriod 是代表"近期"的时间长度。例如，将 recencyWeightingFactor 配置为 0.30，并将 recencyWeightingPeriod 配置为 24。这些设置表示，前 24 小时的数据占所有被考虑数据的 30%。对于一周的数据价值来说，前六天的所有数据将占总数据的 70%，最后一天的数据占总数据的 30%。

写入的登台表数据

每个会话都将以下数据写入到学习登台表：

- 商品联系
- 接受商品
- 学习属性

聚集器按照可配置的时间间隔从登台表中读取数据，编译数据，然后将其写入到表中。学习模块读取此聚集数据并将其用于计算中。

启用学习模块

所有运行时服务器都具有内置学习模块。缺省情况下，将禁用此学习模块。通过更改配置属性来启用学习模块。

过程

在运行时环境的 Marketing Platform 中，编辑 Interact > offerserving 类别中的以下配置属性。

配置属性	设置
optimizationType	**BuiltInLearning**

学习属性

学习模块学习使用访问者属性和商品接受数据。您可以选择要监视的访问者属性。这些访问者属性可以是客户概要文件中的任何对象，其中包括您实时收集的某些事件参数。

在学习中不支持维度表中的属性。

尽管您可以配置任意数量的属性来进行监视，但是 IBM 建议您配置 10 个以内的静态与动态学习属性之间的学习属性，并遵循以下准则。

- 选择独立属性。

  请勿选择类似属性。例如，如果您创建一个名为 HighValue 的属性，并且该属性由某个基于工资的计算来定义，那么不要同时选择 HighValue 和 Salary。类似属性无助于学习算法。
- 选择具有离散值的属性。

  如果属性具有值范围，那么您必须选择一个确切值。例如，如果您希望使用工资作为属性，那么您应为每个工资范围提供一个特定值：范围 20,000-30,000 应为 A，30,001-40,000 应为 B，以此类推。
- 限制您跟踪的属性数量，以便不会妨碍性能。

  您可以跟踪的属性数量取决于您的性能需求以及您的 Interact 安装。如果可以，请使用其他建模工具（如 PredictiveInsight）来确定前十个预测属性。您可以将学习模块配置为自动清理非预测性并且还具有性能成本的属性。

您可以通过同时定义监视的属性数量以及要对每个属性监视的值数量来管理性能。Campaign > partitions > partition1 > Interact > learning > maxAttributeNames 属性定义您跟踪的访问者属性的最大数量。maxAttributeValues 属性定义了您为每个属性跟踪的值的最大数量。所有其他值都将被分配到 otherAttributeValue 属性的值所定义的类别中。但是，学习引擎仅跟踪其遇到的第一个值。例如，您正在跟踪访问者属性"眼睛颜色"。您仅关注于值"blue"、"brown"和"green"，因此您将 maxAttributeValues 设置为 3。但是前三个访问者的值为"blue""brown"和"hazel"。 这表示，绿色眼睛的所有访问者将被分配到 otherAttributeValue。

您还可以使用动态学习属性，它使您可以更加具体地定义您的学习标准。动态学习属性使您可以作为单个条目来学习两个属性的组合。例如，请考虑以下概要文件信息：

访问者标识	卡类型	卡余额
1	Gold Card	1000 美元
2	Gold Card	9,000 美元

访问者标识	卡类型	卡余额
3	Bronze Card	1000 美元
4	Bronze Card	9,000 美元

如果您使用标准学习属性，那么您只能单独学习卡类型和余额。根据"卡类型"，访问者 1 和 2 将被一起分到同一个组；根据"卡余额"，访问者 2 和 4 将被分到同一个组。这可能不是商品接受行为的精确谓词。如果金卡持有人趋向于具有更高的余额，那么访问者 2 的行为可能完全不同于访问者 4，这将偏离标准学习属性。但是，如果您使用动态学习属性，将单独学习其中每个访问者，并且预测将更加精确。

如果您使用动态学习属性，并且访问者对某个属性具有两个有效值，那么学习模块将选择其发现的第一个值。

如果将 enablePruning 属性设置为 yes，那么学习模块将通过算法来确定哪些属性不可预测并且在计算权重时停止考虑这些属性。例如，如果您在跟踪一个表示头发颜色的属性，并且学习模块根据访问者的头发颜色确定不存在接受商品的模式，那么学习模块将停止考虑头发颜色属性。每次学习聚集过程运行时（由 aggregateStatsIntervalInMinutes 属性来定义），便对属性进行重新评估。还将清理动态学习属性。

定义学习属性

使用此过程来定义学习属性。

关于此任务

您最多可以配置 maxAttributeNames 个访问者属性。

(learningAttributes) 是用于创建新学习属性的模板。您必须为每个属性输入一个新名称。不能创建两个具有相同名称的类别

过程

在设计环境的 Marketing Platform 中，编辑 Campaign > partitions > partition*n* > Interact > learning 类别中的以下配置属性。

配置属性	设置
attributeName	attributeName 必须与概要文件数据中"名称/值"对的名称匹配。此名称不区分大小写。

定义动态学习属性

要定义动态学习属性，您必须填充学习数据源中的 UACI_AttributeList 表。

此表中的所有列均为 varchar(64) 类型。

列	描述
AttributeName	要作为学习依据的动态属性的名称。此值必须是 AttributeNameCol 中可能存在的实际值。

列	描述
AttributeNameCol	可以在其中找到 AttributeName 的标准列名（从概要文件表开始的分层结构）。此列名称不必是标准学习属性。
AttributeValueCol	可以在其中找到 AttributeName 的关联值的标准列名（从概要文件表开始的分层结构）。

例如，考虑以下概要文件表及其相关联的维度表。

表 *6. MyProfileTable*

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

表 *7. MyDimensionTable*

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

以下是与卡类型和余额相匹配的样本 UACI_AttributeList 表。

表 *8. UACI_AttributeList*

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

将运行时环境配置为识别外部学习模块

您可以使用学习 Java™ API 来编写您自己的学习模块。您必须将运行时环境配置为在 Marketing Platform 中识别您的学习实用程序。

关于此任务

您必须重新启动 Interact 运行时服务器以使这些更改生效。

过程

1. 在运行时环境的 Marketing Platform 中，编辑 Interact > offerserving 类别中的以下配置属性。学习优化器 API 的配置属性位于 Interact > offerserving > External Learning Config 类别中。

配置属性	设置
optimizationType	**ExternalLearning**
externalLearningClass	外部学习的类名
externalLearningClassPath	运行时服务器上用于外部学习的类文件或 JAR 文件的路径。如果您在使用服务器组并且所有运行时服务器都引用 Marketing Platform 的同一实例，那么每个服务器必须在相同位置具有这些类文件或 JAR 文件的副本。

2. 重新启动 Interact 运行时服务器以使这些更改生效。

第 5 章 了解 Interact API

Interact 向各种广泛接触点动态提供商品。例如，您可以配置运行时环境和您的接触点以将消息发送给您的呼叫中心员工，告知其最佳向上销售或交叉销售潜在客户（其已经通过特定类型的服务查询进行了致电）。您还可以配置运行时环境和接触点以向进入了 Web 站点的特定区域的客户（访问者）提供定制商品。

Interact 应用程序编程接口 (API) 使您可以配置接触点和运行时服务器以共同提供最佳可能商品。使用 API，接触点可以从运行时服务器中请求信息以将访问者分配到组（细分市场）并根据该细分市场来呈现商品。您还可以记录数据以供稍后分析，从而优化您的商品呈现策略。

Interact API 还允许最终用户客户机通过 JavaScript 与服务器通信。

为了向您提供将 Interact 与您的环境进行集成方面的最大可能灵活性，IBM 提供了一个可使用 Interact API 来访问的 Web Service。

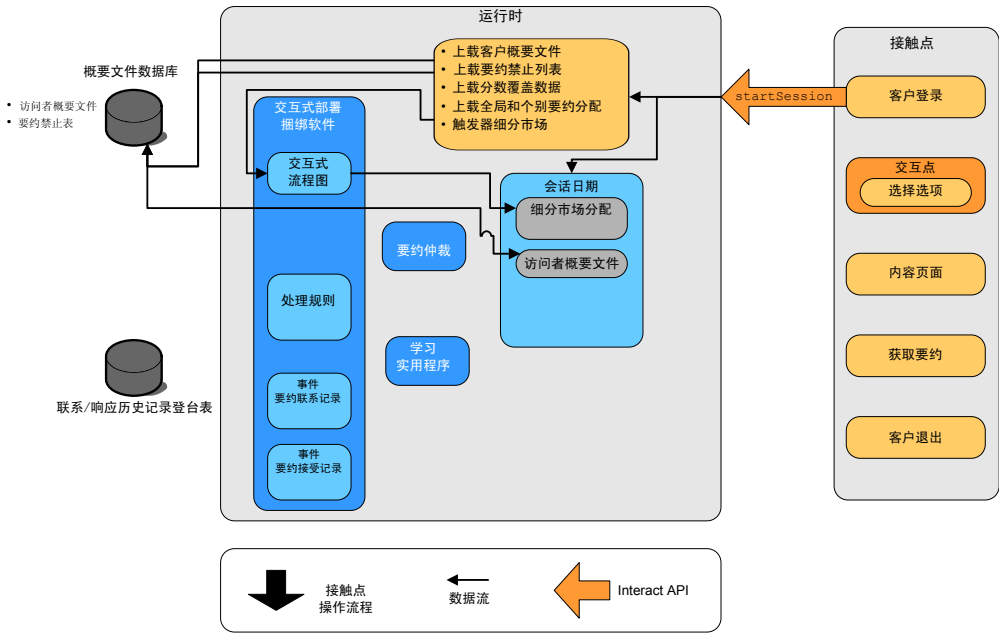
Interact API 数据流

此示例说明了 API 在接触点与运行时环境之间的工作方式。访问者仅执行四个操作 - 登录，浏览到显示商品的页面，选择商品，然后注销。您可以在性能需求的限制内，将您的集成设计为所需要的复杂程度。

此图显示了 Interact API 的简单实现。

访问者登录到 Web 站点并浏览至显示商品的页面。访问者选择了一个商品，然后注销。此交互虽然简单，但是在接触点和运行时服务器中都发生了多个事件：

1. 启动会话
2. 浏览至页面
3. 选择商品
4. 关闭会话



启动会话

在访问者登录时，将触发 startSession。

startSession 方法执行四个操作：

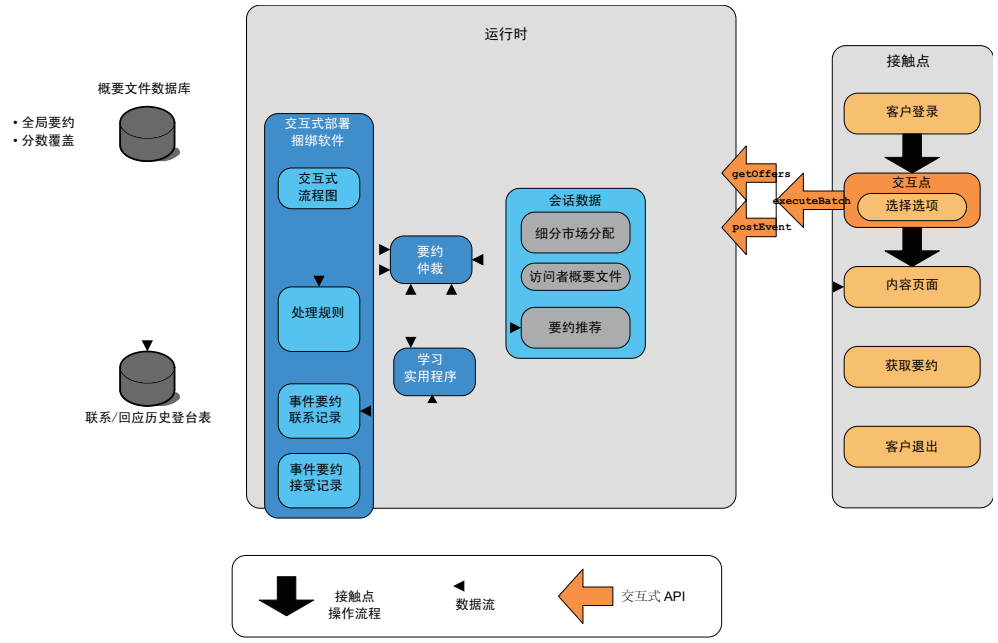
1. 创建新的运行时会话
2. 发送请求以将客户概要文件数据装入到会话
3. 发送请求以使用概要文件数据并启动交互式流程图，以将客户置于细分市场。此流程图运行是异步的。
4. 运行时服务器将任何商品禁止信息以及全局和个别商品处理信息装入到会话。在会话期间，会话数据存放在内存中。

浏览至页面

访问者浏览站点，直至访问达到了预定义的交互点。在图中，第二个交互点（选择选项）是访问者单击链接（用于呈现一组商品）的位置。接触点管理员已将该链接配置为触发 executeBatch 方法以选择商品。

选择商品

此图显示了触发 executeBatch 方法的 API 调用。

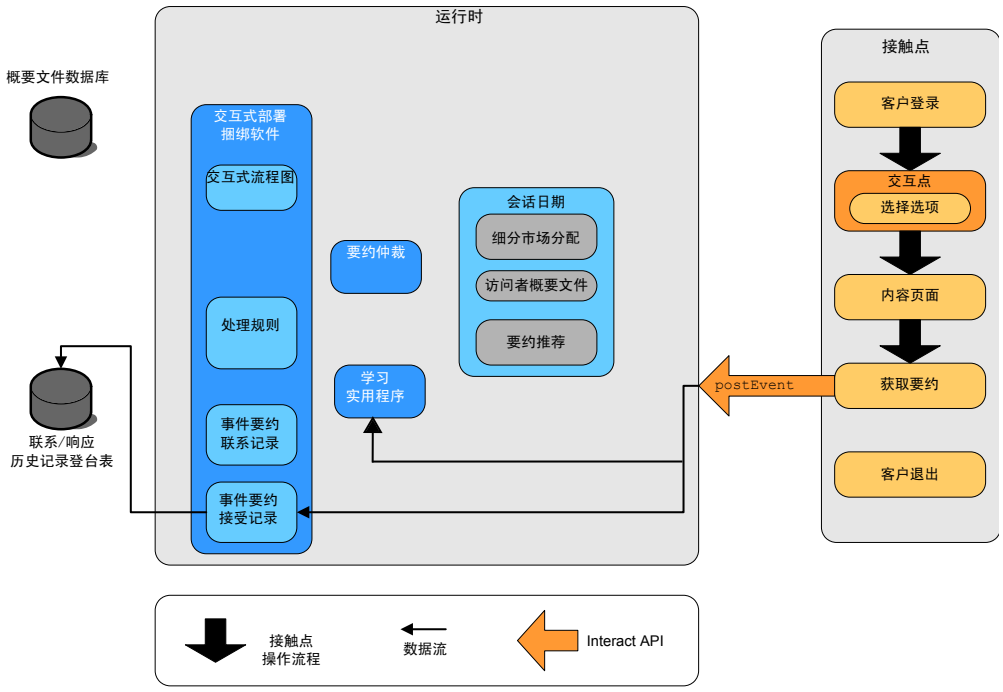


executeBatch 方法使您可以在对运行时服务器的单个调用中调用多个方法。此特定 executeBatch 将调用两个其他方法：getOffers 和 postEvent。getOffers 方法请求商品的列表。运行时服务器使用细分市场数据、商品禁止列表、处理规则以及学习模块来推荐一组商品。运行时服务器返回一组在内容页面上显示的商品。

postEvent 方法触发在设计环境中定义的其中一个事件。在此特定情况下，事件会发送请求以将呈示的商品记录到联系历史记录。

访问者选择其中一个商品（选取商品）。

此图显示了 postEvent 方法。

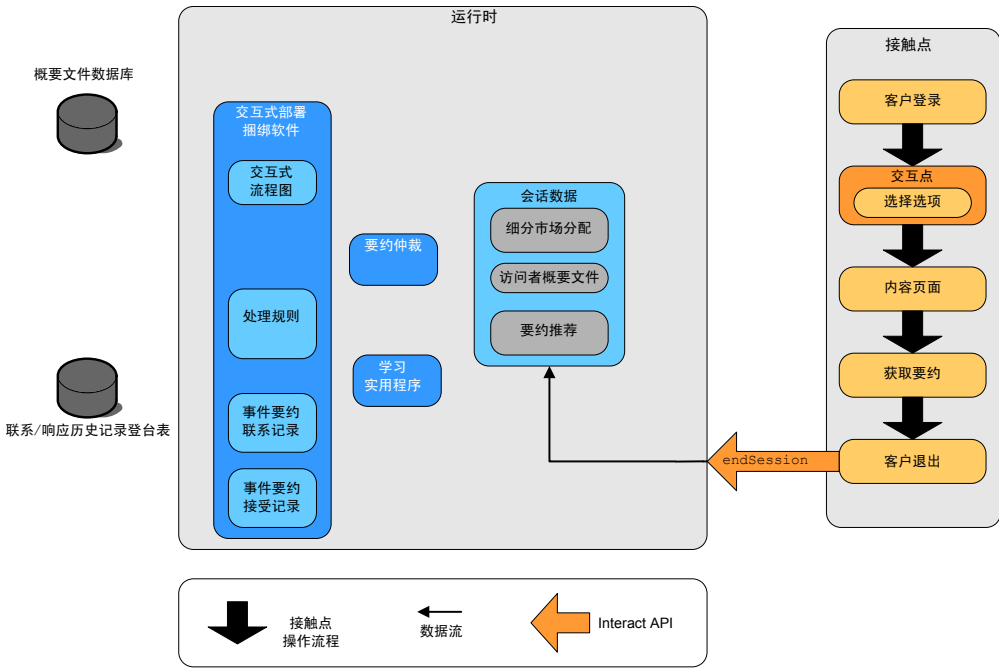


与选择商品相关联的用户界面控件已配置为发送另一个 postEvent 方法。此事件发送一个请求以将商品接受记录到响应历史记录。

关闭会话

在访问者选择商品之后，访问者已完成对 Web 站点的访问，然后会进行注销。注销命令链接到 endSession 方法。

此图显示了 endSession 方法。



endSession 方法关闭会话。如果访问者忘记了注销，那么有一个可配置的会话超时可确保所有会话最终都将结束。如果您希望保留传递到会话的任何数据（如 startSession 或 setAudience 方法的参数中包含的信息），请与创建交互式流程图的人员协作。创建交互式流程图的人员可以使用快照进程将该数据写入到数据库中，以避免会话结束和该数据丢失。然后，可以使用 postEvent 方法来调用包含快照流程的交互式流程图。

简单交互规划示例

在此示例中，您正在为一家移动电话公司的 Web 站点设计交互。您将创建三个不同商品，设置对这些商品的日志记录，指定商品的处理代码，并显示一系列链接至这些商品的照片。

设计流程

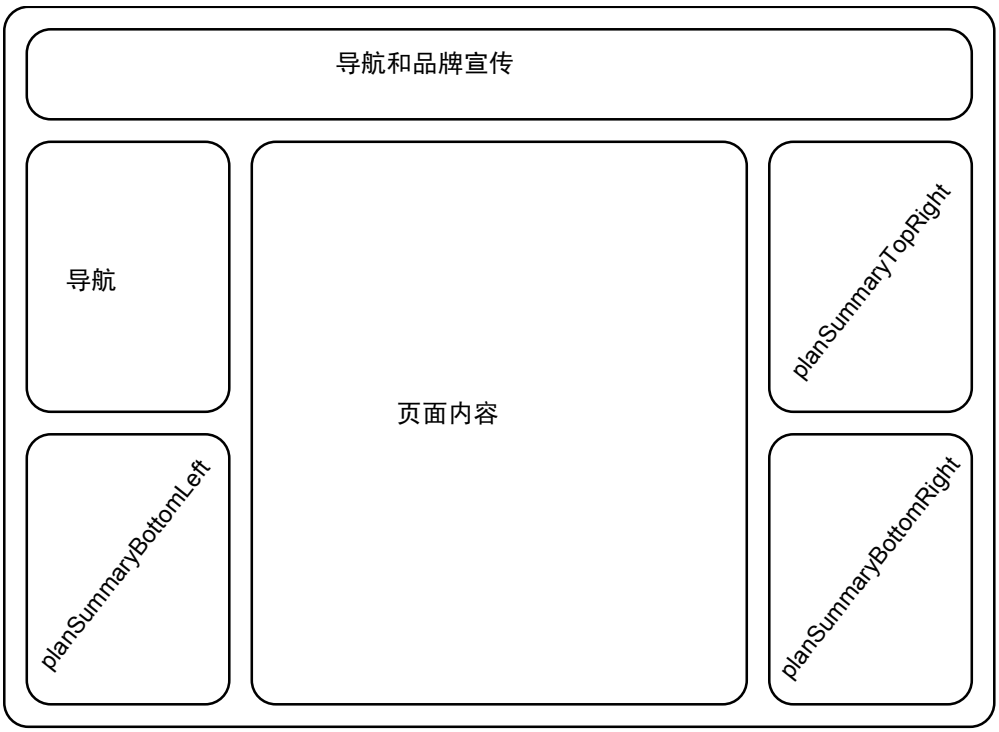
要为此客户机设计交互，请执行以下操作：

1. 确定客户机的摘要页面的需求
2. 为商品需求创建交互点
3. 配置对商品的日志记录
4. 创建处理代码
5. 将一系列轮换图像链接至商品

此示例是编写集成的基本方式，并未显示编写集成的最佳方式。例如，这些示例中均不包含使用 Response 类的任何错误检查。

确定移动电话计划摘要页面的需求

下图显示了移动电话计划摘要页面中的布局。



请定义以下各项以满足移动电话合约摘要页面的需求。

需求	实施
在专用于升级相关商品的区域中显示的一项商品 必须定义页面上用于显示升级商品的区域。此外，在 Interact 选取了要显示的商品之后，必须记录信息。	• 交互点：ip_planSummaryBottomRight • 事件：evt_logOffer
两个用于手机升级的商品 必须定义页面上用于显示手机升级的每个区域。	• 交互点：ip_planSummaryTopRight • 交互点： ip_planSummaryBottomLeft
要进行分析，您需要分别记录已接受和已拒绝的商品。	• 事件：evt_offerAccept • 事件：evt_offerReject
您还知道，只要您记录商品联系、接受或拒绝，就必须传递商品的处理代码。	NameValuePair
在页面上显示三个轮换图像。将这些图像链接至商品。	

创建交互点

现在，您可以请设计环境用户来为您创建交互点和事件，而您同时开始编写用于与接触点集成的代码。

对于每个将显示商品的交互点，您需要首先获取商品，然后抽取显示该商品所需要的信息。例如，请求位于 Web 页面的右下方区域的商品 (planSummaryBottomRight)

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

此响应调用将返回包括 OfferList 响应的响应对象。但是，您的 Web 页面无法使用 OfferList 对象。您需要商品的图像文件，并且您知道此图像文件是其中一个商品属性 (offerImg)。您需要从 OfferList 中抽取所需要的商品属性。

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

配置日志记录

由于您在显示商品，因此您希望将其记录为联系。

```
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)
```

无需单独调用其中每个方法，而是可以对 Web 页面的 planSummaryBottomLeft 部分使用 executeBatch 方法，如以下示例中所示。

```
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);
```

在此示例中，您不需要定义 UACIOfferTrackingCode。如果您不提供 UACIOfferTrackingCode，那么 Interact 运行时服务器会自动将最近一次建议的处理列表记录为联系。

创建处理代码

必要时，您将创建 NameValuePair 以包含处理代码，如以下示例中所示。

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

将图像链接至商品

对于页面上用于显示手机升级的第二个区域，您编写了某些内容以更改每隔 30 秒显示一次的图像。您决定轮换三个图像，因此应使用以下代码来检索要进行高速缓存的一组商品，以在用于轮换图像的代码中使用。

```
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
  }
 }
}
```

您必须将客户机代码编写为从本地高速缓存中进行访存和记录，以在显示每个商品的图像之后，仅联系该商品一次。要记录联系，需要一如既往地发布 UACITrackingCode 参数。每个商品具有不同的跟踪代码。

```
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);
  }
 }
}
```

对于每个商品，如果单击了该商品，那么您应记录已接受的商品和已拒绝的商品。（在此情况下，未显式选择的商品将被视为已拒绝。）以下是在选择了 ip_planSummaryTopRight 商品的情况下的示例：

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

实际上，最好是通过 executeBatch 方法来发送这三个 postEvent 调用。

设计 Interact API 集成

构建与接触点的 Interact API 集成要求您在开始实现之前进行某些设计。您需要与您的市场营销团队协作以决定您希望运行时环境在接触点中的哪些位置中提供商品（定义您的交互点）以及您要使用哪些类型的跟踪或交互式功能（定义您的事件）。

在设计阶段中，这些可能只是大纲。例如，对于一个电信 Web 站点，那么客户的合约摘要页面应显示一个有关合约升级的商品和两个用于电话升级的商品。

在您的公司已决定与客户进行交互的位置和方式之后，您需要使用 Interact 来定义详细信息。流程图作者需要设计一个将在进行重新细分事件时使用的交互式流程图。您需要决定交互点和事件的数量及名称，以及需要为正确细分市场、事件发布和商品检索所传递的数据。设计环境用户定义交互式渠道的交互点和事件。然后，当您在运行时环境中编写用于与您的接触点进行集成的代码时，您将使用这些名称。您还应定义所需要的度量值信息，以定义需要记录商品联系和响应的时间。

要考虑的问题

设计交互时，请牢记不合格的商品、无法访问的运行时服务器以及流程计时对交互的影响。特别是定义商品拒绝时。请考虑可以增强交互的可选产品功能部件。

设计交互时：

创建一些缺省填充内容

为可以呈现商品的每个交互点创建缺省填充内容，例如，亲切的品牌宣传消息或空内容。如果在当前情况下，没有任何适合于向当前访问者提供的商品，那么将使用此填充内容。您应将此缺省填充内容指定为交互点的缺省字符串。

包括呈现内容的替代方法

包括某个呈现内容的方法，以在接触点由于某个不可预见原因而无法访问运行时服务器组的情况下使用。

考虑运行流程图所需的时间

当您触发会将访问者重新细分的事件（其中包括 `postEvent` 和 `setAudience`）时，请牢记运行流程图确实需要一些时间。`getOffers` 方法将等待细分过程完成，然后才会运行。过于频繁地进行重新细分可能影响 `getOffers` 调用的响应性能。

决定"商品拒绝"所代表的含义

某些报告（例如"渠道商品绩效摘要"报告）会提供商品被拒绝的次数。此报告会显示 `postEvent` 已触发"记录商品拒绝"操作的次数。您需要确定"记录商品拒绝"操作是针对实际拒绝（例如，单击带有**不，谢谢**标签的链接），还是针对忽略的商品（例如，某个页面显示三个不同横幅广告，而用户未选择其中任何一个）。

确定要使用的商品选择功能

有若干可选功能，您可以使用它们来增强 Interact 商品选择功能。这些功能包括：

- 学习
- 商品禁止
- 单独商品分配
- 用于提供商品的其他元素

您需要确定这些可选功能中的多少个功能（如果有）将会增强您的交互。

第 6 章 管理 IBM Interact API

只要使用 startSession 方法，您便将在运行时服务器上创建 Interact 运行时会话。您可以使用配置属性在运行时服务器上管理会话。

在您实施与您的接触点的 Interact 集成时，您可能需要配置这些设置。

这些配置属性位于 sessionManagement 类别中。

语言环境和 Interact API

您可以对非英语接触点使用 Interact。API 中的接触点和所有字符串使用为运行时环境用户定义的语言环境。

您只能为每个服务器组选择一个语言环境。

例如，在运行时环境中，您创建两个用户，用户语言环境设置为英语的 asm_admin_en 以及用户语言环境设置为法语的 asm_admin_fr。如果您的接触点是专为操法语者而设计，请将运行时环境的 asmUserForDefaultLocale 属性定义为 asm_admin_fr。

关于 JMX 监视

Interact 提供了您可以通过 JMX 监视应用程序来访问的 Java 管理扩展 (JMX) 监视服务。此 JMX 监视使您可以监视和管理您的运行时服务器。

JMX 属性提供了有关运行时服务器的大量详细信息。例如，JMX 属性 ErrorCount 给出了自最近一次重置或系统启动之后已记录的错误消息数量。您可以使用此信息来查看您的系统中出现错误的频率。如果您已对 Web 站点进行编码以仅调用结束会话，并且某人完成了事务，那么您还可以将 startSessionCount 与 endSessionCount 比较以查看不完整事务的数量。

Interact 支持 JSR 160 所定义的 RMI 和 JMXMP 协议。您可以使用兼容 JSR160 的 JMX 客户机来连接到 JMX 监视服务。

只能通过 JMX 监视来监视交互式流程图。有关交互式流程图的信息不显示在 Campaign 监视中。

注：如果您将 IBM WebSphere® 与节点管理器配合使用，那么您必须定义通用 JVM 参数以启用 JMX 监视。

将 Interact 配置为使用 RMI 协议进行 JMX 监视

使用此过程将 Interact 配置为使用 RMI 协议进行 JMX 监视。

关于此任务

RMI 协议的缺省监视地址是 service:jmx:rmi:///jndi/rmi://*RuntimeServer*:*port*/interact。

过程

在运行时环境的 Marketing Platform 中，编辑 Interact > monitoring 类别中的以下配置属性。

配置属性	设置
协议	**RMI**
port	JMX 服务的端口号
enableSecurity	**False** RMI 协议的 Interact 实现不支持安全性。

将 Interact 配置为使用 JMXMP 协议进行 JMX 监视

使用此过程将 Interact 配置为使用 JMXMP 协议进行 JMX 监视。

开始之前

JMXMP 协议在类路径中需要按以下顺序的两个额外库：InteractJMX.jar 和 jmxremote_optional.jar。这两个文件都位于您的运行时环境安装的 lib 目录中。

关于此任务

如果启用安全性，那么用户名和密码必须与运行时环境的 Marketing Platform 中的用户相匹配。不能使用空密码。

JMXMP 协议的缺省监视地址是 service:jmx:jmxmp://*RuntimeServer*:*port*。

过程

1. 验证 InteractJMX.jar 和 jmxremote_optional.jar 库是否在类路径中并且顺序正确。如果它们不在类路径中，请将它们添加到类路径中。
2. 在运行时环境的 Marketing Platform 中，编辑 Interact > monitoring 类别中的以下配置属性。

配置属性	设置
协议	**JMXMP**
port	JMX 服务的端口号
enableSecurity	**False** 可禁用安全性，**True** 可启用安全性

将 Interact 配置为使用 jconsole 脚本进行 JMX 监视

如果没有单独的 JMX 监视应用程序，那么可使用随 JVM 安装的 jconsole。可以使用 Interact/tools 目录中的启动脚本来启动 jconsole。

关于此任务

缺省情况下，jconsole 脚本使用 JMXMP 协议进行监视。jconsole.bat 的缺省设置为：

JMXMP 连接

```
%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 连接

```
%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
```

过程

1. 在文本编辑器中打开 Interact\tools\jconsole.bat (Windows) 或 Interact/tools/jconsole.sh (UNIX)。
2. 将 INTERACT_LIB 设置为 *InteractInstallationDirectory*/lib 目录的完整路径。
3. 将 HOST 设置为要监视的运行时服务器的主机名。
4. 将 PORT 设置为您已通过 Interact > monitoring > port 属性为 JMX 配置的侦听端口。
5. 可选： 如果要使用 RMI 协议进行监视，请在 JMXMP 连接前面添加注释并除去 RMI 连接前面的注释。

JMX 属性

有多个属性可用于 JMX 监视。设计环境属性包括联系响应历史记录 ETL 监视。运行时环境属性包括异常、若干不同流程图属性、语言环境、记录器和线程池统计信息。还可使用若干服务统计信息属性。JMX 监视提供的所有数据都是自最近一次重置或系统启动之后的数据。例如，计数是自上一次重置或系统启动之后（而非安装之后）项的数量。

联系响应历史记录 ETL 监视器属性

联系响应历史记录 ETL 监视器属性是设计环境的一部分。所有以下属性都是运行时环境的一部分。

表 *9.* 联系响应历史记录 *ETL* 监视器

属性	描述
AvgCHExecutionTime	联系和响应历史记录模块写入到联系历史记录表所花费的平均毫秒数。仅对成功的操作以及至少有一条记录写入到联系历史记录表的操作计算此平均值。
AvgETLExecutionTime	联系和响应历史记录模块从运行时环境读取数据所花费的平均毫秒数。此平均值包括成功操作以及失败操作的时间。
AvgRHExecutionTime	联系和响应历史记录模块写入到响应历史记录表所花费的平均毫秒数。仅对成功的操作以及至少有一条记录写入到响应历史记录表的操作计算此平均值。
ErrorCount	自最近一次重置或系统启动之后已记录的错误消息的数目（如果有任何错误消息）。

表 *9.* 联系响应历史记录 *ETL* 监视器 *(续)*

属性	描述
HighWaterMarkCHExecutionTime	联系和响应历史记录模块写入到联系历史记录表所花费的最大毫秒数。仅对成功的操作以及至少有一条记录写入到联系历史记录表的操作计算此值。
HighWaterMarkETLExecutionTime	联系和响应历史记录模块从运行时环境读取数据所花费的最大毫秒数。此计算包含成功操作以及失败的操作。
HighWaterMarkRHExecutionTime	联系和响应历史记录模块写入到响应历史记录表所花费的最大毫秒数。仅对成功的操作以及至少有一条记录写入到响应历史记录表的操作计算此值。
LastExecutionDuration	联系和响应历史记录模块执行上一次复制所花费的毫秒数。
NumberOfExecutions	在初始化之后联系和响应历史记录模块已运行的次数。
LastExecutionStart	上一次运行的联系和响应历史记录模块的启动时间。
LastExecutionSuccessful	如果为 true，那么联系和响应历史记录模块的最近一次运行已成功。如果为 false，那么发生错误。
NumberOfContactHistoryRecordsMarked	在联系和响应历史记录模块的当前运行期间，UACI_CHStaging 表中正在移动的联系历史记录的数目。仅当联系和响应历史记录模块正在运行时，此值才大于零。
NumberOfResponseHistoryRecordsMarked	在联系和响应历史记录模块的当前运行期间，UACI_RHStaging 表中正在移动的响应历史记录的数目。仅当联系和响应历史记录模块正在运行时，此值才大于零。

异常属性

异常属性是运行时环境的一部分。

表 *10.* 异常

属性	描述
errorCount	自最近一次重置或系统启动之后已记录的错误消息的数目。
warningCount	自最近一次重置或系统启动之后已记录的警告消息的数目。

流程图引擎统计信息属性

流程图引擎统计信息属性是运行时环境的一部分。

表 *11.* 流程图引擎统计信息

属性	描述
activeProcessBoxThreads	当前正在运行流程图过程线程（在所有执行之间共享）的活动计数。
activeSchedulerThreads	当前正在运行的流程图调度程序线程的活动计数。
avgExecutionTimeMillis	流程图的平均执行时间（毫秒）。
CurrentJobsInProcessBoxQueue	等待由流程图过程线程来运行的作业数量。
CurrentJobsInSchedulerQueue	等待由流程图调度程序线程来运行的作业数量。
maximumProcessBoxThreads	可以运行流程图过程线程（在所有执行之间共享）的最大数量。
maximumSchedulerThreads	可以运行流程图调度程序线程（每个执行有一个线程）的最大数量。
numExecutionsCompleted	已完成的流程图执行的总数。
numExecutionsStarted	已启动的流程图执行的总数。

按交互式渠道属性划分的特定流程图

按交互式渠道属性划分的特定流程图是运行时环境的一部分。

表 *12.* 按交互式渠道划分的特定流程图

属性	描述
AvgExecutionTimeMillis	此流程图在此交互式渠道中的平均执行时间（毫秒）。
HighWaterMarkForExecutionTime	此流程图在此交互式渠道中的最长执行时间（以毫秒计）。
LastCompletedExecutionTimeMillis	在此交互式渠道中最近一次完成此流程图的执行时间（毫秒）。
NumExecutionsCompleted	在此交互式渠道中为此流程图完成的执行总数。
NumExecutionsStarted	在此交互式渠道中启动此流程图的总执行次数。

语言环境属性

语言环境属性是运行时环境的一部分。

表 *13.* 语言环境

属性	描述
locale	JMX 的语言环境设置。

记录器配置属性

记录器配置属性是运行时环境的一部分。

表 *14.* 记录器配置

属性	描述
类别	更改可在其中操纵日志级别的日志类别。

服务线程池统计信息属性

服务线程池统计信息属性是运行时环境的一部分。

表 *15.* 服务线程池统计信息

属性	描述
activeContactHistThreads	主动为联系历史记录和响应历史记录运行任务的大约线程数。
activeFlushCacheToDBThreads	主动运行任务以将高速缓存的统计信息清空到数据存储器的大约线程数。
activeOtherStatsThreads	主动为合格统计信息、事件活动和缺省统计信息运行任务的大约线程数。
CurrentHighWaterMarkInContactHistQueue	排队等待由服务（此服务收集联系和响应历史记录数据）进行记录的条目的最大数量。
CurrentHighWaterMark InFlushCachetoDBQueue	排队等待由服务（此服务将高速缓存中的数据写入到数据库表）进行记录的条目的最大数量。
CurrentHighWaterMarkInOtherStatsQueue	排队等待由服务（此服务收集商品合格统计信息、缺省字符串使用情况统计信息、事件活动统计信息以及定制记录到表数据）进行记录的条目的最大数量。
currentMsgsInContactHistQueue	用于联系历史记录和响应历史记录的线程池的队列中作业的数量。
currentMsgsInFlushCacheToDBQueue	用于将高速缓存的统计信息清空到数据存储器的线程池的队列中作业的数量。
currentMsgsInOtherStatsQueue	用于合格统计信息、事件活动和缺省统计信息的线程池的队列中作业的数量。
maximumContactHistThreads	在用于联系历史记录和响应历史记录的池中曾经同时存在的线程的最大数量。
maximumFlushCacheToDBThreads	在用于将高速缓存的统计信息清空到数据存储器的池中曾经同时存在的线程的最大数量。
maximumOtherStatsThreads	在用于合格统计信息、事件活动和缺省统计信息的池中曾经同时存在的线程的最大数量。

服务统计信息属性

服务统计信息针对每个服务都包含一组属性。

- 联系历史记录内存高速缓存统计信息 - 用于为联系历史记录登台表收集数据的服务。
- 定制记录器统计信息 - 用于收集定制数据以写入到表（用于 UACICustomLoggerTableName 事件参数的事件）的服务。
- 缺省统计信息 - 用于收集统计信息的服务，该统计信息与使用交互点的缺省字符串的次数有关。
- 合格统计信息 - 用于写入合格商品统计信息的服务。
- 事件活动统计信息 - 用于收集事件统计信息的服务，其中的事件包括系统事件（例如 getOffer 或 startSession）和 postEvent 触发的用户事件。
- 响应历史记录内存高速缓存统计信息 - 写入到响应历史记录登台表的服务。
- 跨会话响应统计信息 - 用于收集跨会话响应跟踪数据的服务。

表 *16.* 服务统计信息

属性	描述
计数	已处理消息的数量。
ExecTimeInsideMutex	处理此服务的消息所花的时间（以毫秒计），不包括等待其他线程所花的时间。如果 ExecTimeInsidMutex 与 ExecTimeMillis 之间差值较大，那么您可能需要更改此服务的线程池大小。
ExecTimeMillis	处理此服务的消息所花的时间（以毫秒计），包括等待其他线程所花的时间。
ExecTimeOfDBInsertOnly	仅处理批量插入部分所花费的时间长度（毫秒）。
HighWaterMark	为此服务处理的消息的最大数量。
NumberOfDBInserts	运行批量插入的总次数。
TotalRowsInserted	插入到数据库的总行数。

服务统计信息 - 数据库装入实用程序属性

服务统计信息 - 数据库装入实用程序属性是运行时环境的一部分。

表 *17.* 服务统计信息 - 数据库装入实用程序

属性	描述
ExecTimeOfWriteToCache	写入到文件高速缓存（包括写入到文件以及必要时从数据库中读取主键）所花费的时间长度（毫秒）。
ExecTimeOfLoaderDBAccessOnly	仅运行数据库装入器部分所花费的时间长度（毫秒）。
ExecTimeOfLoaderThreads	数据库装入器线程所花费的时间长度（毫秒）。
ExecTimeOfFlushCacheFiles	清除高速缓存并重新创建新的高速缓存所花的时间（以毫秒计）。

表 17. 服务统计信息 - 数据库装入实用程序（续）

属性	描述
ExecTimeOfRetrievePKDBAccess	检索主键数据库访问所花费的时间长度（毫秒）。
NumberOfDBLoaderRuns	数据库装入器运行的总数。
NumberOfLoaderStagingDirCreated	创建的登台目录的总数。
NumberOfLoaderStagingDirRemoved	移除的登台目录的总数。
NumberOfLoaderStaging DirMovedToAttention	重命名为 attention 的登台目录的总数。
NumberOfLoaderStagingDirMovedToError	重命名为 error 的登台目录的总数。
NumberOfLoaderStagingDirRecovered	已恢复的登台目录的总数，包括在启动时间和后台线程的重新运行。
NumberOfTimesRetrievePKFromDB	从数据库中检索主键的总次数。
NumberOfLoaderThreadsRuns	数据库装入器线程运行的总数。
NumberOfFlushCacheFiles	清除文件高速缓存的总次数。

API 统计信息属性

API 统计信息属性是运行时环境的一部分。

表 18. API 统计信息

属性	描述
endSessionCount	自最近一次重置或系统启动之后 endSession API 调用的数量。
endSessionDuration	最近一次调用 endSession API 所耗用的时间（以毫秒计）。
executeBatchCount	自最近一次重置或系统启动之后 executeBatch API 调用的数量。
executeBatchDuration	最近一次调用 executeBatch API 所耗用的时间（以毫秒计）。
getOffersCount	自最近一次重置或系统启动之后 getOffers API 调用的数量。
getOffersDuration	最近一次调用 getOffer API 所耗用的时间（以毫秒计）。
getProfileCount	自最近一次重置或系统启动之后 getProfile API 调用的数量。
getProfileDuration	最近一次调用 getProfileDuration API 所耗用的时间（以毫秒计）。
getVersionCount	自最近一次重置或系统启动之后 getVersion API 调用的数量。
getVersionDuration	最近一次调用 getVersion API 所耗用的时间（以毫秒计）。
loadOfferSuppressionDuration	最近一次调用 loadOfferSuppression API 所耗用的时间。
LoadOffersBySQLCount	自最近一次重置或系统启动之后 LoadOffersBySQL API 调用的数量。

表 *18. API* 统计信息 *(续)*

属性	描述
LoadOffersBySQLDuration	最近一次调用 LoadOffersBySQL API 所耗用的时间（以毫秒计）。
loadProfileDuration	最近一次调用 loadProfile API 所耗用的时间（以毫秒计）。
loadScoreOverrideDuration	最近一次调用 loadScoreOverride API 所耗用的时间（以毫秒计）。
postEventCount	自最近一次重置或系统启动之后 postEvent API 调用的数量。
postEventDuration	最近一次调用 postEvent API 所耗用的时间（以毫秒计）。
runSegmentationDuration	最近一次调用 runSegmentation API 所耗用的时间（以毫秒计）。
setAudienceCount	自最近一次重置或系统启动之后 setAudience API 调用的数量。
setAudienceDuration	最近一次调用 setAudience API 所耗用的时间（以毫秒计）。
setDebugCount	自最近一次重置或系统启动之后 setDebug API 调用的数量。
setDebugDuration	最近一次调用 setDebug API 所耗用的时间（以毫秒计）。
startSessionCount	自最近一次重置或系统启动之后 startSession API 调用的数量。
startSessionAverage	最近一次调用 startSession API 所耗用的平均时间（以毫秒计）。
ActiveSessionCount	Interact 运行时实例中当前处于活动状态的会话数。 **注：** JMX MBean com.unicacorp.interact:type=api, group=Statistics 中的 ActiveSessionCount 不会视为已超时的事件，因此，它会显示不正确的活动会话计数。

学习优化器统计信息属性

学习优化器统计信息属性是运行时环境的一部分。

表 *19.* 学习优化器统计信息

属性	描述
LearningOptimizerAcceptCalls	传递到学习模块的接受事件的数量。
LearningOptimizer AcceptTrackingDuration	在学习模块中记录接受事件所花费的总毫秒数。
LearningOptimizerContactCalls	传递到学习模块的联系事件的数量。
LearningOptimizer ContactTrackingDuration	在学习模块中记录联系事件所花费的总毫秒数。

表 19. 学习优化器统计信息 (续)

属性	描述
LearningOptimizerLogOtherCalls	传递到学习模块的非联系事件和非接受事件的数量。
LearningOptimizer LogOtherTrackingDuration	在学习模块中记录其他事件（非联系和非接受）所花费的持续时间（毫秒）。
LearningOptimizer NonRandomCalls	应用了已配置的学习实施的次数。
LearningOptimizer RandomCalls	绕过已配置的学习实施并应用了随机选择的次数。
LearningOptimizer RecommendCalls	传递到学习模块的建议请求的数量。
LearningOptimizer RecommendDuration	在学习建议逻辑中花费的总毫秒数。

缺省商品统计信息属性

缺省商品统计信息属性是运行时环境的一部分。

表 20. 缺省商品统计信息

属性	描述
LoadDefaultOffersDuration	装入缺省商品所耗用的时间。
DefaultOffersCalls	缺省商品装入的次数。

"触发式消息分派器"属性

"触发式消息分派器"属性是运行时环境的一部分。

表 21. 触发式消息分派器

属性	描述
NumRequested	这是请求使用此分派器进行分派的商品总数。
NumDispatched	这是此分派器已成功分派的商品总数。
AvgExecutionTime	这是此分派器分派商品所用的平均时间（以毫秒计）。在此计算中，仅计入已成功分派到网关的商品。
CurrentQueueSize	这是当前正在等待分派的商品的数目。
GatewayInvocation	这是此分派器分派到各个网关的商品数以及平均分派时间（以毫秒计）。此属性值的格式为 {gateway name=[number of offers, average dispatching time]}。

"触发式消息网关"属性

"触发式消息网关"属性是运行时环境的一部分。

表 22. 触发式消息网关

属性	描述
NumValidationRequested	这是此网关请求进行验证的商品总数。
NumValidated	这是此网关已成功验证的商品总数。

表 22. 触发式消息网关（续）

属性	描述
AvgValidationTime	这是此网关验证商品所用的平均时间（以毫秒计）。在此计算中，仅计入已成功验证的商品。
NumDeliveryRequested	这是此网关请求进行交付的商品总数。
NumDelivered	这是此网关已成功交付的商品总数。
AvgDeliveryTime	这是此网关交付商品所用的平均时间（以毫秒计）。在此计算中，仅计入已成功交付的商品。

"触发式消息消息"属性

"触发式消息消息"属性是运行时环境的一部分。

表 23. 触发式消息消息

属性	描述
ProcessSuccessCount	这是此触发式消息已成功执行的总次数。
AvgSuccessProcessTime	这是此触发式消息每次成功执行所用的平均时间（以毫秒计）。
ProcessErrorCount	这是此触发式消息未成功执行的总次数。
AvgErrorProcessTime	这是此触发式消息每次失败执行所用的平均时间（以毫秒计）。
SelectBranchCount	这是处理触发式消息时执行分支选择的总次数。
AvgSelectBranchTime	这是处理触发式消息时，执行分支选择所使用的平均时间（以毫秒计）。
SelectOfferCount	这是处理触发式消息时执行商品选择的总次数。
AvgSelectOfferTime	这是处理触发式消息时，执行商品选择所使用的平均时间（以毫秒计）。
SelectChannelCount	这是处理触发式消息时执行渠道选择的总次数。
AvgSelectChannelTime	这是处理触发式消息时，执行渠道选择所使用的平均时间（以毫秒计）。
FlowchartWaitCount	这是此触发式消息等待细分市场完成的总次数。
AvgFlowchartWaitTime	这是此触发式消息在每次执行中等待细分市场完成的平均时间（以毫秒计）。
WaitFlowchartTimeoutCount	这是此触发式消息在等待细分市场完成时发生超时的总次数。

JMX 操作

有若干操作可用于 JMX 监视。

下表描述了可用于 JMX 监视的操作。

组	属性	描述
记录器配置	activateDebug	将 Interact/conf/interact_log4j.properties 中定义的日志文件的记录级别设置为 debug。
记录器配置	activateError	将 Interact/conf/interact_log4j.properties 中定义的日志文件的记录级别设置为 error。
记录器配置	activateFatal	将 Interact/conf/interact_log4j.properties 中定义的日志文件的记录级别设置为 fatal。
记录器配置	activateInfo	将 Interact/conf/interact_log4j.properties 中定义的日志文件的记录级别设置为 info。
记录器配置	activateTrace	将 Interact/conf/interact_log4j.properties 中定义的日志文件的记录级别设置为 trace。
记录器配置	activateWarn	将 Interact/conf/interact_log4j.properties 中定义的日志文件的记录级别设置为 warn。
语言环境	changeLocale	更改 JMX 客户机的语言环境。Interact 支持的语言环境为 de、en、es 和 fr。
ContactResponseHistory ETLMonitor	reset	重置所有计数器。
缺省商品统计信息	updatePollPeriod	更新 defaultOfferUpdatePollPeriod。此值告知系统在更新高速缓存中的缺省商品之前要等待的时间长度（以秒为单位）。如果设置为 -1，那么系统仅在启动时读取缺省商品的数量。

第 7 章 IBM Interact Java、SOAP 和 REST API 的类和方法

以下各节列出在您使用 Interact API 之前应了解的需求和其他详细信息。

注：本节假定您熟悉接触点、Java 编程语言以及对基于 Java 的 API 的使用。

Interact API 具有一个 Java 客户机适配器，其使用基于 HTTP 的 Java 序列化。此外，Interact 将提供一个 WSDL 来支持 SOAP 客户机。WSDL 与 Java 客户机适配器展现一组相同功能，因此以下各节仍适用（示例除外）。

注：不支持单个 API 调用中任何参数多次出现。

Interact API 类

Interact API 基于 `InteractAPI` 类。

有 6 个支持接口。

- `AdvisoryMessage`
- `BatchResponse`
- `NameValuePair`
- 商品
- `OfferList`
- `Response`

这些接口具有 3 个支持的具体类。需要将以下两个具体类实例化，并作为参数传递到 Interact API 方法：

- NameValuePairImpl
- CommandImpl

第三个具体类名为 `AdvisoryMessageCode`，通过此类可提供某些常量以用于区分从服务器返回的消息代码（如果适用）。

本节的其余内容描述了构成 Interact API 的方法。

基于 HTTP 的 Java 序列化先决条件

Java 客户机适配器使用基于 HTTP 的 Java 序列化。

对基于 HTTP 的 Java 序列化使用 Java 客户机适配器的先决条件为：

1. 将以下文件添加到 `CLASSPATH`：

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

2. 在客户机与服务器之间来回传递的所有对象均可在软件包 `com.unicacorp.interact.api` 中找到。有关更多详细信息，请参阅安装在运行时服务器上 `Interact_Home/docs/apiJavaDoc` 中的 Interact API Javadoc。您可以通过使用任何 Web 浏览器打开该位置中的 `index.html` 文件来查看 Javadoc。

3. 要获取 InteractAPI 类的实例，请通过 Interact 运行时服务器的 URL 来调用静态方法 getInstance。

SOAP 先决条件

必须先执行若干先决条件任务以配置环境，然后才能使用 SOAP 来访问运行时服务器。

要点：性能测试表明，Java 序列化适配器的性能远高于生成的 SOAP 客户机的性能。为了获得最佳性能，请尽可能使用 Java 序列化适配器。

要使用 SOAP 来访问运行时服务器，必须执行以下操作：

1. 使用您选择的 SOAP 工具箱转换 Interact API WSDL。

   Interact API WSDL 随 Interact 一起安装，且安装在 Interact/conf 目录中。

   使用 WSDL XML 文件配置 SOAP 时，必须将 URL 修改为运行时服务器的主机名和端口。

   《Interact 管理指南》的末尾提供了 WSDL 的文本。
2. 安装并配置运行时服务器。

   运行时服务器必须正在运行才能完全测试您的集成。
3. 验证您是否在使用正确的 SOAP 版本。

   Interact 在 Interact 运行时服务器上使用 axis2 1.3 作为 SOAP 基础结构。有关 axis2 1.3 所支持的 SOAP 版本的详细信息，请访问以下 Web 站点：

   Apache Axis2

   Interact 已通过 axis2、XFire、JAX-WS-Ri、DotNet、SOAPUI 和 IBM RAD SOAP 客户机进行了测试。

REST 先决条件

调用 Interact API 的其中一种方法是通过 HTTP 使用 JSON（JavaScript 对象表示法）格式调用（此处称为 REST API）。REST API 的优势是比 SOAP 的性能更好，尽管 Java 序列化适配器仍是适用于 Interact API 调用的最快捷方法。

在开始使用 REST API 之前，请注意以下事项：

- 支持对 Interact API 的 REST 调用的 URL

  http://Interact_Runtime_Server:PORT/interact/servlet/RestServlet替代部署 Interact 的 Interact 运行时服务器和端口的实际主机名或 IP 地址。
- 存在两个特定于 REST API 的 Interact 类：RestClientConnector（作为帮助程序提供，以通过 JSON 格式的 REST 连接至 Interact 运行时实例）；RestFieldConstants（描述用于 API 请求和响应的 JSON 消息的底层格式）。
- 样本 REST 客户机在 Interact _Home/samples/javaApi/InteractRestClient.java 处提供。尽管样本代码为简单示例，但是它可以为演示如何使用 REST API 提供一个很好的起点。
- 要获取 REST API 类的完整描述以及所有其他 Interact API 信息，请参阅安装在运行时服务器上 Interact_Home/docs/apiJavaDoc 处的 Javadoc。

- REST API 以 HTML 转义格式而不是 Unicode 格式返回会话标识和消息。

除了此处提到的信息，REST API 还提供了有关使用 Interact API 的其他协议所支持的所有方法。

API JavaDoc

除了《Interact 管理员指南》之外，Interact API 的 Javadoc 也与运行时服务器一起安装。Javadoc 安装在 Interact_Home/docs/apiJavaDoc 目录中以供您参考。

API 示例

本指南中的所有示例都是使用基于 HTTP 适配器的 Java 序列化来创建的。通过 WSDL 生成的类可能会根据您选择的 SOAP 工具包和选项而有所不同。如果您在使用 SOAP，那么这些示例在您的环境中可能不会产生相同的效果。

处理会话数据

在您通过 startSession 方法启动会话时，会话数据将装入到内存中。在整个会话中，您可以读取和写入会话数据（这是静态概要文件数据的超集）。

此会话包含以下数据：

- 静态概要文件数据
- 细分市场分配
- 实时数据
- 商品推荐

在您调用 endSession 方法或者 sessionTimeout 时间经过之后，所有会话数据可用。一旦会话结束，未显式保存到联系或响应历史记录或某些其他数据库表的所有数据都将丢失。

数据将存储为一组名称/值对。如果数据是读取自数据库表，那么名称为表的列。

您可以在您使用 Interact API 时创建这些名称/值对。您不需要在全局区域中声明所有名称/值对。如果您将新的事件参数设置为名称/值对，那么运行时环境会将名称/值对添加到会话数据。例如，如果您将事件参数与 postEvent 方法配合使用，那么运行时环境会将事件参数添加到会话数据，即使事件参数在概要文件数据中不可用。此数据仅存在于会话数据中。

您可以随时覆盖会话数据。例如，如果客户概要文件的一部分包含 creditScore，那么您可以使用定制类型 NameValuePair 来传递事件参数。在 NameValuePair 类中，您可以使用 setName 和 setValueAsNumeric 方法来更改该值。名称需要匹配。在会话数据中，名称不区分大小写。因此，名称 creditscore 或 CrEdItScOrE 都将覆盖 creditScore。

仅保留写入到会话数据的最后数据。例如，startSession 将为 lastOffer 的值装入概要文件数据。postEvent 方法将覆盖 lastOffer。然后，另一个 postEvent 方法将覆盖 lastOffer。运行时环境仅在会话数据中保留由第二个 postEvent 方法写入的数据。

在会话结束时，数据将丢失，除非您有特别考虑（例如，在交互式流程图中使用快照进程以将数据写入到数据库表）。如果您计划使用快照进程，切记名称需要符合数据库的限制。例如，如果仅允许您为列的名称输入 256 个字符，那么名称/值对的名称不应超过 256 个字符。

关于 InteractAPI 类

InteractAPI 类包含可用于将您的接触点与运行时服务器进行集成的方法。Interact API 中的所有其他类和方法均支持此类中的方法。

您必须针对位于 Interact 运行时环境安装的 lib 目录中的 interact_client.jar 来编译您的实现。

endSession

endSession 方法标记运行时会话的结束。在运行时服务器收到此方法时，运行时服务器将记录到历史记录，清除内存等等。

endSession(String *sessionID*)

- **sessionID** - 用于标识会话的唯一字符串。

如果未调用 endSession 方法，那么运行时会话将超时。可以通过 sessionTimeout 属性来配置超时周期。

返回值

运行时服务器通过填充了以下属性的 Response 对象来响应 endSession 方法：

- SessionID
- ApiVersion
- StatusCode
- AdvisoryMessages

示例

以下示例显示了 endSession 方法以及您能够如何解析响应。sessionId 是相同字符串，用于标识启动此会话的 startSession 调用所使用的会话。

```
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 方法使您可以通过对运行时服务器的单一请求来执行多个方法。

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

- **sessionID** - 用于标识会话标识的字符串。此会话标识用于此方法调用所运行的所有命令。
- **commandImpl[]** - 一组 CommandImpl 对象，一个对象对应于您要执行的一个命令。

调用此方法的结果等效于显式调用 Command 数组中的每个方法。此方法可在最大程度上降低对运行时服务器的实际请求的数量。运行时服务器连续运行每个方法；对于每个调用，将在对应于此方法调用的 Response 对象中捕获任何错误或警告。如果遇到错误，那么 executeBatch 将继续运行批处理中的其余调用。如果运行任何方法导致了错误，那么 BatchResponse 对象的顶级状态将反映此错误。如果没有发生错误，那么顶级状态反映了可能发生的任何警告。如果未发生警告，那么顶级状态反映成功运行了批处理。

返回值

运行时服务器对应于包含 BatchResponse 对象的 executeBatch。

示例

以下示例表明如何通过单一 executeBatch 调用来调用所有 getOffer 和 postEvent 方法，以及有关如何处理响应的建议。

```
/** 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());      }
}
```

针对 Interact SOAP API 写入 executeBatch() XML 请求

使用这些步骤针对 Interact SOAP API 写入 executeBatch() XML 请求。

关于此任务

不得将单操作 SOAP API 调用（startSession、getOffers、setAudience、endSession 等）的请求 XML 直接复制或粘贴到多操作 executeBatch() 调用。executeBatch() 中的子命令与单操作 API 调用的子命令具有的 WSDL 和 XML 请求结构稍微有些不同。如果将 XML 元素从单操作 API 请求复制并粘贴到多操作 executeBatch 请求，那么结构化差异会导致服务器响应失败。

样本失败响应：

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

使用这些步骤写入 executeBatch() XML 请求。您可以在这些步骤期间引用参数值的单操作 API 调用请求，但不要复制和粘贴 XML 元素。

过程

1. 使用 WSDL 处理工具（例如，SoapUI）从 Interact WSDL 文件创建格式正确的 executeBatch() XML 请求。
2. 在针对 executeBatch() 子元素执行 WSDL 定义后，向请求添加子命令。
3. 在针对 executeBatch() 子元素执行 WSDL 定义后，填写子命令参数。

getInstance

getInstance 方法创建一个与指定运行时服务器进行通信的 Interact API 的实例。

```
getInstance(String URL)
```

要点：您使用 Interact API 编写的每个应用程序必须调用 getInstance 以实例化映射到 InteractAPI 对象（此对象映射到 URL 参数所指定的运行时服务器）。

对于服务器组，如果您在使用负载均衡器，请使用您通过负载均衡器配置的主机名和端口。如果您没有负载均衡器，那么您将必须包含用于在可用运行时服务器之间进行切换的逻辑。

此方法仅适用于基于 HTTP 适配器的 Java 序列化。SOAP WSDL 中未定义对应的方法。每个 SOAP 客户机实现都具有其自身的方法来建立端点 URL。

- **URL** - 用于标识运行时实例的 URL 的字符串。例如，`http://localhost:7001/Interact/servlet/InteractJSService`。

返回值

运行时服务器返回 InteractAPI。

示例

以下示例表明了如何实例化 InteractAPI 对象，此对象指向与您的接触点在同一个机器上运行的运行时服务器实例。

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

getOffers

`getOffers` 方法使您可以从运行时服务器中请求商品。

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

- **sessionID** - 用于标识当前会话的字符串。
- **interactionPoint** - 字符串，用于标识此方法引用的交互点的名称。

  注：此名称必须与交互式渠道中定义的交互点的名称完全匹配。
- **numberOfOffers** - 用于标识所请求商品数的整数。

`getOffers` 方法在运行之前，将等待 `segmentationMaxWaitTimeInMS` 属性中定义的毫秒数以使所有重新细分完成。因此，如果您调用可触发重新细分的 `postEvent` 方法，或者在 `getOffers` 调用的前一刻调用 `setAudience` 方法，那么可能会有延迟。

返回值

运行时服务器对应于 `getOffers`，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

示例

此示例显示了为"概述页面条幅 1"交互点请求单个商品以及处理响应的方法。

`sessionId` 是相同字符串，用于标识启动此会话的 `startSession` 调用所使用的运行时会话。

```
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 方法使您可以从运行时服务器中为具有重复删除的多个交互点请求商品。

getOffersForMultipleInteractionPoints(String *sessionID*, String *requestStr*)

- **sessionID** - 用于标识当前会话的字符串。
- **requestStr** - 提供一组 GetOfferRequest 对象的字符串。

  每个 GetOfferRequest 对象指定：
  - **ipName** - 对象正在为其请求商品的交互点 (IP) 名称
  - **numberRequested** - 指定 IP 需要的唯一商品的数量
  - **offerAttributes** - 使用 OfferAttributeRequirements 实例的已交付商品的属性的需求
  - **duplicationPolicy** - 将交付的商品的复制策略标识

    复制策略确定是否将在单个方法调用中跨不同交互点来返回重复的商品。（在单个交互点之内，从不返回重复商品。）目前支持两个复制策略。

- NO_DUPLICATION（标识值 = 1）。先前 GetOfferRequest 实例中已包含的商品均不包含在此 GetOfferRequest 实例中（即，Interact 将应用重复删除）。
- ALLOW_DUPLICATION（标识值 = 2）。满足此 GetOfferRequest 实例中指定的需求的任何商品都将包含在内。先前 GetOfferRequest 实例中已包含的商品将不会进行调整。

当交付商品时，数组参数中请求的顺序也是优先级顺序。

例如，假设请求中的 IP 为首先是 IP1，然后是 IP2，那么不允许重复商品（重复策略标识 = 1），并且每个 IP 请求两个商品。如果 Interact 为 IP1 发现商品 A、B 和 C，为 IP2 发现商品 A 和 D，那么响应将为 IP1 包含商品 A 和 B，为 IP2 仅包含商品 D。

另请注意，当复制策略标识为 1 时，通过优先级更高的 IP 交付的商品将不会通过此 IP 来交付。

getOffersForMultipleInteractionPoints 方法在运行之前，将等待 segmentationMaxWaitTimeInMS 属性中定义的毫秒数以使所有重新细分完成。因此，如果您调用可触发重新细分的 postEvent 方法，或者在 getOffers 调用的前一刻调用 setAudience 方法，那么可能会有延迟。

返回值

运行时服务器对应于 getOffersForMultipleInteractionPoints，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- 一组 OfferList
- SessionID
- StatusCode

示例

```
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());
 }
```

请注意，requestStr 的语法如下：

```
requests_for_IP[<requests_for_IP]
```

其中，

```
<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
```

在以上显示的示例中，requestForIP1 ({IP1,5,1,(5,attr1=1|numeric;attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))}) 表示，对于名为 IP1 的交互点，至多交付 5 个在此同一方法调用期间无法为任何其他交互点返回的不同商品。所有这 5 个商品都必须具有一个名为 attr1（值必须为 1）的数字属性，并且必须具有一个名为 attr2（值必须为 *value2*）的字符串属性。在这 5 个商品之外，至多 3 个必须具有一个名为 attr3（值必须为 *value3*）的字符串属性，至多 3 个必须具有名为 attr4（值必须为 *4*）的数字属性。

允许的属性类型为数字、字符串和日期时间，日期时间属性值的格式必须为 MM/dd/yyyyHH:mm:ss。要检索返回的商品，请使用方法 Response.getAllOfferLists()。为帮助了解语法，setGetOfferRequests 中的示例构建了 GetOfferRequests 的相同实例，但使用的是 Java 对象（这是首选）。

getProfile

getProfile 方法使您可以检索有关访问接触点的访问者的概要文件和临时信息。

getProfile(String *sessionID*)

- **sessionID** - 用于标识会话标识的字符串。

返回值

运行时服务器对应于 getProfile，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

示例

以下是 getProfile 的用法以及响应处理方法的示例。

sessionId 是相同字符串，用于标识启动此会话的 startSession 调用所使用的会话。

```
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 方法返回 Interact 运行时服务器的当前实现的版本。

```
getVersion()
```

最佳实践是在您通过 Interact API 初始化接触点时使用此方法。

返回值

运行时服务器对应于 getVersion，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- StatusCode

示例

此示例描述了一种用于调用 getVersion 并处理结果的简单方法。

```
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 方法使您可以执行交互式渠道中定义的任何事件。

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

- **sessionID**：用于标识会话标识的字符串。
- **eventName**：用于标识事件名称的字符串。

  注：事件的名称必须与交互式渠道中定义事件名称相匹配。此名称不区分大小写。
- **eventParameters**：NameValuePairImpl 对象标识需要与事件一起传递的任何参数。这些值存储在会话数据中。

  如果此事件触发重新细分，那么您必须确保交互式流程图需要的所有数据在会话数据中均可用。如果其中任何值尚未由先前操作填充（例如，从 startSession 或 setAudience，或者装入概要文件表），那么您必须为每个缺失值包含一个 eventParameter。例如，如果您已配置了所有要装入到内存的概要文件表，那么必须为交互式流程图所需的临时数据包含一个 NameValuePair。

  如果您在使用一个以上的受众级别，那么您很有可能针对每个受众级别具有不同组的 eventParameters。您应包含某些逻辑以确保您在为受众级别选择一组正确的参数。

  要点：如果此事件记录到响应历史记录，那么您必须为商品传递处理代码。您必须将 NameValuePair 的名称定义为"UACIOfferTrackingCode"。

  您只能为每个事件传递一个处理代码。如果不为商品联系传递处理代码，那么 Interact 将为上一个商品建议列表中的每个商品记录一个商品联系。如果不为响应传递处理代码，那么 Interact 将返回错误。
- 还有其他一些保留参数用于 postEvent 和其他方法，本节中后面的部分将进行讨论。

针对重新细分或写入到联系或响应历史记录的任何请求都不会等待响应。

重新细分不会清除当前受众级别的先前细分结果。您可使用 UACIExecuteFlowchartByName 参数定义要运行的特定流程图。getOffers 方法在运行之前将等待重新细分完成。因此，如果在 getOffers 调用的前一刻调用可触发重新细分的 postEvent 方法，那么可能会有延迟。

返回值

运行时服务器对应于 postEvent，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

示例

以下 postEvent 示例表明了为某个触发重新细分的事件发送新参数以及处理响应的方法。

sessionId 是相同字符串，用于标识启动此会话的 startSession 调用所使用的会话。

```
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 方法使您可以为访问者设置受众标识和级别。

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

- **sessionID** - 用于标识会话标识的字符串。
- **audienceID** - 用于定义受众标识的一组 NameValuePairImpl 对象。
- **audienceLevel** - 用于定义受众级别的字符串。
- **parameters** - NameValuePairImpl 对象，用于标识需要与 setAudience 一起传递的任何参数。这些值存储在会话数据中，并且可以用于细分市场。

  您必须为概要文件中的每个列提供一个值。这是为交互式渠道和任何实时数据定义的所有表中所有列的超集。如果您已使用 startSession 或 postEvent 填充了所有会话数据，那么不需要发送新参数。

setAudience 方法可触发重新细分。 getOffers 方法在运行之前将等待重新细分完成。因此，如果您在 getOffers 调用之前立即调用 setAudience 方法，那么可能会有延迟。

setAudience 方法还将装入受众标识的概要文件数据。您可以使用 setAudience 方法来强制重新装入由 startSession 方法装入的相同概要文件数据。

返回值

运行时服务器对应于 setAudience，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

示例

对于此示例，受众级别保持相同，但是标识发生更改，就如同匿名用户登录并且变为已知。

sessionId 和 audienceLevel 是相同字符串，用于标识启用此会话的 startSession 调用所使用的会话和受众级别。

```
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 方法使您可以为会话的所有代码路径设置记录详细级别。

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

- **sessionID** - 用于标识会话标识的字符串。
- **debug** - 用于启用或禁用调试信息的布尔值。有效值为 true 或 false。如果为 true，那么 Interact 会将调试信息记录到运行时服务器日志。

返回值

运行时服务器对应于 setDebug，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

示例

以下示例描述了更改会话的调试级别。

sessionId 是相同字符串，用于标识启动此会话的 startSession 调用所使用的会话。

```
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 方法创建并定义运行时会话。

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

startSession 可以至多触发五个操作：

- 创建运行时会话。
- 将当前受众级别的访问者概要文件数据装入到运行时会话，包括标记为装入到针对交互式渠道定义的表映射的任何维度表。
- 触发细分市场，从而为当前受众级别运行所有交互式流程图。
- 将商品禁止数据装入到会话（如果 enableOfferSuppressionLookup 属性设置为 true）。
- 将分数覆盖数据装入到会话（如果 enableScoreOverrideLookup 属性设置为 true）。

startSession 方法需要以下参数：

- **sessionID** - 用于标识会话标识的字符串。您必须定义会话标识。例如，您可以使用客户标识与时间戳记的组合。

  要定义构成运行时会话的内容，必须指定一个会话标识。此值由客户机管理。同一会话标识的所有方法调用都必须由客户机同步。具有相同会话标识的并发 API 调用的行为是未定义的。
- **relyOnExistingSession** - 一个布尔值，用于定义此方法是使用新会话还是现有会话。有效值为 true 或 false。如果为 true，那么您必须为 startSession 方法提供现有会话标识。如果为 false，那么必须提供新会话标识。

  如果将 relyOnExistingSession 设置为 true 并且存在某个会话，那么运行时环境将使用现有会话数据，并且不会重新装入任何数据，也不会触发细分市场划分。如果会话不存在，那么运行时环境将创建一个新会话，包括装入数据和触发细分市场划分。如果您的接触点的会话长度超过了运行时会话的长度，那么将 relyOnExistingSession 设置为 true 并且将其与所有 startSession 调用配合使用会很有用。例如，某个 Web 站点会员的生存时间是 2 小时，但是运行时会话的生存时间仅为 20 分钟。

如果您使用同一会话标识来调用 startSession 两次，那么在 relyOnExistingSession 为 false 的情况下，第一个 startSession 调用中的所有会话数据都将丢失。

- **debug** - 用于启用或禁用调试信息的布尔值。有效值为 true 或 false。如果为 true，那么 Interact 会将调试信息记录到运行时服务器日志。将为每个会话单独设置一个调试标志。因此，您可以跟踪某一个别会话的调试数据。
- **interactiveChannel** - 一个字符串，用于定义此会话引用的交互式渠道的名称。此名称必须与 Campaign 中定义的交互式渠道的名称精确匹配。
- **audienceID** - 一组 NameValuePairImpl 对象，其中的名称必须与包含受众标识的任何表的物理列名称匹配。
- **audienceLevel** - 用于定义受众级别的字符串。
- **parameters** - NameValuePairImpl 对象，用于标识需要与 startSession 一起传递的任何参数。这些值存储在会话数据中，并且可以用于细分市场。

  如果您对同一受众级别具有若干交互式流程图，那么必须包含所有表中所有列的超集。如果您配置运行时以装入概要文件表，并且概要文件表包含您需要的所有列，那么不需要传递任何参数，除非您希望覆盖概要文件表中的数据。如果您的概要文件表包含必需列的子集，那么必须作为参数来包含缺失的列。

如果 audienceID 或 audienceLevel 无效，并且 relyOnExistingSession 为 false，那么 startSession 调用将失败。如果 interactiveChannel 无效，那么 startSession 将失败，无论 relyOnExistingSession 为 true 还是 false。

如果 relyOnExistingSession 为 true，并且您使用同一 sessionID 来进行第二个 startSession 调用，但是第一个会话已过期，那么 Interact 将创建一个新会话。

如果 relyOnExistingSession 为 true，并且您使用相同 sessionID 但不同 audienceID 或 audienceLevel 来进行第二个 startSession 调用，那么运行时服务器将更改现有会话的受众。

如果 relyOnExistingSession 为 true，并且您使用相同 sessionID 但不同 interactiveChannel 来进行第二个 startSession 调用，那么运行时服务器将创建一个新会话。

返回值

运行时服务器对应于 startSession，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages（如果 StatusCode 不等于 0）
- ApiVersion
- SessionID
- StatusCode

示例

以下示例显示了一种调用 startSession 的方法。

```
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 是一个用于处理 startSession 返回的响应对象的方法。

```
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());    }
```

跨商品属性的商品取消复制

通过使用 Interact 应用程序编程接口 (API)，两个 API 调用会交付商品：getOffers 和 getOffersForMultipleInteractionPoints。getOffersForMultipleInteractionPoints 可以阻止在 *OfferID* 级别返回重复商品，但不能跨商品类别取消复制商品。因此，如果 Interact 要从每一个商品类别中只返回一件商品，先前需要一种变通方法。通过在 startSession API 调用中引入两个参数，则可能会在商品属性（例如，类别)中取消重复商品属性。

此列表汇总了已添加至 startSession API 调用的参数。有关这些参数或 Interact API 的任何方面的更多信息，请参阅《IBM Interact 管理员指南》或 <Interact_Home>/docs/apiJavaDoc 中 Interact 安装版本随附的 Javadoc 文件。

- UACIOfferDedupeAttribute。要使用商品取消复制创建 startSession API 调用，以便后续的 getOffer 调用只从每一种类别中返回一件商品，那么必须将 UACIOfferDedupeAttribute 参数包括为 API 调用的一部分。您可以采用 name,value,type 格式来指定参数，如下所示：

  UACIOfferDedupeAttribute,<attributeName>,string

  在本示例中，应将 *<attributeName>* 替换为您要用作取消复制条件的商品属性的名称，例如 Category。

  注：Interact 会检查具有您指定的相同属性值（例如 Category）的商品，并取消复制以移除所有商品（具有最高评分的商品除外）。如果具有重复属性的商品也具有相同评分，那么 Interact 会从匹配的商品中随机选择一个并返回。

- UACINoAttributeDedupeIfFewerOf。将 UACIOfferDedupeAttribute 包括在 startSession 调用中时，还可以设置此 UACINoAttributeDedupeIfFewerOf 参数以指定下列情况的行为：商品列表在取消复制之后不再包含足够的商品，无法满足原始请求。

  例如，如果设置 UACIOfferDedupeAttribute 以使用商品类别来取消复制商品，并且后续的 getOffers 调用请求返回 8 件商品，那么取消复制可能会导致合格的商品少于 8 件。在那种情况下，将 UACINoAttributeDedupeIfFewerOf 参数设置为 true 会导致将某些重复项添加到合格列表中以满足所请求的商品数。在本示例中，如果将该参数设置为 false，那么所返回的商品数将少于所请求的商品数。

  缺省情况下，UACINoAttributeDedupeIfFewerOf 设置为 true。

例如，假定您指定了取消复制条件为商品 Category 的 startSession 参数，如下所示：

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

这些参数一起导致 Interact 根据商品属性 Category 来取消复制商品，并只返回取消复制的商品，即使所产生的商品数小于所请求的商品数（UACINoAttributeDedupeIfFewerOffer 为 false）也是如此。

当您发出 getOffers API 调用时，原始的合格商品集可能包括以下商品：

- Category=Electronics：商品 A1 的评分为 100，且商品 A2 的评分为 50。
- Category=Smartphones：商品 B1 的评分为 100，商品 B2 的评分为 80，且商品 B3 的评分为 50。
- Category=MP3Players：商品 C1 的评分为 100，且商品 C2 的评分为 50。

在这种情况下，有两件重复商品与第一个类别相匹配，有三件重复商品与第二个类别相匹配，有两件重复商品与第三个类别相匹配。所返回的商品将是每一个类别中评分最高的商品，即商品 A1、商品 B1 和商品 C1。

如果 getOffers API 调用请求了 6 件商品，并且此示例将 UACINoAttributeDedupeIfFewerOffer 设置为 false，因此将只返回 3 件商品。

如果 getOffers API 调用请求了 6 件商品，并且此示例省略了 UACINoAttributeDedupeIfFewerOffer 参数或明确地将该参数设置为 true，那么将在结果中包括某些重复商品以满足所请求的数量。

保留参数

存在一些与 Interact API 结合使用的保留参数。部分保留参数是运行时服务器所必需的，另外的保留参数则可供您用于其他功能。

postEvent 功能

功能	参数	描述
记录到定制表	UACICustomLoggerTableName	运行时表数据源中表的名称。如果您提供的此参数包含有效表名，那么运行时环境会将所有会话数据写入到所选表。将填充表中与会话数据名称/值对相匹配的所有列名称。运行时环境将使用空值来填充与会话名称/值对不匹配的任何列。您可以通过 customLogger 配置属性来管理写入到数据库的过程。
多个响应类型	UACILogToLearning	值为 1 或 0 的整数。1 指示运行时环境应将事件记录为接受学习系统或在会话中启用商品抑制。0 指示运行时环境不应将事件记录到学习系统，也不应在会话中启用商品抑制。此参数使您可以创建若干 postEvent 方法，这些方法记录不同响应类型，而不影响学习。您不需要为设置为记录联系、接受或拒绝的事件定义此参数。必须将此参数与 UACIResponseTypeCode 结合使用。如果不定义 UACILOGTOLEARNING，那么运行时环境假定缺省值为 0（除非事件触发了记录联系、接受或拒绝）。
	UACIResponseTypeCode	表示响应类型代码的值。值必须是 UA_UsrResponseType 表中的有效条目

功能	参数	描述
响应跟踪	UACIOfferTrackingCode	商品的处理代码。如果事件记录到联系或响应历史记录，那么您必须定义此参数。您只能为每个事件传递一个处理代码。如果不为商品联系传递处理代码，那么运行时环境将为上一个商品建议列表中的每个商品记录一个商品联系。如果不为响应传递处理代码，那么运行时环境将返回错误。如果配置了跨会话响应跟踪，那么可使用 UACIOfferTrackingcodeType 参数来定义要使用的跟踪代码类型（处理代码除外）。
跨会话响应跟踪	UACIOfferTrackingCodeType	用于定义跟踪代码类型的数字。1 表示缺省处理代码，2 表示商品代码。所有代码都必须是 UACI_TrackingType 表中的有效条目。您可以向此表中添加其他定制代码。
特定流程图执行	UACIExecuteFlowchartByName	如果您为任何可触发细分市场的方法（触发重新细分的 startSession、setAudience 或 postEvent）定义此参数，那么 Interact 不是运行当前受众级别的所有流程图，而是仅运行指定的流程图。您可提供一个由管道字符（	）分隔的流程图列表。

运行时环境保留参数

以下保留参数由运行时环境使用。请勿将以下名称用于您的事件参数。

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

关于 AdvisoryMessage 类

advisoryMessage 类包含用于定义咨询消息对象的方法。咨询消息对象包含在 Response 对象中。InteractAPI 中的每个方法均返回一个 Response 对象。（executeBatch 方法除外，此方法返回一个 batchResponse 对象。）

如果有错误或警告，那么 Interact 服务器将填充咨询消息对象。咨询消息对象包含以下属性：

- **DetailMessage** - 咨询消息的详细描述。此属性并非对所有咨询消息都可用。如果可用，那么 DetailMessage 可能未本地化。
- **Message** - 咨询消息的简短描述。
- **MessageCode** - 咨询消息的代码编号。
- **StatusLevel** - 咨询消息严重性的代码编号。

您可以使用 getAdvisoryMessages 方法来检索 advisoryMessage 对象。

getDetailMessage

getDetailMessage 方法返回咨询消息对象的详细、完整描述。并非所有消息都具有详细消息。

```
getDetailMessage()
```

返回值

咨询消息对象返回一个字符串。

示例

```
// 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 方法返回咨询消息对象的简要描述。

```
getMessage()
```

返回值

咨询消息对象返回一个字符串。

示例

以下方法将打印出 AdvisoryMessage 对象的消息和详细消息。

```
// 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 方法在状态级别为 2 (STATUS_LEVEL_ERROR) 的情况下返回与咨询消息对象相关联的内部错误代码。

```
getMessageCode()
```

返回值

AdvisoryMessage 对象返回一个整数。

示例

以下方法可打印出 AdvisoryMessage 对象的消息代码。

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

getStatusLevel

getStatusLevel 方法返回咨询消息对象的状态级别。

```
getStatusLevel()
```

返回值

咨询消息对象返回一个整数。

- 0 - STATUS_LEVEL_SUCCESS - 调用的方法已完成，没有任何错误。
- 1 - STATUS_LEVEL_WARNING - 调用的方法已完成，且至少有一个警告（但没有错误）。
- 2 - STATUS_LEVEL_ERROR - 调用的方法未成功完成，并且至少有一个错误。

示例

以下方法可打印出 AdvisoryMessage 对象的状态级别。

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

关于 AdvisoryMessageCode 类

advisoryMessageCode 类包含用于定义咨询消息代码的方法。您将通过 getMessageCode 方法来检索咨询消息代码。

咨询消息代码

您将通过 getMessageCode 方法来检索咨询消息代码。

此表列示并描述了咨询消息代码。

代码	消息文本	描述
1	INVALID_SESSION_ID	会话标识未引用有效会话。
2	ERROR_TRYING_TO_ABORT_ SEGMENTATION	在 endSession 调用期间尝试终止细分市场划分时发生错误。
3	INVALID_INTERACTIVE_CHANNEL	传入的交互式渠道参数未引用有效的交互式渠道。
4	INVALID_EVENT_NAME	传入的事件参数未引用当前交互式渠道的有效事件。
5	INVALID_INTERACTION_POINT	传入的交互点参数未引用当前交互式渠道的有效交互点。

代码	消息文本	描述
6	ERROR_WHILE_MAKING_INITIAL_SEGMENTATION_REQUEST	提交细分市场划分请求时遇到错误。
7	SEGMENTATION_RUN_FAILED	细分市场划分已运行一部分，但导致了错误。
8	PROFILE_LOAD_FAILED	尝试装入概要文件表或维度表失败。
9	OFFER_SUPPRESSION_LOAD_FAILED	尝试装入商品禁止表失败。
10	COMMAND_METHOD_ UNRECOGNIZED	为 executeBatch 调用中的命令指定的命令方法无效。
11	ERROR_TRYING_TO_POST_EVENT_PARAMETERS	发布事件参数时发生错误。
12	LOG_SYSTEM_EVENT_EXCEPTION	尝试提交系统事件（结束会话、获取商品、获取概要文件、设置受众、设置调试或开始会话）以进行记录时发生异常。
13	LOG_USER_EVENT_EXCEPTION	尝试提交用户事件以进行记录时发生异常。
14	ERROR_TRYING_TO_LOOK_UP_EVENT	尝试查找事件名称时发生错误。
15	ERROR_TRYING_TO_LOOK_UP_INTERACTIVE_CHANNEL	尝试查找交互式渠道名称时发生错误。
16	ERROR_TRYING_TO_LOOK_UP_INTERACTION_POINT	尝试查找交互点名称时发生错误。
17	RUNTIME_EXCEPTION_ ENCOUNTERED	遇到了意外的运行时异常。
18	ERROR_TRYING_TO_EXECUTE_ASSOCIATED_ACTION	尝试运行相关联的操作（触发重新细分、记录商品联系、记录商品接受或记录商品拒绝）时发生错误。
19	ERROR_TRYING_RUN_FLOWCHART	尝试运行流程图时发生错误。
20	FLOWCHART_FAILED	流程图运行失败。
21	FLOWCHART_ABORTED	流程图运行已异常中止。
22	FLOWCHART_NEVER_RUN	指定的流程图从未运行。
23	FLOWCHART_STILL_RUNNING	流程图仍在运行。
24	ERROR_WHILE_READING_ PARAMETERS	读取参数时发生错误。
25	ERROR_WHILE_LOADING_RECOMMENDED_OFFERS	装入推荐的商品时出错
26	ERROR_WHILE_LOGGING_DEFAULT_TEXT_STATISTICS	记录缺省文本统计信息（交互点的缺省字符串的显示次数）时发生错误。
27	SCORE_OVERRIDE_LOAD_FAILED	未能装入分数覆盖表。
28	NULL_AUDIENCE_ID	受众标识为空。
29	UNRECOGNIZED_AUDIENCE_LEVEL	指定了无法识别的受众级别。
30	MISSING_AUDIENCE_FIELD	缺少受众字段。
31	INVALID_AUDIENCE_FIELD_TYPE	指定了无效受众字段类型。
32	UNSUPPORTED_AUDIENCE_FIELD_TYPE	受众字段类型不受支持

代码	消息文本	描述
33	TIMEOUT_REACHED_ON_GET_OFFERS_CALL	getOffers 调用已超时，没有返回商品。
34	INTERACT_INITIALIZATION_NOT_COMPLETED_SUCCESSFULLY	运行时服务器初始化未成功完成。
35	SESSION_ID_UNDEFINED	未定义会话标识。
36	INVALID_NUMBER_OF_OFFERS_REQUESTED	请求的商品数无效。
37	NO_SESSION_EXIST_BUT_WILL_CREATE_NEW_ONE	不存在任何会话，但是已创建一个会话。
38	AUDIENCE_ID_NOT_FOUND_IN_PROFILE_TABLE	指定的受众标识未在概要文件表中定义。
39	LOG_CUSTOM_LOGGER_EVENT_EXCEPTION	尝试提交定制记录数据事件时发生异常。
40	SPECIFIED_FLOWCHART_FOR_EXECUTION_DOES_NOT_EXIST	指定的流程图无法运行，因为它不存在。
41	AUDIENCE_NOT_DEFINED_IN_ CONFIGURATION	指定的受众未在配置中定义。

关于 BatchResponse 类

BatchResponse 类包含用于定义 executeBatch 方法的结果的方法。

BatchResponse 对象包含以下属性：

- **BatchStatusCode** - executeBatch 方法请求的所有响应的最高状态码值。
- **Responses** - executeBatch 方法请求的一组响应对象。

getBatchStatusCode

getBatchStatusCode 方法返回 executeBatch 方法执行的一组命令中的最高状态码。

```
getBatchStatusCode()
```

返回值

getBatchStatusCode 方法返回一个整数。

- 0 - STATUS_SUCCESS - 调用的方法已完成，没有任何错误。
- 1 - STATUS_WARNING - 调用的方法已完成，且至少有一个警告（但没有错误）。
- 2 - STATUS_ERROR - 调用的方法未成功完成，并且至少有一个错误。

示例

以下代码样本提供了如何检索 BatchStatusCode 的示例。

```
// 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 方法返回一组响应对象，这些响应对象对应于 executeBatch 方法执行的一组命令。

```
getResponses()
```

返回值

getResponses 方法返回一组 Response 对象。

示例

以下示例选择所有响应并在命令未成功的情况下打印出任何咨询消息。

```
for(Response response : batchResponse.getResponses())
{
    if(response.getStatusCode()!=Response.STATUS_SUCCESS)
    {
        printDetailMessageOfWarningOrError("executeBatchCommand",
      response.getAdvisoryMessages());        }
}
```

关于 Command 接口

executeBatch 方法要求您传递一组用于实现命令接口的对象。您应使用缺省实现 CommandImpl 来传递命令对象。

下表列出了命令、命令表示的 InteractAPI 类的方法以及您必须对每个命令使用的命令接口方法。不需要包含会话标识，因为 executeBatch 方法已包含会话标识。

命令	Interact API 方法	命令接口方法
COMMAND_ENDSESSION	endSession	无。
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	无。
COMMAND_GETVERSION	getVersion	无。
COMMAND_POSTEVENT	postEvent	• setEvent • setEventParameters

命令	Interact API 方法	命令接口方法
COMMAND_SETAUDIENCE	setAudience	• setAudienceID • setAudienceLevel • setEventParameters
COMMAND_SETDEBUG	setDebug	setDebug
COMMAND_STARTSESSION	startSession	• setAudienceID • setAudienceLevel • setDebug • setEventParameters • setInteractiveChannel • setRelyOnExistingSession

setAudienceID

setAudienceID 方法定义 setAudience 和 startSession 命令的受众标识。

setAudienceID(*audienceID*)

- **audienceID** - 用于定义受众标识的一组 NameValuePair 对象。

返回值

无。

示例

以下示例摘自用于调用 startSession 和 setAudience 的 executeBatch 方法。

```
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 方法定义 setAudience 和 startSession 命令的受众级别。

setAudienceLevel(*audienceLevel*)

- *audienceLevel* - 包含受众级别的字符串。

  要点：*audienceLevel* 的名称必须与 Campaign 中定义的受众级别的名称精确匹配。它是区分大小写的。

返回值

无。

示例

以下示例摘自用于调用 startSession 和 setAudience 的 executeBatch 方法。

```
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 方法定义 startSession 命令的提示级别。

```
setDebug(debug)
```

如果为 true，那么运行时服务器会将调试信息记录到运行时服务器日志。如果为 false，那么运行时服务器不会记录任何调试信息。将为每个会话单独设置一个调试标志。因此，您可以为个别运行时会话跟踪调试数据。

- **debug** - 布尔值（true 或 false）。

返回值

无。

示例

以下示例摘自用于调用 startSession 和 setDebug 的 executeBatch 方法。

```
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 方法定义了 postEvent 命令使用的事件的名称。

setEvent(*event*)

- **event** - 包含事件名称的字符串。

要点：*event* 的名称必须与交互式渠道中定义事件名称完全匹配。它是区分大小写的。

返回值

无。

示例

以下示例摘自调用 postEvent 的 executeBatch 方法。

```
String eventName = "SearchExecution";

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

setEventParameters

setEventParameters 方法定义 postEvent 命令使用的事件参数。这些值存储在会话数据中。

setEventParameters(*eventParameters*)

- **eventParameters** - 用于定义事件参数的一组 NameValuePair 对象。

例如，如果事件正在将商品记录到联系历史记录，那么您必须包含商品的处理代码。

返回值

无。

示例

以下示例摘自调用 postEvent 的 executeBatch 方法。

```
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 方法设置的参数用于检索 getOffersForMultipleInteractionPoints 命令所使用的商品。

setGetOfferRequests(*numberRequested*)

- **numberRequested** - 一组 GetOfferRequest 对象，这些对象定义用于检索商品的参数。

返回值

无。

示例

以下示例摘自调用 setGetOfferRequests 的 GetOfferRequest 方法。

```
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 方法定义了 startSession 命令使用的交互式渠道的名称。

setInteractiveChannel(*interactiveChannel*)

- **interactiveChannel** - 包含交互式渠道名称的字符串。

要点：*interactiveChannel* 必须与 Campaign 中定义的交互式渠道的名称精确匹配。它是区分大小写的。

返回值

无。

示例

以下示例摘自调用 startSession 的 executeBatch 方法。

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

setInteractionPoint

setInteractionPoint 方法定义 getOffers 和 postEvent 命令使用的交互点的名称。

setInteractionPoint(*interactionPoint*)

- **interactionPoint** - 包含交互点名称的字符串。

要点：*interactionPoint* 必须与交互式渠道中定义的交互点的名称精确匹配。它是区分大小写的。

返回值

无。

示例

以下示例摘自调用 getOffers 的 executeBatch 方法。

```
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 方法定义了命令对象中包含的命令类型。

setMethodIdentifier(*methodIdentifier*)

- **methodIdentifier** - 包含命令类型的字符串。

  有效值为：
  - **COMMAND_ENDSESSION** - 表示 endSession 方法。
  - **COMMAND_GETOFFERS** - 表示 getOffers 方法。
  - **COMMAND_GETPROFILE** - 表示 getProfile 方法。
  - **COMMAND_GETVERSION** - 表示 getVersion 方法。
  - **COMMAND_POSTEVENT** - 表示 postEvent 方法。
  - **COMMAND_SETAUDIENCE** - 表示 setAudience 方法。
  - **COMMAND_SETDEBUG** - 表示 setDebug 方法。
  - **COMMAND_STARTSESSION** - 表示 startSession 方法。

返回值

无。

示例

以下示例摘自用于调用 getVersion 和 endSession 的 executeBatch 方法。

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

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

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

setNumberRequested

setNumberRequested 方法定义 getOffers 命令请求的商品数量。

setNumberRequested(*numberRequested*)

- **numberRequested** - 用于定义 getOffers 命令请求的商品数量的整数。

返回值

无。

示例

以下示例摘自调用 getOffers 的 executeBatch 方法。

```
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 方法定义一个布尔值，此布尔值定义了 startSession 命令是否使用现有会话。

setRelyOnExistingSession(*relyOnExistingSession*)

如果为 true，那么 executeBatch 的会话标识必须与现有会话标识匹配。如果为 false，那么您必须为 executeBatch 方法提供一个新的会话标识。

- **relyOnExistingSession** - 布尔值（true 或 false）。

返回值

无。

示例

以下示例摘自调用 startSession 的 executeBatch 方法。

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

关于 NameValuePair 接口

Interact API 中的很多方法返回 NameValuePair 对象，或者要求您将 NameValuePair 对象作为参数来传递。作为参数传递到方法时，应使用缺省实现 NameValuePairImpl。

getName

getName 方法返回 NameValuePair 对象的名称构成项。

getName()

返回值

getName 方法返回一个字符串。

示例

以下示例摘自用于处理 getProfile 的响应对象的方法。

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

getValueAsDate

getValueAsDate 方法返回 NameValuePair 对象的值。

```
getValueAsDate()
```

在使用 getValueAsDate 之前，您应首先使用 getValueDataType 以确认正在引用正确的数据类型。

返回值

getValueAsDate 方法返回一个日期。

示例

以下示例摘自用于处理 NameValuePair 并打印值（如果值为日期）的方法。

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

getValueAsNumeric

getValueAsNumeric 方法返回 NameValuePair 对象的值。

```
getValueAsNumeric()
```

在使用 getValueAsNumeric 之前，您应首先使用 getValueDataType 以确认正在引用正确的数据类型。

返回值

getValueAsNumeric 方法返回一个双精度类型。例如，如果您在检索一个最初以整数格式存储在概要文件表中的值，那么getValueAsNumeric 将返回一个双精度类型。

示例

以下示例摘自用于处理 NameValuePair 并打印值（如果值为数字）的方法。

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

getValueAsString

getValueAsString 方法返回 NameValuePair 对象的值。

```
getValueAsString()
```

在使用 getValueDataType 之前，您应首先使用 getValueDataType 以确认正在引用正确的数据类型。

返回值

getValueAsString 方法返回一个字符串。例如，如果您在检索一个最初以 char、varchar 或 char[10] 格式存储在概要文件表中的值，那么 getValueAsString 将返回一个字符串。

示例

以下示例摘自用于处理 NameValuePair 并打印值（如果是字符串）的方法。

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

getValueDataType

getValueDataType 方法返回 NameValuePair 对象的数据类型。

```
getValueDataType()
```

在使用 getValueAsDate、getValueAsNumeric 或 getValueAsString 之前，您应首先使用 getValueDataType 以确认正在引用正确的数据类型。

返回值

getValueDataType 方法返回一个字符串，用于指示 NameValuePair 是包含数据、数字还是字符串。

有效值为：

- **DATA_TYPE_DATETIME** - 包含日期和时间值的日期。
- **DATA_TYPE_NUMERIC** - 包含数字值的双精度值。
- **DATA_TYPE_STRING** - 包含文本值的字符串。

示例

以下示例摘自用于处理 getProfile 方法中的响应对象的方法。

```
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 方法定义 NameValuePair 对象的名称构成项。

setName(*name*)

- **name** - 包含 NameValuePair 对象的名称构成项的字符串。

返回值

无。

示例

以下示例描述如何定义 NameValuePair 的名称构成项。

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

setValueAsDate

setValueAsDate 方法定义 NameValuePair 对象的值。

setValueAsDate(*valueAsDate*)

- **valueAsDate** - 包含 NameValuePair 对象的日期和时间值的日期。

返回值

无。

示例

以下示例描述在值为日期的情况下如何定义 NameValuePair 的值构成项。

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

setValueAsNumeric

setValueAsNumeric 方法定义 NameValuePair 对象的值。

setValueAsNumeric(*valueAsNumeric*)

- **valueAsNumeric** - 包含 NameValuePair 对象的数字值的双精度值。

返回值

无。

示例

以下示例描述在值为数字的情况下如何定义 NameValuePair 的值构成项。

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

setValueAsString

setValueAsString 方法定义 NameValuePair 对象的值。

setValueAsString(*valueAsString*)

- **valueAsString** - 包含 NameValuePair 对象的值的字符串

返回值

无。

示例

以下示例描述在值为数字的情况下如何定义 NameValuePair 的值构成项。

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

setValueDataType

setValueDataType 方法定义 NameValuePair 对象的数据类型。

getValueDataType(*valueDataType*)

有效值为：

- **DATA_TYPE_DATETIME** - 包含日期和时间值的日期。
- **DATA_TYPE_NUMERIC** - 包含数字值的双精度值。
- **DATA_TYPE_STRING** - 包含文本值的字符串。

返回值

无。

示例

以下示例描述如何设置 NameValuePair 的值的数据类型。

```
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);
```

关于 Offer 类

Offer 类包含用于定义 Offer 对象的方法。此 Offer 对象包含 Campaign 中商品的很多相同属性。

Offer 对象包含以下属性：

- **AdditionalAttributes** - 包含您已在 Campaign 中定义的任何定制商品属性的 NameValuePair。
- **Description** - 商品的描述。
- **EffectiveDate** - 商品的生效日期。
- **ExpirationDate** - 商品的到期日期。
- **OfferCode** - 商品的商品代码。
- **OfferName** - 商品的名称。
- **TreatmentCode** - 商品的处理代码。
- **Score** - 商品的市场营销分数，或者在 enableScoreOverrideLookup 属性为 true 的情况下，由 ScoreOverrideTable 定义的分数。

getAdditionalAttributes

getAdditionalAttributes 方法返回 Campaign 中定义的定制商品属性。

```
getAdditionalAttributes()
```

返回值

getAdditionalAttributes 方法返回一组 NameValuePair 对象。

示例

以下示例对所有其他属性进行排序，从而检查生效日期和截止日期，并打印出其他属性。

```
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 方法返回 Campaign 中定义的商品的描述。

```
getDescription()
```

返回值

getDescription 方法返回一个字符串。

示例

以下示例将打印商品的描述。

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

getOfferCode

getOfferCode 方法返回商品的商品代码（如 Campaign 中定义）。

```
getOfferCode()
```

返回值

getOfferCode 方法返回一组字符串，这些字符串包含商品的商品代码。

示例

以下示例将打印商品的商品代码。

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

getOfferName

getOfferName 方法返回商品的名称（如 Campaign 中定义）。

```
getOfferName()
```

返回值

getOfferName 方法返回一个字符串。

示例

以下示例打印商品的名称。

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

getScore

getScore 方法将返回分数，该分数基于您配置的商品。

```
getScore()
```

getScore 方法返回以下其中一个对象：

- 如果您未启用缺省商品表、分数覆盖表或内置学习，那么此方法将返回"交互策略"选项卡上定义的商品的市场营销分数。
- 如果您启用了缺省商品表或分数覆盖表，但未启用内置学习，那么此方法将返回商品（由缺省商品表之间的优先顺序来定义该商品）的分数、市场营销人员的分数以及分数覆盖表。
- 如果启用了内置学习，那么此方法将返回内置学习用于对商品进行排序的最终分数。

返回值

getScore 方法将返回一个表示商品分数的整数。

示例

以下示例将打印商品的分数。

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

getTreatmentCode

getTreatmentCode 方法返回商品的处理代码（如 Campaign 中定义）。

```
getTreatmentCode()
```

由于 Campaign 使用处理代码来标识所提供的商品的实例，因此在使用 postEvent 方法来记录商品的联系、接受或拒绝事件时，必须作为事件参数来返回此代码。如果在记录商品接受或拒绝，那么必须将表示处理代码的 NameValuePair 的名称值设置 UACIOfferTrackingCode。

返回值

getTreatmentCode 方法返回一个字符串。

示例

以下示例将打印商品的处理代码。

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

关于 OfferList 类

OfferList 类包含用于定义 getOffers 方法的结果的方法。

OfferList 对象包含以下属性：

- **DefaultString** - 为交互式渠道中的交互点定义的缺省字符串。
- **RecommendedOffers** - getOffers 方法所请求的一组 Offer 对象。

OfferList 类可以处理商品的列表。此类与 Campaign 商品列表无关。

getDefaultString

getDefaultString 方法返回交互点的缺省字符串（如 Campaign 中所定义）。

```
getDefaultString()
```

如果 RecommendedOffers 对象为空，那么您应将您的接触点配置为呈现此字符串，以确保将呈现某些内容。仅当 RecommendedOffers 对象为空时，Interact 才会填充 DefaultString 对象。

返回值

getDefaultString 方法返回一个字符串。

示例

以下示例在 offerList 对象不包含任何商品的情况下获取缺省字符串。

```
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 方法返回由 getOffers 方法请求的一组 Offer 对象。

```
getRecommendedOffers()
```

如果 getRecommendedOffer 的响应为空，那么接触点应呈现 getDefaultString 的结果。

返回值

getRecommendedOffers 方法返回一个 Offer 对象。

示例

以下示例处理 OfferList 对象，并打印所有推荐商品的商品名称。

```
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());
```

关于 Response 类

Response 类包含用于定义任何 InteractAPI 类方法的结果的方法。

Response 对象包含以下属性：

- **AdvisoryMessages** - 一组咨询消息。仅当方法运行中出现警告或错误时，才会填充此属性。
- **ApiVersion** - 包含 API 版本的字符串。此属性由 getVersion 方法来填充。
- **OfferList** - 包含 getOffers 方法请求的商品的 OfferList 对象。
- **ProfileRecord** - 包含概要文件数据的一组 NameValuePair。此属性由 getProfile 方法来填充。
- **SessionID** - 用于定义会话标识的字符串。这由所有 InteractAPI 类方法返回。
- **StatusCode** - 一个数字，指示方法的运行是无错误、有错误还是有警告。这由所有 InteractAPI 类方法返回。

getAdvisoryMessages

getAdvisoryMessages 方法从 Response 对象返回一组咨询消息。

```
getAdvisoryMessages()
```

返回值

getAdvisoryMessages 方法返回一组咨询消息对象。

示例

以下示例从某一 Response 对象获取咨询消息对象并对其进行迭代，从而打印消息。

```
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 方法返回 Response 对象的 API 版本。

```
getApiVersion()
```

getVersion 方法填充 Response 对象的 ApiVersion 属性。

返回值

Response 对象返回一个字符串。

示例

以下示例摘自用于处理 getVersion 的响应对象的方法。

```
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 方法返回 Response 对象的 OfferList 对象。

```
getOfferList()
```

getOffers 方法填充 Response 对象的 OfferList 对象。

返回值

Response 对象返回 OfferList 对象。

示例

以下示例摘自用于处理 getOffers 的响应对象的方法。

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

getAllOfferLists

getAllOfferLists 方法返回响应对象一组所有 OfferList。

```
getAllOfferLists()
```

这由 getOffersForMultipleInteractionPoints 方法使用，此方法填充了 Response 对象的 OfferList 数组对象。

返回值

Response 对象返回 OfferList 数组。

示例

以下示例摘自用于处理 getOffers 的响应对象的方法。

```
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 方法作为一组 NameValuePair 对象来返回当前会话的概要文件记录。这些概要文件记录还包括之前在运行时会话中添加的任何 eventParameters。

```
getProfileRecord()
```

getProfile 方法填充 Response 对象的概要文件 NameValuePair 对象。

返回值

Response 对象返回一组 NameValuePair 对象。

示例

以下示例摘自用于处理 getOffers 的响应对象的方法。

```
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 方法返回会话标识。

```
getSessionID()
```

返回值

getSessionID 方法返回一个字符串。

示例

以下示例描述了您可以在错误处理的开头或结尾显示的消息，用于指示与任何错误相关的会话。

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

getStatusCode

getStatusCode 方法返回 Response 对象的状态码。

```
getStatusCode()
```

返回值

Response 对象返回一个整数。

- 0 - STATUS_SUCCESS - 调用的方法已完成，没有任何错误。可能有也可能没有咨询消息。
- 1 - STATUS_WARNING - 调用的方法已完成，且至少有一个警告消息（但没有错误）。请查询咨询消息以获取更多详细信息。
- 2 - STATUS_ERROR - 调用的方法未成功完成，并且至少有一个错误消息。请查询咨询消息以获取更多详细信息。

示例

以下是您能够如何在错误处理中使用 getStatusCode 的示例。

```
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 IBM Interact JavaScript API 的类和方法

以下各部分列出在您使用 Interact JavaScript API 之前应了解的需求和其他详细信息。

Interact API 支持 javascript flavor，以允许最终用户客户机（浏览器）与服务器通信。

注：本部分假设您熟悉基于 JavaScript 的 API。

注：不支持单个 API 调用中任何参数多次出现。

JavaScript 先决条件

在 Web 站点上使用 Interact JavaScript API 之前，必须在 Web 页面上包含 `interactapi.js` 文件。

处理会话数据

在您通过 `startSession` 方法启动会话时，会话数据将装入到内存中。在整个会话中，您可以读取和写入会话数据（这是静态概要文件数据的超集）。

此会话包含以下数据：

- 静态概要文件数据
- 细分市场分配
- 实时数据
- 商品推荐

在您调用 `endSession` 方法或者 `sessionTimeout` 时间经过之后，所有会话数据可用。一旦会话结束，未显式保存到联系或响应历史记录或某些其他数据库表的所有数据都将丢失。

数据将存储为一组名称/值对。如果数据是读取自数据库表，那么名称为表的列。

您可以在您使用 Interact API 时创建这些名称/值对。您不需要在全局区域中声明所有名称/值对。如果您将新的事件参数设置为名称/值对，那么运行时环境会将名称/值对添加到会话数据。例如，如果您将事件参数与 `postEvent` 方法配合使用，那么运行时环境会将事件参数添加到会话数据，即使事件参数在概要文件数据中不可用。此数据仅存在于会话数据中。

您可以随时覆盖会话数据。例如，如果客户概要文件的一部分包含 `creditScore`，那么您可以使用定制类型 `NameValuePair` 来传递事件参数。在 `NameValuePair` 类中，您可以使用 `setName` 和 `setValueAsNumeric` 方法来更改该值。名称需要匹配。在会话数据中，名称不区分大小写。因此，名称 `creditscore` 或 `CrEdItScOrE` 都将覆盖 `creditScore`。

仅保留写入到会话数据的最后数据。例如，`startSession` 将为 `lastOffer` 的值装入概要文件数据。`postEvent` 方法将覆盖 `lastOffer`。然后，另一个 `postEvent` 方法将覆盖 `lastOffer`。运行时环境仅在会话数据中保留由第二个 `postEvent` 方法写入的数据。

在会话结束时，数据将丢失，除非您有特别考虑（例如，在交互式流程图中使用快照进程以将数据写入到数据库表）。如果您计划使用快照进程，切记名称需要符合数据库的限制。例如，如果仅允许您为列的名称输入 256 个字符，那么名称/值对的名称不应超过 256 个字符。

使用回调参数

回调函数是 Interact JavaScript API 的每个方法的额外参数。

主要浏览进程为单线程事件循环。在单线程事件循环中执行长时间运行的操作会阻塞此进程。在等待完成操作期间，此进程无法处理其他时间。为了阻止长时间运行的操作出现阻塞，XMLHttpRequest 提供了异步接口。向此接口传递回调以在操作完成后运行，此接口处理时，会向主要事件循环返回控制而不是阻塞此进程。

如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

例如，如果希望在 Web 页面上显示商品，那么将使用 getOffers 方法并使用回调以在页面上显示。Web 页面正常运行，不会等待 Interact 返回商品。而是在 Interact 返回商品时，在回调参数中往回发送响应。您可以解析回调数据并在页面上显示商品。

您可以将一个通用回调用于所有函数，还可以将特定回调用于特定函数。

您可以使用 `var callback = InteractAPI.Callback.create(onSuccess, onError);` 创建通用回调函数。

您可以使用以下函数为 getOffers 方法创建特定回调函数。

```
var callbackforGetOffer = InteractAPI.Callback.create(onSuccessofGetOffer,
onErrorofGetOffer);
```

关于 InteractAPI 类

InteractAPI 类包含可用于将您的接触点与运行时服务器进行集成的方法。Interact API 中的所有其他类和方法均支持此类中的方法。

您必须针对位于 Interact 运行时环境安装的 lib 目录中的 interact_client.jar 来编译您的实现。

startSession

startSession 方法创建并定义运行时会话。

```
function callStartSession(commandsToExecute, callback) {

        //read configured start session
        var ssId = document.getElementById('ss_sessionId').value;
        var icName = document.getElementById('ic').value;
        var audId = document.getElementById('audienceId').value;
        var audLevel = document.getElementById('audienceLevel').value;
        var params = document.getElementById('ss_parameters').value;
        var relyOldSs = document.getElementById('relyOnOldSession').value;
        var debug = document.getElementById('ss_isDebug').value;

        InteractAPI.startSession(ssId, icName,
                                    getNameValuePairs(audId), audLevel,
```

```
                                    getNameValuePairs(params), relyOldSs,
                                    debug, callback) ;
}
```

startSession 可以至多触发五个操作：

- 创建运行时会话。
- 将当前受众级别的访问者概要文件数据装入到运行时会话，包括标记为装入到针对交互式渠道定义的表映射的任何维度表。
- 触发细分市场，从而为当前受众级别运行所有交互式流程图。
- 将商品禁止数据装入到会话（如果 enableOfferSuppressionLookup 属性设置为 true）。
- 将分数覆盖数据装入到会话（如果 enableScoreOverrideLookup 属性设置为 true）。

startSession 方法需要以下参数：

- **sessionID** - 用于标识会话标识的字符串。您必须定义会话标识。例如，您可以使用客户标识与时间戳记的组合。

  要定义构成运行时会话的内容，必须指定一个会话标识。此值由客户机管理。同一会话标识的所有方法调用都必须由客户机同步。具有相同会话标识的并发 API 调用的行为是未定义的。
- **relyOnExistingSession** - 一个布尔值，用于定义此方法是使用新会话还是现有会话。有效值为 true 或 false。如果为 true，那么您必须为 startSession 方法提供现有会话标识。如果为 false，那么必须提供新会话标识。

  如果将 relyOnExistingSession 设置为 true 并且存在某个会话，那么运行时环境将使用现有会话数据，并且不会重新装入任何数据，也不会触发细分市场划分。如果会话不存在，那么运行时环境将创建一个新会话，包括装入数据和触发细分市场划分。如果您的接触点的会话长度超过了运行时会话的长度，那么将 relyOnExistingSession 设置为 true 并且将其与所有 startSession 调用配合使用会很有用。例如，某个 Web 站点会员的生存时间是 2 小时，但是运行时会话的生存时间仅为 20 分钟。

  如果您使用同一会话标识来调用 startSession 两次，那么在 relyOnExistingSession 为 false 的情况下，第一个 startSession 调用中的所有会话数据都将丢失。
- **debug** - 用于启用或禁用调试信息的布尔值。有效值为 true 或 false。如果为 true，那么 Interact 会将调试信息记录到运行时服务器日志。将为每个会话单独设置一个调试标志。因此，您可以跟踪某一个别会话的调试数据。
- **interactiveChannel** - 一个字符串，用于定义此会话引用的交互式渠道的名称。此名称必须与 Campaign 中定义的交互式渠道的名称精确匹配。
- **audienceID** - 一组 NameValuePairImpl 对象，其中的名称必须与包含受众标识的任何表的物理列名称匹配。
- **audienceLevel** - 用于定义受众级别的字符串。
- **parameters** - NameValuePairImpl 对象，用于标识需要与 startSession 一起传递的任何参数。这些值存储在会话数据中，并且可以用于细分市场。

  如果您对同一受众级别具有若干交互式流程图，那么必须包含所有表中所有列的超集。如果您配置运行时以装入概要文件表，并且概要文件表包含您需要的所有列，

那么不需要传递任何参数，除非您希望覆盖概要文件表中的数据。如果您的概要文件表包含必需列的子集，那么必须作为参数来包含缺失的列。

- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

如果 audienceID 或 audienceLevel 无效，并且 relyOnExistingSession 为 false，那么 startSession 调用将失败。如果 interactiveChannel 无效，那么 startSession 将失败，无论 relyOnExistingSession 为 true 还是 false。

如果 relyOnExistingSession 为 true，并且您使用同一 sessionID 来进行第二个 startSession 调用，但是第一个会话已过期，那么 Interact 将创建一个新会话。

如果 relyOnExistingSession 为 true，并且您使用相同 sessionID 但不同 audienceID 或 audienceLevel 来进行第二个 startSession 调用，那么运行时服务器将更改现有会话的受众。

如果 relyOnExistingSession 为 true，并且您使用相同 sessionID 但不同 interactiveChannel 来进行第二个 startSession 调用，那么运行时服务器将创建一个新会话。

返回值

运行时服务器对应于 startSession，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages（如果 StatusCode 不等于 0）
- ApiVersion
- SessionID
- StatusCode

跨商品属性的商品取消复制

通过使用 Interact 应用程序编程接口 (API)，两个 API 调用会交付商品：getOffers 和 getOffersForMultipleInteractionPoints。getOffersForMultipleInteractionPoints 可以阻止在 *OfferID* 级别返回重复商品，但不能跨商品类别取消复制商品。因此，如果 Interact 要从每一个商品类别中只返回一件商品，先前需要一种变通方法。通过在 startSession API 调用中引入两个参数，则可能会在商品属性（例如，类别)中取消重复商品属性。

此列表汇总了已添加至 startSession API 调用的参数。有关这些参数或 Interact API 的任何方面的更多信息，请参阅《IBM Interact 管理员指南》或 <Interact_Home>/docs/apiJavaDoc 中 Interact 安装版本随附的 Javadoc 文件。

-

  UACIOfferDedupeAttribute。要使用商品取消复制创建 startSession API 调用，以便后续的 getOffer 调用只从每一种类别中返回一件商品，那么必须将 UACIOfferDedupeAttribute 参数包括为 API 调用的一部分。您可以采用 name,value,type 格式来指定参数，如下所示：

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

  在本示例中，应将 *<attributeName>* 替换为您要用作取消复制条件的商品属性的名称，例如 Category。

注：Interact 会检查具有您指定的相同属性值（例如 Category）的商品，并取消复制以移除所有商品（具有最高评分的商品除外）。如果具有重复属性的商品也具有相同评分，那么 Interact 会从匹配的商品中随机选择一个并返回。

- UACINoAttributeDedupeIfFewerOffer。将 UACIOfferDedupeAttribute 包括在 startSession 调用中时，还可以设置此 UACINoAttributeDedupeIfFewerOffer 参数以指定下列情况中的行为：商品列表在消除重复项之后不再包含足够的商品，无法满足原始请求。

  例如，如果设置 UACIOfferDedupeAttribute 以使用商品类别来取消复制商品，并且后续的 getOffers 调用请求返回 8 件商品，那么取消复制可能会导致合格的商品少于 8 件。在此情况下，将 UACINoAttributeDedupeIfFewerOffer 参数设置为 true 会导致将某些重复项添加到合格列表中以满足所请求的商品数。在本示例中，如果将该参数设置为 false，那么所返回的商品数将少于所请求的商品数。

  缺省情况下，UACINoAttributeDedupeIfFewerOffer 设置为 true。

例如，假定您指定了取消复制条件为商品 Category 的 startSession 参数，如下所示：

```
UACIOfferDedupeAttribute,Category,string;

UACINoAttributeDedupeIfFewerOffer,1,string
```

缺省情况下，如果返回的数目少于所请求的数目，那么 UACIOfferDedupeAttribute 将不会消除重复商品。但是，要确保在返回的数目少于所请求的数目时消除重复项，必须提供 UACINoAttributeDedupeIfFewerOffer 参数并将其设置为 1。

这些参数一起导致 Interact 根据商品属性 Category 来取消复制商品，并只返回取消复制的商品，即使所产生的商品数小于所请求的商品数（UACINoAttributeDedupeIfFewerOffer 为 false）也是如此。

当您发出 getOffers API 调用时，原始的合格商品集可能包括以下商品：

- Category=Electronics：商品 A1 的评分为 100，且商品 A2 的评分为 50。
- Category=Smartphones：商品 B1 的评分为 100，商品 B2 的评分为 80，且商品 B3 的评分为 50。
- Category=MP3Players：商品 C1 的评分为 100，且商品 C2 的评分为 50。

在这种情况下，有两件重复商品与第一个类别相匹配，有三件重复商品与第二个类别相匹配，有两件重复商品与第三个类别相匹配。所返回的商品将是每一个类别中评分最高的商品，即商品 A1、商品 B1 和商品 C1。

如果 getOffers API 调用请求了 6 件商品，并且此示例将 UACINoAttributeDedupeIfFewerOffer 设置为 false，因此将只返回 3 件商品。

如果 getOffers API 调用请求了 6 件商品，并且此示例省略了 UACINoAttributeDedupeIfFewerOffer 参数或明确地将该参数设置为 true，那么将在结果中包括某些重复商品以满足所请求的数量。

postEvent

postEvent 方法使您可以执行交互式渠道中定义的任何事件。

```
function callPostEvent(commandsToExecute, callback) {

        var ssId = document.getElementById('pe_sessionId').value;
        var ev = document.getElementById('event').value;
        var params = document.getElementById('parameters').value;

        InteractAPI.postEvent(ssId, ev, getNameValuePairs(params), callback);

}
```

- **sessionID**：用于标识会话标识的字符串。
- **eventName**：用于标识事件名称的字符串。

  注：事件的名称必须与交互式渠道中定义事件名称相匹配。此名称不区分大小写。
- **eventParameters**：NameValuePairImpl 对象标识需要与事件一起传递的任何参数。这些值存储在会话数据中。

  如果此事件触发重新细分，那么您必须确保交互式流程图需要的所有数据在会话数据中均可用。如果其中任何值尚未由先前操作填充（例如，从 startSession 或 setAudience，或者装入概要文件表），那么您必须为每个缺失值包含一个 eventParameter。例如，如果您已配置了所有要装入到内存的概要文件表，那么必须为交互式流程图所需的临时数据包含一个 NameValuePair。

  如果您在使用一个以上的受众级别，那么您很有可能针对每个受众级别具有不同组的 eventParameters。您应包含某些逻辑以确保您在为受众级别选择一组正确的参数。

  要点：如果此事件记录到响应历史记录，那么您必须为商品传递处理代码。您必须将 NameValuePair 的名称定义为"UACIOfferTrackingCode"。

  您只能为每个事件传递一个处理代码。如果不为商品联系传递处理代码，那么 Interact 将为上一个商品建议列表中的每个商品记录一个商品联系。如果不为响应传递处理代码，那么 Interact 将返回错误。
- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。
- 还有其他一些保留参数用于 postEvent 和其他方法，本节中后面的部分将进行讨论。

针对重新细分或写入到联系或响应历史记录的任何请求都不会等待响应。

重新细分不会清除当前受众级别的先前细分结果。您可使用 UACIExecuteFlowchartByName 参数定义要运行的特定流程图。getOffers 方法在运行之前将等待重新细分完成。因此，如果在 getOffers 调用的前一刻调用可触发重新细分的 postEvent 方法，那么可能会有延迟。

返回值

运行时服务器对应于 postEvent，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- OfferList
- 概要分析
- SessionID
- StatusCode

getOffers

getOffers 方法使您可以从运行时服务器中请求商品。

```
function callGetOffers(commandsToExecute, callback) {

        var ssId = document.getElementById('go_sessionId').value;
        var ip = document.getElementById('go_ipoint').value;
        var nofRequested = 5 ;
        var nreqString = document.getElementById('offersRequested').value;

        InteractAPI.getOffers(ssId, ip, nofRequested, callback);

}
```

- **session ID** - 用于标识当前会话的字符串。
- **Interaction point** - 字符串，用于标识此方法引用的交互点的名称。

  注：此名称必须与交互式渠道中定义的交互点的名称完全匹配。
- **nofRequested** - 用于标识所请求商品数的整数。
- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

getOffers 方法在运行之前，将等待 segmentationMaxWaitTimeInMS 属性中定义的毫秒数以使所有重新细分完成。因此，如果您调用可触发重新细分的 postEvent 方法，或者在 getOffers 调用的前一刻调用 setAudience 方法，那么可能会有延迟。

返回值

运行时服务器对应于 getOffers，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- OfferList
- 概要分析
- SessionID
- StatusCode

getOffersForMultipleInteractionPoints

getOffersForMultipleInteractionPoints 方法使您可以从运行时服务器中为具有重复删除的多个交互点请求商品。

```
function callGetOffersForMultipleInteractionPoints(commandsToExecute, callback) {

        var ssId = document.getElementById('gop_sessionId').value;
        var requestDetailsStr = document.getElementById('requestDetail').value;

        //trim string
        var trimmed = requestDetailsStr.replace(/\{/g, "");
        var parts = trimmed.split("}");

        //sanitize strings
        for(i = 0; i < parts.length; i += 1) {
            parts[i] = parts[i].replace(/^\s+|\s+$/g, "");
        }

        //build get offer requests
        var getOffReqs = [];
```

```
        for(var i = 0; i < parts.length; i += 1) {
            var getofReqObj = parseGetOfferReq(parts[i]);
            if (getofReqObj) {
                getOffReqs.push(getofReqObj);

        InteractAPI.getOffersForMultipleInteractionPoints
        (ssId, getOffReqs, callback);
}
```

- **session ID** - 用于标识当前会话的字符串。
- **requestDetailsStr** - 提供一组 GetOfferRequest 对象的字符串。

  每个 GetOfferRequest 对象指定：

  – **ipName** - 对象正在为其请求商品的交互点 (IP) 名称

  – **numberRequested** - 指定 IP 需要的唯一商品的数量

  – **offerAttributes** - 使用 OfferAttributeRequirements 实例的已交付商品的属性的需求

  – **duplicationPolicy** - 将交付的商品的复制策略标识

    复制策略确定是否将在单个方法调用中跨不同交互点来返回重复的商品。（在单个交互点之内，从不返回重复商品。）目前支持两个复制策略。

    - NO_DUPLICATION（标识值 = 1）。先前 GetOfferRequest 实例中已包含的商品均不包含在此 GetOfferRequest 实例中（即，Interact 将应用重复删除）。
    - ALLOW_DUPLICATION（标识值 = 2）。满足此 GetOfferRequest 实例中指定的需求的任何商品都将包含在内。先前 GetOfferRequest 实例中已包含的商品将不会进行调整。

  – **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

  当交付商品时，数组参数中请求的顺序也是优先级顺序。

  例如，假设请求中的 IP 为首先是 IP1，然后是 IP2，那么不允许重复商品（重复策略标识 = 1），并且每个 IP 请求两个商品。如果 Interact 为 IP1 发现商品 A、B 和 C，为 IP2 发现商品 A 和 D，那么响应将为 IP1 包含商品 A 和 B，为 IP2 仅包含商品 D。

  另请注意，当复制策略标识为 1 时，通过优先级更高的 IP 交付的商品将不会通过此 IP 来交付。

getOffersForMultipleInteractionPoints 方法在运行之前，将等待 segmentationMaxWaitTimeInMS 属性中定义的毫秒数以使所有重新细分完成。因此，如果您调用可触发重新细分的 postEvent 方法，或者在 getOffers 调用的前一刻调用 setAudience 方法，那么可能会有延迟。

返回值

运行时服务器对应于 getOffersForMultipleInteractionPoints，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages

- ApiVersion
- 一组 OfferList
- 概要分析
- SessionID
- StatusCode

setAudience

setAudience 方法使您可以为访问者设置受众标识和级别。

```
function callSetAudience(commandsToExecute, callback) {

        var ssId = document.getElementById('sa_sessionId').value;
        var audId = document.getElementById('sa_audienceId').value;
        var audLevel = document.getElementById('sa_audienceLevel').value;
        var params = document.getElementById('sa_parameters').value;

        InteractAPI.setAudience(ssId, getNameValuePairs(audId),audLevel,
                                    getNameValuePairs(params), callback);

}
```

- **sessionID** - 用于标识会话标识的字符串。
- **audienceID** - 用于定义受众标识的一组 NameValuePairImpl 对象。
- **audienceLevel** - 用于定义受众级别的字符串。
- **parameters** - NameValuePairImpl 对象，用于标识需要与 setAudience 一起传递的任何参数。这些值存储在会话数据中，并且可以用于细分市场。

  您必须为概要文件中的每个列提供一个值。这是为交互式渠道和任何实时数据定义的所有表中所有列的超集。如果您已使用 startSession 或 postEvent 填充了所有会话数据，那么不需要发送新参数。
- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

setAudience 方法可触发重新细分。 getOffers 方法在运行之前将等待重新细分完成。因此，如果您在 getOffers 调用之前立即调用 setAudience 方法，那么可能会有延迟。

setAudience 方法还将装入受众标识的概要文件数据。您可以使用 setAudience 方法来强制重新装入由 startSession 方法装入的相同概要文件数据。

setAudience 方法在现有会话中重新装入白名单表和黑名单表。您可以使用带有 UACIPurgePriorWhiteListOnLoad 和 UACIPurgePriorBlackListOnLoad 参数的 setAudience 方法在现有会话中重新装入白名单表和黑名单表。

缺省情况下，调用 setAudience 方法时，将移除黑名单的所有内容。您可以按如下所示在 setAudience 调用中设置 UACIPurgePriorWhiteListOnLoad 和 UACIPurgePriorBlackListOnLoad 参数：

- 如果设置 UACIPurgePriorBlackListOnLoad= 0>，那么将保留白名单表的所有内容。
- 如果设置 UACIPurgePriorWhiteListOnLoad= 1>，那么将移除该表的内容，并且将从数据库中装入受众标识的白名单或黑名单的内容。完成后，将启动重新细分市场的过程。

返回值

运行时服务器对应于 setAudience，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- OfferList
- 概要分析
- SessionID
- StatusCode

getProfile

getProfile 方法使您可以检索有关访问接触点的访问者的概要文件和临时信息。

```
function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        InteractAPI.getProfile(ssId, callback);

}
```

- **session ID** - 用于标识会话标识的字符串。
- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

返回值

运行时服务器对应于 getProfile，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- OfferList
- ProfileRecord
- SessionID
- StatusCode

endSession

endSession 方法标记运行时会话的结束。在运行时服务器收到此方法时，运行时服务器将记录到历史记录，清除内存等等。

```
function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        InteractAPI.endSession(ssId, callback);

}
```

- **session ID** - 用于标识会话的唯一字符串。
- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

如果未调用 endSession 方法，那么运行时会话将超时。可以通过 sessionTimeout 属性来配置超时周期。

返回值

运行时服务器通过填充了以下属性的 Response 对象来响应 endSession 方法：

- SessionID
- ApiVersion
- OfferList
- 概要分析
- StatusCode
- AdvisoryMessages

setDebug

setDebug 方法使您可以为会话的所有代码路径设置记录详细级别。

```
function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        InteractAPI.setDebug(ssId, isDebug, callback);

}
```

- **sessionID** - 用于标识会话标识的字符串。
- **debug** - 用于启用或禁用调试信息的布尔值。有效值为 true 或 false。如果为 true，那么 Interact 会将调试信息记录到运行时服务器日志。
- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

返回值

运行时服务器对应于 setDebug，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- OfferList
- 概要分析
- SessionID
- StatusCode

getVersion

getVersion 方法返回 Interact 运行时服务器的当前实现的版本。

```
function callGetVersion(commandsToExecute, callback) {

        InteractAPI.getVersion(callback);


}
```

最佳实践是在您通过 Interact API 初始化接触点时使用此方法。

- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

返回值

运行时服务器对应于 getVersion，后者包含填充了以下属性的 Response 对象：

- AdvisoryMessages
- ApiVersion
- OfferList
- 概要分析
- SessionID
- StatusCode

executeBatch

executeBatch 方法使您可以通过对运行时服务器的单一请求来执行多个方法。

```
function callExecuteBatch(commandsToExecute, callback) {

        if (!commandsToExecute)
            return ;

        InteractAPI.executeBatch(commandsToExecute.ssid,
        commandsToExecute.commands, callback);
}
```

- **session ID** - 用于标识会话标识的字符串。此会话标识用于此方法调用所运行的所有命令。
- **commands** - 一组命令对象，一个对象对应于您要执行的一个命令。
- **callback** - 如果此方法成功，那么回调函数会调用 onSuccess。如果此方法失败，那么回调函数会调用 onError。

调用此方法的结果等效于显式调用 Command 数组中的每个方法。此方法可在最大程度上降低对运行时服务器的实际请求的数量。运行时服务器连续运行每个方法；对于每个调用，将在对应于此方法调用的 Response 对象中捕获任何错误或警告。如果遇到错误，那么 executeBatch 将继续运行批处理中的其余调用。如果运行任何方法导致了错误，那么 BatchResponse 对象的顶级状态将反映此错误。如果没有发生错误，那么顶级状态反映了可能发生的任何警告。如果未发生警告，那么顶级状态反映成功运行了批处理。

返回值

运行时服务器对应于包含 BatchResponse 对象的 executeBatch。

JavaScript API 示例

```
function isJavaScriptAPISelected() {
    var radios = document.getElementsByName('api');
    for (var i = 0, length = radios.length; i < length; i++) {
        if (radios[i].checked) {
            if (radios[i].value === 'JavaScript')
                  return true;
            else  // only one radio can be logically checked
                break;
```

```
        }
    }
    return false;    }

function processFormForJSInvocation(e) {

    if (!isJavaScriptAPISelected())
        return;

    if (e.preventDefault) e.preventDefault();

    var serverurl = document.getElementById('serviceUrl').value ;
    InteractAPI.init( { "url" : serverurl } );

    var commandsToExecute = { "ssid" : null, "commands" : [] };
    var callback = InteractAPI.Callback.create(onSuccess, onError);

    callStartSession(commandsToExecute, callback);
    callGetOffers(commandsToExecute, callback);
    callGetOffersForMultipleInteractionPoints(commandsToExecute, callback);
    callPostEvent(commandsToExecute, callback);
    callSetAudience(commandsToExecute, callback);
    callGetProfile(commandsToExecute, callback);
    callEndSession(commandsToExecute, callback);
    callSetDebug(commandsToExecute, callback);
    callGetVersion(commandsToExecute, callback);

    callExecuteBatch(commandsToExecute, callback);

    // You must return false to prevent the default form behavior
    return false;    }

function callStartSession(commandsToExecute, callback) {

    //read configured start session
    var ssId = document.getElementById('ss_sessionId').value;
    var icName = document.getElementById('ic').value;
    var audId = document.getElementById('audienceId').value;
    var audLevel = document.getElementById('audienceLevel').value;
    var params = document.getElementById('ss_parameters').value;
    var relyOldSs = document.getElementById('relyOnOldSession').value;
    var debug = document.getElementById('ss_isDebug').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                    createStartSessionCmd(
                                    icName, getNameValuePairs(audId),
                                    audLevel, getNameValuePairs(params),
                                    relyOldSs, debug));
    }
    else {
        InteractAPI.startSession(ssId, icName,
                                getNameValuePairs(audId), audLevel,
                                getNameValuePairs(params), relyOldSs,
                                debug, callback) ;
    }

}

function callGetOffers(commandsToExecute, callback) {

    var ssId = document.getElementById('go_sessionId').value;
    var ip = document.getElementById('go_ipoint').value;
```

```
    var nofRequested = 5 ;
    var nreqString = document.getElementById('offersRequested').value;
    if (!nreqString && nreqString!== '')
        nofRequested = Number(nreqString);

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                    createGetOffersCmd(ip, nofRequested));
    }
    else {
        InteractAPI.getOffers(ssId, ip, nofRequested, callback);
    }
}

function callPostEvent(commandsToExecute, callback) {

    var ssId = document.getElementById('pe_sessionId').value;
    var ev = document.getElementById('event').value;
    var params = document.getElementById('parameters').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.
                                    CommandUtil.createPostEventCmd
                                    (ev, getNameValuePairs(params)));
    }
    else {
        InteractAPI.postEvent(ssId, ev, getNameValuePairs(params), callback);
    }
}

function callGetOffersForMultipleInteractionPoints
(commandsToExecute, callback) {

    var ssId = document.getElementById('gop_sessionId').value;
    var requestDetailsStr = document.getElementById('requestDetail').value;

    //trim string
    var trimmed = requestDetailsStr.replace(/\{/g, "");
    var parts = trimmed.split("}");

    //sanitize strings
    for(i = 0; i < parts.length; i += 1) {
        parts[i] = parts[i].replace(/^\s+|\s+$/g, "");
    }

    //build get offer requests
    var getOffReqs = [];
    for(var i = 0; i < parts.length; i += 1) {
        var getofReqObj = parseGetOfferReq(parts[i]);
        if (getofReqObj) {
            getOffReqs.push(getofReqObj);
        }
    }

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
```

```
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                    createGetOffersForMultiple
                                    InteractionPointsCmd(getOffReqs));
    }
    else {
        InteractAPI.getOffersForMultipleInteractionPoints
                                    (ssId, getOffReqs, callback);
    }
}

function parseGetOfferReq(ofReqStr) {

    if (!ofReqStr || ofReqStr==="")
             return null;

    var posIp = ofReqStr.indexOf(',');
    var ip = ofReqStr.substring(0,posIp);
    var posNmReq = ofReqStr.indexOf(',', posIp+1);
    var numReq = ofReqStr.substring(posIp+1,posNmReq);
    var posDup = ofReqStr.indexOf(',', posNmReq+1);
    var dupPolicy = null;
    var reqAttributes = null;

    if (posDup===-1)
        dupPolicy = ofReqStr.substring(posNmReq+1);
    else
        dupPolicy = ofReqStr.substring(posNmReq+1,posDup);

    //check if request string has attributes
    var reqAttrPos = ofReqStr.search(/\(/g);
    if (reqAttrPos!==-1) {
        var reqAttributesStr = ofReqStr.substring(reqAttrPos);
        reqAttributesStr = trimString(reqAttributesStr);
        reqAttributesStr = removeOpenCloseBrackets(reqAttributesStr);
        reqAttributes = parseReqAttributes(reqAttributesStr);
    }

   return InteractAPI.GetOfferRequest.create(ip, parseInt(numReq),
                                parseInt(dupPolicy), reqAttributes);
}

//trim string
function trimString(strToTrim) {
    if (strToTrim)
        return strToTrim.replace(/^\s+|\s+$/g, "");
    else
             return null;
}

function trimStrArray(strArray) {
    if (!strArray) return ;
    for(var i = 0; i < strArray.length; i += 1) {
        strArray[i] = trimString(strArray[i]);
    }
}

//remove open and close brackets in the end
function removeOpenCloseBrackets(strToUpdate) {
    if (strToUpdate)
        return strToUpdate.replace(/^\(+|\)+$/g, "");
    else
             return null;
}

function parseReqAttributes(ofReqAttrStr) {

    //sanitize string
```

```
ofReqAttrStr = trimString(ofReqAttrStr);
ofReqAttrStr = removeOpenCloseBrackets(ofReqAttrStr);

if (!ofReqAttrStr || ofReqAttrStr==="")
         return null;

//get the number requested
var pos = ofReqAttrStr.indexOf(",");
var numRequested = ofReqAttrStr.substring(0,pos);
ofReqAttrStr = ofReqAttrStr.substring(pos+1);

//first part will be attribute and rest will be child attributes
var parts = [];
pos = ofReqAttrStr.indexOf(",");
if (pos!==-1) {
    parts.push(ofReqAttrStr.substring(0,pos));
    parts.push(ofReqAttrStr.substring(pos+1));
}
else {
    parts.push(ofReqAttrStr);
}

for(var i = 0; i < parts.length; i += 1) {
    //sanitize string
    parts[i] = trimString(parts[i]);
    parts[i] = removeOpenCloseBrackets(parts[i]);
    parts[i] = trimString(parts[i]);
}

//build list of attributes
var attributes = [];
var idx = 0;
if (parts[0]) {
    var attParts = parts[0].split(";");
    for (idx=0; idx<attParts.length; idx++) {
        attParts[idx] = trimString(attParts[idx]);
        attParts[idx] = removeOpenCloseBrackets(attParts[idx]);
        attParts[idx] = trimString(attParts[idx]);

        var atrObj = parseAttribute(attParts[idx]);
        if (atrObj) attributes.push(atrObj);
    }
}

//build list of child attributes
var childAttributes = [];
if (parts[1]) {
    var childAttParts = parts[1].split(")");
    for (idx=0; idx<childAttParts.length; idx++) {

        childAttParts[idx] = trimString(childAttParts[idx]);
        childAttParts[idx] = removeOpenCloseBrackets(childAttParts[idx]);
        childAttParts[idx] = trimString(childAttParts[idx]);


        //get the number requested
        var noReqPos = childAttParts[idx].indexOf(",");
        var numReqAt = childAttParts[idx].substring(0,noReqPos);
        childAttParts[idx] = childAttParts[idx].substring(noReqPos+1);
        childAttParts[idx] = trimString(childAttParts[idx]);


        var atrObjParsed = parseAttribute(childAttParts[idx]);
        if (atrObjParsed) {
            var childReq = InteractAPI.OfferAttributeRequirements.create
            (parseInt(numReqAt), [atrObjParsed], null);
            childAttributes.push(childReq);
```

```
                }
            }
        }

        return InteractAPI.OfferAttributeRequirements.create(parseInt(numRequested),
        attributes, childAttributes);
    }

    function parseAttribute(attStr) {

        attStr = trimString(attStr);

        if (!attStr || attStr==="")
                return null;

        var pos1 = attStr.indexOf("=");
        var pos2 = attStr.indexOf("|");
        var nvp = InteractAPI.NameValuePair.create
                            ( attStr.substring(0,pos1),
                              attStr.substring(pos1+1, pos2),
                              attStr.substring(pos2+1));
        return nvp;
    }
function callSetAudience(commandsToExecute, callback) {
        if (!document.getElementById('checkSetAudience').checked)
            return ;

        var ssId = document.getElementById('sa_sessionId').value;
        var audId = document.getElementById('sa_audienceId').value;
        var audLevel = document.getElementById('sa_audienceLevel').value;
        var params = document.getElementById('sa_parameters').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                            createSetAudienceCmd
            (getNameValuePairs(audId), audLevel, getNameValuePairs(params)));
        }
        else {
            InteractAPI.setAudience(ssId, getNameValuePairs(audId),
                                    audLevel, getNameValuePairs(params),
                                    callback);
        }
    }

    function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createGetProfileCmd());
        }
        else {
            InteractAPI.getProfile(ssId, callback);
        }
    }

    function callEndSession(commandsToExecute, callback) {
```

```
    var ssId = document.getElementById('es_sessionId').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createEndSessionCmd());
    }
    else {
        InteractAPI.endSession(ssId, callback);
    }
}

function callSetDebug(commandsToExecute, callback) {

    var ssId = document.getElementById('sd_sessionId').value;
    var isDebug = document.getElementById('isDebug').value;

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createSetDebugCmd(isDebug));
    }
    else {
        InteractAPI.setDebug(ssId, isDebug, callback);
    }
}

function callGetVersion(commandsToExecute, callback) {

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createGetVersionCmd());
    }
    else {
        InteractAPI.getVersion(callback);
    }
}

function callExecuteBatch(commandsToExecute, callback) {

    if (!commandsToExecute)
        return ;

    InteractAPI.executeBatch(commandsToExecute.ssid,
    commandsToExecute.commands, callback);
}

function getNameValuePairs(parameters) {

    if (parameters === '')
          return null;

    var parts = parameters.split(';');
    var nvpArray = new Array(parts.length);

    for(i = 0; i < parts.length; i += 1) {
        var nvp = parts[i].split(',') ;
        var value = null;
        if (nvp[2]===InteractAPI.NameValuePair.prototype.TypeEnum.NUMERIC) {
            if (isNaN(nvp[1])) {
                value = nvp[1];  //a non number was provided as number,
```

```
                pass it to API as it is
            }
            else {
                value = Number(nvp[1]);
            }
        }
        else {
            value = nvp[1];
        }
        //special handling NULL value
        if (value && typeof value === 'string') {
            if (value.toUpperCase() === 'NULL') {
                value = null;
            }
        }
        nvpArray[i] = InteractAPI.NameValuePair.create(nvp[0], value, nvp[2]) ;
    }

    return nvpArray;
}

function showResponse(textDisplay) {
    var newWin = open('','Response','height=300,width=300,titlebar=no,
    scrollbars=yes,toolbar=no,
   resizable=yes,menubar=no,location=no,status=no');

    if (newWin.locationbar !== 'undefined' && newWin.locationbar
    && newWin.locationbar.visible)
        newWin.locationbar.visible = false;

    var displayHTML = '<META HTTP-EQUIV="Content-Type"
    CONTENT="text/html; charset=UTF-8">
    <html><head><style>TD { border-width : thin; border-style : solid }</style.'
                        + "<script language='Javascript'>"
                        + "var desiredDomain = 'unicacorp.com'; "
                        + "if (location.href.indexOf(desiredDomain)>=0) "
                        + "{ document.domain = desiredDomain;} "
                        + "</script></head><body> "
                        + textDisplay
                        + "</body></html>" ;
    newWin.document.body.innerHTML = displayHTML;
    newWin.focus() ;
}

function onSuccess(response) {
    showResponse("********Response********<br> " + JSON.stringify(response)) ;
}

function onError(response) {
    showResponse("********Error********<br> " + response) ;
}

function formatResoponse(response) {

}

function printBatchResponse(batResponse) {

}

function printResponse(response) {

}
```

示例响应 JavaScript 对象 onSuccesss

此示例显示响应 JavaScript 对象的三个变量：offerLists、messages 和 profile。

如果将 getOffer 或 getOffersForMultipleInteractionPoints 作为 API 或批处理命令的一部分调用，那么 offerList 会返回非空列表。您应该针对此情况始终检查是否为空，然后再对此变量执行操作。

您应该始终检查 messages JavaScript 响应的状态。

如果将 getProfile 用作 API 或批处理命令的一部分，那么 Profile 会返回非空。如果未使用 getProfile，那么可忽略此变量。您应该针对此情况始终检查是否为空，然后再对此变量执行操作。

```
function onSuccess(response)
InteractAPI.ResponseTransUtil._buildResponse = function(response) {
        'use strict';

        if (!response) return null;

        var offerList = null;
        //transform offerLists to JS Objects
        if (response.offerLists) {
            offerList = [];
            for (var ofListCt=0; ofListCt<response.offerLists.length;ofListCt++) {
                var ofListObj = this._buildOfferList(response.offerLists[ofListCt]);
                if (ofListObj) offerList.push(ofListObj);
            }
        }

        var messages = null;
        //transform messages to JS Objects
        if (response.messages) {
            messages = [];
            for (var msgCt=0; msgCt<response.messages.length;msgCt++) {
                var msgObj = this._buildAdvisoryMessage(response.messages[msgCt]);
                if (msgObj) messages.push(msgObj);
            }
        }

        var profile = null;
        //transform profile nvps to JS Objects
        if (response.profile) {
            profile = [];
            for (var nvpCt=0; nvpCt<response.profile.length;nvpCt++) {
                var nvpObj = this._buildNameValuePair(response.profile[nvpCt]);
                if (nvpObj) profile.push(nvpObj);
            }
        }

        return InteractAPI.Response.create(response.sessionId,
                                           response.statusCode, offerList,
                                           profile, response.version,
                                           messages) ;
    };
```

第 9 章 关于 ExternalCallout API

Interact 提供一个可扩展宏 EXTERNALCALLOUT，以用于您的交互式流程图。此宏使您可以执行定制逻辑以在流程图运行期间与外部系统进行通信。例如，如果您希望在流程图运行期间计算某一客户的信用分数，那么可创建一个 Java 类（调出）来执行此操作，然后在交互式流程图中的选择进程中使用 EXTERNALCALLOUT 宏以从调出中获取信用分数。

配置 EXTERNALCALLOUT 有两个主要步骤。首先，您必须创建一个用于实现 ExternalCallout API 的 Java 类。其次，您必须在 Interact | flowchart | ExternalCallouts 类别中配置运行时服务器上必需的 Marketing Platform 配置属性。

除了本节中的信息之外，任何 Interact 运行时服务器的 Interact/docs/externalCalloutJavaDoc 目录中也提供了 ExternalCallout API 的 JavaDoc。

IAffiniumExternalCallout 接口

ExternalCallout API 包含在接口 IAffiniumExternalCallout 中。您必须实现 IAffiniumExternalCallout 接口才能使用 EXTERNALCALLOUT 宏。

实现 IAffiniumExternalCallout 的类应具有一个构造函数，运行时服务器通过该构造函数来初始化此类。

- 如果在类中没有构造函数，那么 Java 编译器将创建一个缺省构造函数，这已足够。
- 如果有包含参数的构造函数，那么应提供一个不包含参数的公共构造函数，这将由运行时服务器使用。

开发您的外部调出时，请记住下列事项：

- 包含外部调出的每个表达式求值将创建类的新实例。您必须为类中的静态成员管理线程安全问题。
- 如果您的外部调出使用系统资源（如文件或数据库连接），那么您必须管理连接。运行时服务器没有自动清除连接的工具。

您必须针对位于 IBM Interact 运行时环境安装的 lib 目录中的 interact_externalcallout.jar 来编译您的实现。

IAffiniumExternalCallout 使运行时服务器能够从 Java 类中请求数据。该接口由四个方法组成：

- getNumberOfArguments
- getValue
- initialize
- shutdown

添加 Web Service 以与 EXTERNALCALLOUT 宏配合使用

使用此过程来添加要与 EXTERNALCALLOUT 宏配合使用的 Web Service。仅当您定义了相应配置属性时，EXTERNALCALLOUT 宏才可识别调出。

过程

在运行时环境的 Marketing Platform 中，在 Interact > flowchart > externalCallouts 类别中添加或定义以下配置属性。

配置属性	设置
externalCallouts 类别	为外部调出创建类别
class	外部调出的类名
classpath	外部调出类文件的类路径
Parameter Data 类别	如果您的外部调出需要参数，请为其创建新参数配置属性并为每个属性分配一个值

getNumberOfArguments

getNumberOfArguments 方法返回您正在与其集成的Java 类所期望的参数数量。

```
getNumberOfArguments()
```

返回值

getNumberOfArguments 方法返回一个整数。

示例

以下示例表明打印参数数量。

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

getValue 方法执行调出的核心功能并返回结果。

```
getValue(audienceID, configData, arguments)
```

getValue 方法需要以下参数：

- **audienceID** - 用于标识受众标识的值。
- **configData** - 一个映射，包含调出所必需的配置数据的键值对。
- **arguments** - 调出所必需的参数。每个参数可以是一个字符串、双精度类型、日期或它们其中一个的列表。列表参数可以包含空值，但是列表不能包含字符串和双精度类型（举例而言）。

  应在您的实现中进行参数类型检查。

如果 getValue 方法由于任何原因而失败，那么将返回 CalloutException。

返回值

getValue 方法返回字符串的列表。

示例

```
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

每当运行时服务器启动时，便会调用一次 initialize 方法。如果有任何可能在运行时期间妨碍性能的操作（例如，装入数据库表），那么应通过此方法来执行这些操作。

```
initialize(configData)
```

initialize 方法需要以下参数：

- **configData** - 一个映射，包含调出所必需的配置数据的键值对。

  Interact 从 Interact > Flowchart > External Callouts > [External Callout] > Parameter Data 类别中定义的外部调出参数中读取这些值。

如果 initialize 方法由于任何原因而失败，那么将返回 CalloutException。

返回值

无。

示例

```
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

每当运行时服务器关闭时，便会调用一次 shutdown 方法。如果具有您的调出所必需的任何清除任务，那么这些任务应在此时运行。

```
shutdown(configData)
```

shutdown 方法需要以下参数：

- **configData** - 一个映射，包含调出所必需的配置数据的键值对。

如果 shutdown 方法由于任何原因而失败，那么将返回 CalloutException。

返回值

无。

示例

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
    // shutdown scoreQueryUtility here
}
```

外部调出 API 示例

此示例创建用于获取信用分数的外部调出。

创建用于获取信用分数的外部调出：

1. 创建名为 GetCreditScore.java 并具有以下内容的文件。此文件假定存在一个名为 ScoreQueryUtility 的类，该类将从建模应用程序中访存分数。

```
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. 将 GetCreditScore.java 编译为 GetCreditScore.class。
3. 创建一个名为 creditscore.jar 的 JAR 文件，其中包含 GetCreditScore.class以及该文件使用的其他类文件。
4. 将 JAR 文件复制到运行时服务器上的某个位置，例如，/data/interact/creditscore.jar。
5. 在"管理配置"页面上的 externalCallouts 类别中，创建一个名称为 GetCreditScore 且类路径为 /data/interact/creditscore.jar 的外部调出。

6. 在交互式流程图中，该调出可以用作 EXTERNALCALLOUT('GetCreditScore')。

IInteractProfileDataService 接口

Profile Data Services API 包含在接口 iInteractProfileDataService 中。在 Interact 会话开始或者 Interact 会话的受众标识发生更改时，此接口使您能够通过一个或多个外部数据源（例如，平面文件、Web Service 等等）将分层数据导入至 Interact 会话。

要使用 Profile Data Services API 来开发分层结构数据导入，您必须编写用于从任何数据源中抽取信息的 Java 类并将其映射至 ISessionDataRootNode 对象，然后使用交互式流程图中"选择"进程中的 EXTERNALCALLOUT 宏引用该映射数据。

您必须针对位于 IBM Interact 运行时环境安装的 lib 目录中的 interact_externalcallout.jar 来编译您的实现。

有关使用此接口的一整套 Javadoc 文档，请使用任何一个 Web 浏览器查看 *Interact_home*/docs/externalCalloutJavaDoc 中的文件。

有关如何使用 Profile Data Services 的样本实现（其中包括有关如何实现该示例的带注释的描述），请参阅 *Interact_home*/samples/externalcallout/XMLProfileDataService.java。

注：该样本实现仅打算用作示例。您不应该在您的实现中使用此样本。

添加数据源以与 Profile Data Services 配合使用

使用此过程来添加要与 Profile Data Services 配合使用的数据源。

关于此任务

仅当您定义了相应配置属性时，EXTERNALCALLOUT 宏才可识别数据源以进行 Profile Data Services 分层数据导入。

过程

在运行时环境的 Marketing Platform 中，在 Interact > profile > Audience Levels > [AudienceLevelName] > Profile Data Services 类别中添加或定义以下配置属性。

配置属性	设置
New category Name 类别	您正在定义的数据源的名称。您在此处输入的名称在相同受众级别的数据源中必须唯一。
enabled	指示是否针对在其中定义的受众级别启用数据源。
className	用于实现 IInteractProfileDataService 的数据源类的标准名称。
classPath	您的 Profile Data Services 类文件的类路径。如果您忽略该配置设置，那么缺省情况下，会使用所包含应用程序服务器的类路径。

配置属性	设置
priority 类别	此受众级别内该数据源的优先级。它必须在每个受众级别的所有数据源中是唯一值。（即，如果某个数据源的优先级设置为 100，那么此受众级别内任何其他数据源的优先级都不能是 100。）

IParameterizableCallout 接口

"可参数化的调出"API 包含在 IParameterizableCallout 接口中。

此接口是已公开的 API 接口的基本接口，它可以通过 Marketing Platform 接受配置中的参数。由于此接口是基本接口，因此，不得直接实现此接口。参数从引用此实现的类别下 Parameter Data 节点的子节点中进行检索。在以下示例中，ESB 是概要文件数据服务的定制实现，并且进而实现 IParameterizableCallout 接口。Interact 引擎尝试初始化和终止此实现类时，参数 endPoint 和 login 及其值将传入此实现类。

```
Profile Data Services
...ESB
   ...Parameter Data
      ...endPoint
      ...login
```

该接口由两种方法组成：

- initialize
- shutdown

initialize

initialize 方法用于初始化此实现类。

```
void initialize(java.util.Map<java.lang.String,java.lang.String> configurationData)
               throws CalloutException
```

initialize 方法需要以下参数：

- **configurationData** - 一个图，此图具有用户配置的参数的"名称/值"对

Throws

CalloutException

shutdown

shutdown 方法用于关闭此实现类。

```
void shutdown(java.util.Map<java.lang.String,java.lang.String> configurationData)
              throws CalloutException
```

shutdown 方法需要以下参数：

- **configurationData** - 一个图，此图具有用户配置的参数的"名称/值"对

Throws

CalloutException

ITriggeredMessageAction 接口

"触发式消息操作"API 包含在 ITriggeredMessageAction 接口中。此接口使您能够获取并设置此实例的名称。

ITriggeredMessageAction 接口充当其他接口的基本接口，不得直接实现此接口。

该接口由两种方法组成：

- getName
- setName

getName

getName 方法用于返回 ITriggeredMessageAction 实例的名称。

```
java.lang.String getName()
```

setName

setName 方法用于设置 ITriggeredMessageAction 实例的名称。

```
void setName(java.lang.String name)
```

初始化此接口的实现类时，Interact 会使用配置 UI 中给出的名称设置此接口的名称。

在以下示例中，此网关的名称为 InteractLog。

```
triggeredMessage
           ...gateways
              ...InteractLog
```

setName 方法需要以下参数：

- name - 您希望为 ITriggeredMessageAction 实例设置的名称。

IChannelSelector 接口

"渠道选择器"API 包含在 IChannelSelector 接口中。此接口使您能够根据要发送的商品和会话属性选择出站渠道。

有关如何使用触发式消息操作的样本实现（其中包括有关如何实现该示例的带注释的描述），请参阅 *Interact_home*/samples/triggeredmessage/SampleChannelSelector.java。

注：该样本实现仅打算用作示例。您不应该在您的实现中使用此样本。

您应该尝试使用此实现，而不是编写自己的实现。

该接口由一种方法组成：

- selectChannels

selectChannels

selectChannels 方法用于选择应该使用 IChannelSelector 接口将传入商品发送到的出站渠道。

```
java.util.List<java.lang.String> selectChannels
                    (java.util.Map<java.lang.String,java.util.Map<java.lang.String,
                                  java.lang.Object>> availableChannels,
                                  com.unicacorp.interact.api.Offer offer,
                                  com.unicacorp.interact.treatment.
                                  optimization.IInteractSessionData sessionData)
```

Interact 尝试将此商品发送到返回的所有这些渠道。

selectChannels 方法需要以下参数：

- **availableChannels** - 可用出站渠道图，这些出站渠道在 Interact 设计时设置中的"触发式消息"UI 中进行了配置。在该图的每个条目中，键是渠道的名称，值是 Interact 设计时中为该渠道配置的参数。此图的迭代顺序与该 UI 中定义的顺序匹配。如果在"触发式消息"UI 中使用了概要文件首选渠道，那么在调用此方法之前，该渠道将被替换为实际渠道。另外，如果同一渠道在 UI 中多次出现，那么将仅保留具有最高优先级的实例，并除去所有重复项。
- **offer** - 要交付的商品
- **sessionData** - 相关联 Interact 会话中当前存储的属性

IDispatcher 接口

"分派器"API 包含在 IDispatcher 接口中。此接口用于将商品发送到目标网关。

由于对于每个已配置的分派器仅存在此类的一个实例，因此，从 Interact 的角度看，此接口的实现必须无状态。

有关如何使用触发式消息操作的样本实现（其中包括有关如何实现该示例的带注释的描述），请参阅 *Interact_home*/samples/triggeredmessage/SampleDispatcher.java。

注：该样本实现仅打算用作示例。您不应该在您的实现中使用此样本。

您应该尝试使用此实现，而不是编写自己的实现。

该接口由一种方法组成：

- dispatch

dispatch

dispatch 方法用于将商品发送到 IDispatcher 接口中的目标网关。

```
boolean dispatch(java.lang.String channel,
            java.lang.String gatewayName,
            java.util.Collection<com.unicacorp.interact.api.Offer> offers,
            com.unicacorp.interact.api.NameValuePair[] profileData)
              throws com.unicacorp.interact.exceptions.InteractException
```

为候选商品选择出站渠道之后，Interact 会尝试将候选商品发送到与该渠道相关联的处理程序。系统将根据这些处理程序的已定义优先级按从高到低的顺序尝试使用这些处理程序。对于每个处理程序，Interact 都将调用已配置分派器的此方法。商品传送到目标网关的方法由此分派器实例的实现决定，而目标网关在同一处理程序中配置。如果由于同一触发式消息评估需要将多个商品发送到同一处理程序，那么 Interact 会尝试在一个批次中发送所有这些商品。

dispatch 方法需要以下参数：

- **channel** - 这些商品将发送到的出站渠道
- **gatewayName** - 目标网关的名称
- **offers** - 要成批发送到网关的商品
- **profileData** - 通过 IGateway.validate 填充并且将传递到 IGateway.deliver 的概要文件属性

返回值

dispatch 方法返回分派结果，即，成功还是失败

Throws

com.unicacorp.interact.exceptions.InteractException

IGateway 接口

"网关"API 包含在 IGateway 接口中。此接口用于接收来自 Interact 的商品并将这些商品发送到其目标。

此接口的每种实现都与特定目标进行通信。目标必须执行必需的数据转换、属性填充以及相似的与目标相关的工作。

有关如何使用触发式消息操作的样本实现（其中包括有关如何实现该示例的带注释的描述），请参阅 *Interact_home*/samples/triggeredmessage/SampleOutboundGateway.java。

注：该样本实现仅打算用作示例。您不应该在您的实现中使用此样本。

该接口由两种方法组成：

- deliver
- validate

deliver

调用 deliver 方法可以将一个或多个商品发送到 IGateway 接口中的目标。

```
void deliver(java.util.Collection<com.unicacorp.interact.api.Offer> offers,
                    com.unicacorp.interact.api.NameValuePair[] profileData,
                    java.lang.String channel)
```

deliver 方法需要以下参数：

- **offers** - 要发送的商品
- **profileData** - validate 方法在 parameterMap 中填充的概要文件属性
- **channel** - 这些商品将发送到的出站渠道

validate

validate 方法用于验证 IGateway 接口中的候选商品。

```
java.util.Collection<com.unicacorp.interact.api.Offer> validate
    (com.unicacorp.interact.treatment.optimization.
     IInteractSessionData sessionData,
           java.util.Collection<com.unicacorp.interact.api.Offer> candidateOffers,
           java.util.Map<java.lang.String,java.lang.Object> parameterMap,
           java.lang.String channel)
```

Interact 引擎调用此方法来验证候选商品。此方法的实现应该根据目标要求检查商品、商品属性和会话属性，以确定可以通过此网关发送的商品。另外，它可以在传入图中添加所需参数，这些参数将传回到 deliver 方法。

validate 方法需要以下参数：

- **sessionData** - 相关联 Interact 会话中当前存储的属性
- **candidateOffers** - Interact 根据商品选择方法、该方法的参数和其他因素选择的商品。从 Interact 的角度来看，这些商品符合交付资格，但仍然受网关影响。
- **parameterMap** - 一个图，此方法的实现应该使用此图向其 deliver 方法传递参数
- **channel** - 这些商品将发送到的出站渠道

第 10 章 IBM Interact 实用程序

本节描述了 Interact 提供的管理实用程序。

运行部署实用程序 (runDeployment.sh/.bat)

runDeployment 命令行工具使您能够从命令行部署特定服务器组的交互式渠道，方法是使用概述了所有可能参数的 deployment.properties 文件所提供的和 runDeployment 工具本身所在位置中可用的设置。当您正在使用 OffersBySQL 功能时，从命令行运行交互式渠道部署的能力特别有用。例如，您可能会配置要定期运行的 Campaign 批处理流程图。当流程图运行完成时，可调用触发器以使用此命令行工具来初始化 OffersBySQL 表中商品的部署。

描述

您可以在以下位置中找到自动安装在 Interact 设计时服务器上的 runDeployment 命令行工具：

Interact_home/interactDT/tools/deployment/runDeployment.sh（或者 Windows 服务器上的 runDeployment.bat）

传递到该命令的唯一自变量是文件 deployment.properties 的位置，该文件描述了部署"交互式渠道/运行时服务器组"组合时需要的所有可能参数。提供了样本文件供参考。

注：在使用 runDeployment 实用程序之前，必须首先使用任意文本编辑器对其进行编辑以提供服务器上 Java 运行时环境的位置。例如，如果其中任意一个目录包含您希望该实用程序使用的 Java 运行时，那么您可能会指定 *Interact_home*/jre 或 *Platform_home*/jre 作为路径。或者，您也可以提供 IBM 产品的此发行版支持与其配合使用的任一 Java 运行时环境的路径。

在安全 (SSL) 环境中使用 runDeployment 实用程序

要在 Interact 服务器上启用了安全性（因而可以通过 SSL 端口进行连接）的情况下使用 runDeployment 实用程序，您需要按如下所示添加信任库 Java 属性：

1. 当您正在编辑 deployment.properties 文件以进行交互式渠道部署时，修改 deploymentURL 属性以使用安全 SSL URL，如以下示例中所示：

   ```
   deploymentURL=https://<HOST>.<DOMAIN>:<PORT>/Campaign/interact/
   InvokeDeploymentServlet
   ```

2. 通过使用任意文本编辑器来编辑 runDeployment.sh 或 runDeployment.bat 脚本以将以下自变量添加到以 ${JAVA_HOME} 开头的行中：

   ```
   -Djavax.net.ssl.trustStore=<TrustStorePath>
   ```

   例如，该行在添加信任库自变量后可能如下所示：

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

将 *<TrustStorePath>* 替换为实际 SSL 信任库的路径。

运行实用程序

已编辑实用程序以提供 Java 运行时环境，并且已定制匹配您环境的 deployment.properties 文件的副本之后，您可以使用以下命令来运行实用程序：

Interact_home/interactDT/tools/deployment/runDeployment.sh *deployment.properties*

将 *Interact_home* 替换为 Interact 设计时安装的实际值，并将 *deployment.properties* 替换为您为此部署定制的属性文件的实际路径和名称。

样本 deployment.properties 文件

样本 deployment.properties 文件包含您必须定制以匹配您自己的环境的所有参数的带注释列表。样本文件还包含用于描述各参数的内容以及可能需要定制特定值的原因的注释。

```
##############################################################################
#
# 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
##############################################################################
```

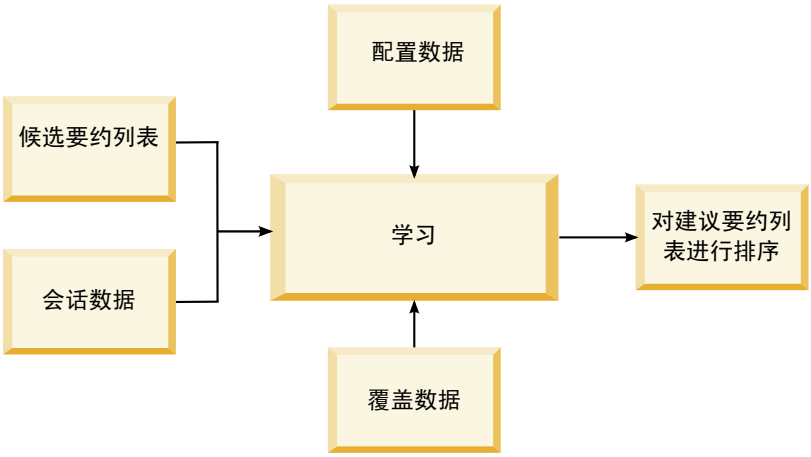
```
icName=ic1

################################################################################
# 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 11 章 关于学习 API

Interact 提供了一个学习模块，此模块使用 naive-bayesian 算法来监视访问者操作并建议最佳商品（在接受程度方面）。 您可以通过学习 API，采用您自己的算法来实现相同 Java 接口，从而创建您自己的学习模块。

注：如果您使用外部学习，那么与学习有关的示例报告（"交互式商品学习详细信息"和"交互式细分市场提升分析"报告）不会返回有效数据。

在最简单的程度上，学习 API 提供了用于从运行时环境中收集数据以及返回推荐商品的已排序列表的方法。



您可以从 Interact 中收集以下数据

- 商品联系数据
- 商品接受数据
- 所有会话数据
- 特定于 Campaign 的商品数据
- 设计环境的 `learning` 类别以及运行时环境的 `offerserving` 类别中定义的配置属性

您可以在您的算法中使用此数据来创建建议商品的列表。然后，您将按照从最高到最低推荐顺序返回推荐商品的列表。

尽管图中未显示，但是您还可以使用学习 API 来为您的学习实现收集数据。您可将此数据保留在内存中，或者将其记录到文件或数据库中以便将来分析。

在创建 Java 类之后，您可以将其转换为 jar 文件。在创建 jar 文件之后，必须还要通过编辑配置属性来配置运行时环境以识别您的外部学习模块。您必须将您的 Java 类或 jar 文件复制到使用您的外部学习模块的每个运行时服务器。

除了本指南中的信息外，任何运行时服务器的 `Interact/docs/learningOptimizerJavaDoc` 目录中也提供了学习优化器 API 的 Javadoc。

您必须针对位于 Interact 运行时环境安装的 `lib` 目录中的 `interact_learning.jar` 来编译您的实现。

编写定制学习实现时，请切记以下准则。

- 性能至关重要。
- 必须使用多线程并且线程必须安全。
- 必须在考虑故障方式和性能的情况下管理所有外部资源。
- 相应地使用异常、记录 (log4j) 和内存。

将运行时环境配置为识别外部学习模块

您可以使用学习 Java API 来编写您自己的学习模块。您必须将运行时环境配置为在 Marketing Platform 中识别您的学习实用程序。

关于此任务

您必须重新启动 Interact 运行时服务器以使这些更改生效。

过程

1. 在运行时环境的 Marketing Platform 中，编辑 `Interact > offerserving` 类别中的以下配置属性。学习优化器 API 的配置属性位于 `Interact > offerserving > External Learning Config` 类别中。

配置属性	设置
`optimizationType`	**ExternalLearning**
`externalLearningClass`	外部学习的类名
`externalLearningClassPath`	运行时服务器上用于外部学习的类文件或 JAR 文件的路径。如果您在使用服务器组并且所有运行时服务器都引用 Marketing Platform 的同一实例，那么每个服务器必须在相同位置具有这些类文件或 JAR 文件的副本。

2. 重新启动 Interact 运行时服务器以使这些更改生效。

ILearning 接口

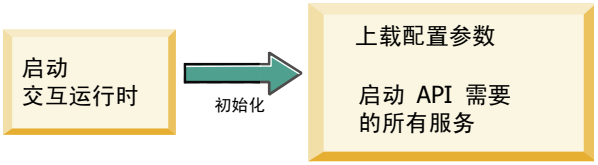
学习 API 是根据 `ILearning` 接口来构建。您必须实现 `ILearning` 接口才能支持您的学习模块的定制逻辑。

其中，`ILearning` 接口使您可以从 Java 类的运行时环境中收集数据，以及将建议商品的列表发回到运行时服务器。

initialize

每当运行时服务器启动时，便会调用一次 `initialize` 方法。如果有任何不需要重复但是可能在运行时期间妨碍性能的操作（例如，从数据库表装入静态数据），那么应通过此方法来执行这些操作。

`initialize(ILearningConfig *config*, boolean *debug*)`



- **config** - ILearningConfig 对象，用于定义与学习相关的所有配置属性。
- **debug** - 布尔值。如果为 true，那么指示运行时环境系统的记录级别详细程度设置为调试。为了获得最佳结果，请在写入日志之前选择此值。

如果 initialize 方法由于任何原因而失败，那么将抛出 LearningException。

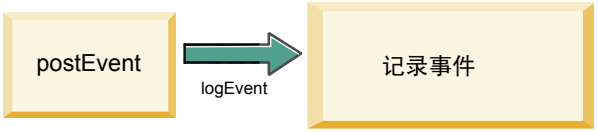
返回值

无。

logEvent

只要 Interact API 发布配置为记录为联系或响应的事件，运行时服务器便会调用 logEvent 方法。使用此方法可将联系和响应数据记录到数据库或文件，以进行报告和学习。例如，如果您要以算法形式，根据条件来确定客户接受某一商品的可能性，请使用此方法来记录数据。

```
logEvent(ILearningContext context,
              IOffer offer,
              IClientArgs clientArgs,
              IInteractSession session,
              boolean debug)
```



- **context** - ILearningContext 对象，用于定义事件（例如，联系、接受或拒绝）的学习上下文。
- **offer** - IOffer 对象，用于定义为其记录此事件的商品。
- **clientArgs** - IClientArgs 对象，用于定义任何参数。目前，logEvent 不需要任何 clientArgs，因此该参数可能为空。
- **session** - IInteractSession 对象，用于定义所有会话数据。
- **debug** - 布尔值。如果为 true，那么指示运行时环境系统的记录级别详细程度设置为调试。为了获得最佳结果，请在写入日志之前选择此值。

如果 logEvent 方法失败，那么将抛出 LearningException。

返回值

无。

optimizeRecommendList

optimizeRecommendList 方法应获取建议商品和会话数据的列表，并且返回一个包含请求商品数量的列表。optimizeRecommendList 方法应使用您自己的学习算法，以某种方

法来对商品进行排序。必须对商品的列表进行排序，以便您要首先提供的商品位于列表的开头。例如，如果您的学习算法为最佳商品给出了较低分数，那么商品应按照 1、2、3 进行排序。如果您的学习算法为最佳商品给出了较高分数，那么您的商品应按照 100、99、98 进行排序。

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

optimizeRecommendList 方法需要以下参数：

- **recList** - 运行时环境建议的处理对象（商品）的列表。
- **clientArg** - IClientArgs 对象，其中至少包含运行时环境所请求的商品数量。
- **session** - IInteractSession 对象，其中包含所有会话数据。
- **debug** - 布尔值。如果为 true，那么指示运行时环境系统的记录级别详细程度设置为调试。为了获得最佳结果，请在写入日志之前选择此值。

如果 optimizeRecommendList 方法失败，那么将抛出 LearningException。

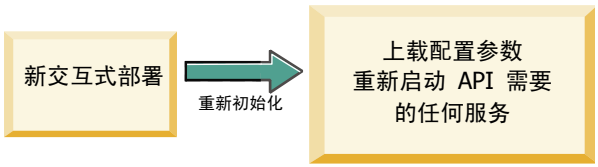
返回值

optimizeRecommendList 方法返回ITreatment 对象的列表。

reinitialize

每当具有新部署时，运行时环境都将调用 reinitialize 方法。此方法可传递所有学习配置数据。如果学习 API 所需要的任何服务要读取配置属性，那么此接口应重新启动这些服务。

```
reinitialize(ILearningConfig config,
                  boolean debug)
```



- **config** - ILearningConfig 对象，其中包含所有配置属性。
- **debug** - 布尔值。如果为 true，那么指示运行时环境系统的记录级别详细程度设置为调试。为了获得最佳结果，请在写入日志之前选择此值。

如果 logEvent 方法失败，那么将抛出 LearningException。

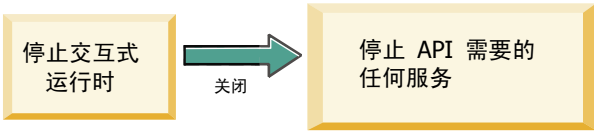
返回值

无。

shutdown

当运行时服务器关闭时，运行时环境将调用 shutdown 方法。如果有您的学习模块所必需的任何清除任务，那么这些任务应在此时运行。

```
shutdown(ILearningConfig config, boolean debug)
```



shutdown 方法需要以下参数。

- **config** - ILearningConfig 对象，用于定义所有配置属性。
- **debug** - 布尔值。如果为 true，那么指示运行时环境系统的记录级别详细程度设置为调试。为了获得最佳结果，请在写入日志之前选择此值。

如果 shutdown 方法由于任何原因而失败，那么将抛出 LearningException。

返回值

无。

IAudienceID 接口

IAudienceID 接口支持 IInteractSession 接口。这是受众标识的接口。由于您的受众标识可能包含多个部分，因此该接口使您可以访问受众标识的所有元素以及受众级别名称。

getAudienceLevel

getAudienceLevel 方法返回受众级别。

```
getAudienceLevel()
```

返回值

getAudienceLevel 方法返回用于定义受众级别的字符串。

getComponentNames

getComponentNames 方法获取用于组成受众标识的构成项的一组名称。例如，如果您的受众标识由 customerName 和 accountID 的值组成，那么 getComponentNames 将返回包含字符串 customerName 和 accountID 的集合。

```
getComponentNames()
```

返回值

一组字符串，其中包含受众标识的构成项的名称。

getComponentValue

getComponentValue 方法返回指定组件的值。

getComponentValue(String *componentName*)

- **componentName** - 一个字符串，用于定义您要检索其值的组件的名称。此字符串不区分大小写。

返回值

getComponentValue 方法返回用于定义组件的值的对象。

IClientArgs

IClientArgs 接口支持 ILearning 接口。此接口是一个抽象，用于向会话数据尚未涵盖的接触点隐藏传递到服务器的任何数据。例如，Interact API 的 getOffers 方法请求的商品的数量。此数据存储在映射中。

getValue

getValue 方法返回请求的映射元素的值。

getValue(int *clientArgKey*)

以下元素在映射中是必需的。

- **1** - NUMBER_OF_OFFERS_REQUESTED。Interact API 的 getOffers 方法所请求的商品数量。此常量返回一个整数。

返回值

getValue 方法返回用于定义请求的映射常量值的对象。

IInteractSession

IInteractSession 接口支持 ILearning 接口。这是运行时环境中当前会话的接口。

getAudienceId

getAudienceId 方法返回 AudienceID 对象。使用 IAudienceID 接口可抽取值。

getAudienceId()

返回值

getAudienceId 方法返回 AudienceID 对象。

getSessionData

getSessionData 方法返回会话数据的不可修改映射，在此映射中，会话变量的名称是键。会话变量的名称始终大写。使用 IInteractSessionData 接口可抽取值。

getSessionData()

返回值

getSessionData 方法返回 IInteractSessionData 对象。

IInteractSessionData 接口

IInteractSessionData 接口支持 ILearning 接口。这是当前访问者的运行时会话数据的接口。会话数据存储为名称/值对的列表。您还可以使用此接口来更改运行时会话中数据的值。

getDataType

getDataType 方法返回指定参数名的数据类型。

```
getDataType(string parameterName)
```

返回值

getDataType 方法返回一个 InteractDataType 对象。IntearctDataType 是一个 Java 枚举，以 Unknown、String、Double、Date 或 List 来表示。

getParameterNames

getParameterNames 方法返回当前会话中数据的所有名称的集合。

```
getParameterNames()
```

返回值

getParameterNames 方法返回一组已为值设置的名称。集合中的每个名称都可以传递到 getValue(String) 以返回某个值。

getValue

getValue 方法返回对应于指定 parameterName 的对象值。对象可以是 String、Double 或 Date。

```
getValue(parameterName)
```

getValue 方法需要以下参数：

- **parameterName** - 用于定义会话数据"名称/值"对的名称的字符串。

返回值

getValue 方法返回包含指定参数的值的对象。

setValue

setValue 方法使您可以为指定 parameterName 设置一个值。值可以是 String、Double 或 Date。

```
setValue(string parameterName, object value)
```

setValue 方法需要以下参数：

- **parameterName** - 用于定义会话数据"名称/值"对的名称的字符串。
- **value** - 用于定义所指定参数的值的对象。

返回值

无。

ILearningAttribute

ILearningAttribute 接口支持 ILearningConfig 接口。这是 learningAttributes 类别中配置属性中定义的学习属性的接口。

getName

getName 方法返回学习属性的名称。

```
getName()
```

返回值

getName 方法返回用于定义学习属性名称的字符串。

ILearningConfig

ILearningConfig 接口支持 ILearning 接口。这是学习的配置属性的接口。所有这些方法都返回属性的值。

该接口由 15 个方法组成：

- **getAdditionalParameters** - 返回 External Learning Config 类别中定义的任何其他属性的映射
- **getAggregateStatsIntervalInMinutes** - 返回一个整数
- **getConfidenceLevel** - 返回一个整数
- **getDataSourceName** - 返回一个字符串
- **getDataSourceType** - 返回一个字符串
- **getInsertRawStatsIntervalInMinutes** - 返回一个整数
- **getLearningAttributes** - 返回 ILearningAttribute 对象的列表
- **getMaxAttributeNames** - 返回一个整数
- **getMaxAttributeValues** - 返回一个整数
- **getMinPresentCountThreshold** - 返回一个整数
- **getOtherAttributeValue** - 返回一个字符串
- **getPercentRandomSelection** - 返回一个整数
- **getRecencyWeightingFactor** - 返回一个浮点数
- **getRecencyWeightingPeriod** - 返回一个整数
- **isPruningEnabled** - 返回一个布尔值

ILearningContext

ILearningContext 接口支持 ILearning 接口。

getLearningContext

getLearningContext 方法返回一个常量，此常量告知我们此常量是联系、接受还是拒绝方案。

```
getLearningContext()
```

- **1**-LOG_AS_CONTACT

- **2**-LOG_AS_ACCEPT
- **3**-LOG_AS_REJECT

4 和 5 保留以供将来使用。

返回值

getLearningContext 方法返回一个整数。

getResponseCode

getResponseCode 方法返回分配给此商品的响应代码。此值必须存在于 Campaign 系统表的 UA_UsrResponseType 表中。

getResponseCode()

返回值

getResponseCode 方法返回用于定义响应代码的字符串。

IOffer

IOffer 接口支持 ITreatment 接口。这是设计环境中定义的商品对象的接口。使用 IOffer 接口可从运行时环境中收集商品详细信息。

getCreateDate

getCreateDate 方法返回创建商品的日期。

getCreateDate()

返回值

getCreateDate 方法返回一个用于定义商品创建日期的日期。

getEffectiveDateFlag

getEffectiveDateFlag 方法返回一个用于定义商品生效日期的日期。

getEffectiveDateFlag()

- **0** - 生效日期为绝对日期，如 2010 年 3 月 15 日。
- **1** - 生效日期是推荐日期。

返回值

getEffectiveDateFlag 方法返回一个用于定义商品生效日期的整数。

getExpirationDateFlag

getExpirationDateFlag 方法返回用于描述商品截止日期的整数值。

getExpirationDateFlag()

- **0** - 一个绝对日期，例如，2010 年 3 月 15 日。
- **1** - 推荐之后的某个天数，例如，14 天。
- **2** - 推荐之后的月尾。如果商品在 3 月 31 日呈现，那么商品于该日截止。

返回值

getExpirationDateFlag 方法返回用于描述商品截止日期的整数。

getOfferAttributes

The getOfferAttributes 方法以 IOfferAttributes 对象的形式返回为商品定义的商品属性。

```
getOfferAttributes()
```

返回值

getOfferAttributes 方法返回 IOfferAttributes 对象。

getOfferCode

getOfferCode 方法返回商品的商品代码（如 Campaign 中定义）。

```
getOfferCode()
```

返回值

getOfferCode 方法返回 IOfferCode 对象。

getOfferDescription

getOfferDescription 方法返回 Campaign 中定义的商品的描述。

```
getOfferDescription()
```

返回值

getOfferDescription 方法返回一个字符串。

getOfferID

getOfferID 方法返回商品标识（如 Campaign 中所定义）。

```
getOfferID()
```

返回值

getOfferID 方法返回一个用于定义商品标识的长整型。

getOfferName

getOfferName 方法返回商品的名称（如 Campaign 中定义）。

```
getOfferName()
```

返回值

getOfferName 方法返回一个字符串。

getUpdateDate

getUpdateDate 方法返回最近一次更新商品的日期。

```
getUpdateDate()
```

返回值

getUpdateDate 方法返回一个用于定义商品最近一次更新日期的日期。

IOfferAttributes

IOfferAttributes 接口支持 IOffer 接口。这是在设计环境中为某个商品定义的商品属性的接口。使用 IOfferAttributes 接口可从运行时环境中收集商品属性。

getParameterNames

getParameterNames 方法返回商品参数名称的列表。

getParameterNames()

返回值

getParameterNames 方法返回一个集合，此集合定义了商品参数名称的列表。

getValue

getValue 方法返回用于定义商品属性的值的对象。

getValue(String *parameterName*)

getValue 方法返回给定商品属性的值。

返回值

IOfferCode 接口

IOfferCode 接口支持 ILearning 接口。这是在设计环境中为某个商品定义的商品代码的接口。商品代码可以由一到多个字符串组成。使用 IOfferCode 接口可从运行时环境中收集商品代码。

getPartCount

getPartCount 方法返回构成商品代码的部分的数量。

getPartCount()

返回值

getPartCount 方法返回用于定义商品代码组成部分的数量的整数。

getParts

getParts 方法获取商品代码部分的不可修改列表。

getParts()

返回值

getParts 方法返回商品代码部分的不可修改列表。

LearningException

LearningException 类支持 ILearning 接口。此接口中的某些方法将需要某些实现来抛出 LearningException（即，java.lang.Exception 的简单子类）。出于调试目的，在存在根异常的情况下，强烈建议通过根异常来构造 LearningException。

IScoreOverride

IScoreOverride 接口支持 ITreatment 接口。此接口使您可以读取分数覆盖或缺省商品表中定义的数据。

getOfferCode

getOfferCode 方法返回此受众成员中分数覆盖表中商品代码列的值。

```
getOfferCode()
```

返回值

getOfferCode 方法返回一个 IOfferCode 对象，此对象定义分数覆盖表中商品代码列的值。

getParameterNames

getParameterNames 方法返回参数的列表。

```
getParameterNames()
```

返回值

getParameterNames 方法返回一个集合，此集合定义了参数的列表。

IScoreOverride 方法包含以下参数。除非另有描述，否则这些参数与分数覆盖表相同。

- ADJ_EXPLORE_SCORE_COLUMN
- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT - 用于覆盖估计的呈示次数（在商品权重计算期间）
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN - 用于创建布尔表达式以确定商品合格性
- PREDICATE_SCORE - 用于创建导致数字分数的表达式
- SCORE_COLUMN
- ZONE_COLUMN

您还可以使用与列相同的名称来引用您添加到分数覆盖表或缺省商品表的任何列。

getValue

getValue 方法返回此受众成员中分数覆盖表中区域列的值。

getValue(String *parameterName*)

- **parameterName** - 一个字符串，用于定义您要获取其值的参数的名称。

返回值

getValue 方法返回一个用于定义所请求参数的值的对象。

ISelectionMethod

ISelection 接口指示用于构成建议列表的方法。Treatment 对象的缺省值为 EXTERNAL_LEARNING，因此您不一定必须设置此值。此值最终存储在详细联系历史记录中以进行报告。

如果您要存储数据以在稍后进行分析，那么您可以在现有常量之外扩展此接口。例如，您可以创建两个不同学习模块并在单独服务器组上实施这两个模块。您可以扩展 ISelection 接口以包含 SERVER_GROUP_1 和 SERVER_GROUP_2。然后您可以比较这两个学习模块的结果。

ITreatment 接口

ITreatment 接口支持 ILearning 接口作为处理信息的接口。处理表示分配给特定单元的商品（如设计环境中定义）。通过此接口，您可以获取单元和商品信息以及分配的市场营销分数。

getCellCode

getCellCode 方法返回单元代码（如 Campaign 中所定义）。单元是指分配给与此商品相关联的智能细分市场的单元。

getCellCode()

返回值

getCellCode 方法返回用于定义单元代码的字符串。

getCellId

getCellId 方法返回单元的内部标识（如 Campaign 中所定义）。单元是指分配给与此商品相关联的智能细分市场的单元。

getOfferName()

返回值

getCellId 方法返回一个用于定义单元标识的长整型。

getCellName

getCellName 方法返回单元的名称（如 Campaign 中定义）。单元是指分配给与此商品相关联的智能细分市场的单元。

getCellName()

返回值

getCellName 方法返回用于定义单元名称的字符串。

getLearningScore

getLearningScore 方法返回此处理的分数。

```
getLearningScore()
```

优先顺序如下所示。

1. 返回覆盖值（如果 IScoreoveride.PREDICATE_SCORE_COLUMN 键控的覆盖值映射中存在）
2. 返回预测分数（如果值不为空）
3. 返回市场营销人员分数（如果 IScoreoveride.SCORE 键控的覆盖值映射中存在）
4. 返回市场营销人员分数

返回值

getLearningScore 方法返回一个整数，此整数定义了学习算法所确定的分数。

getMarketerScore

getMarketerScore 方法返回在商品的"交互策略"选项卡上的滑块所定义的市场营销人员的分数。

```
getMarketerScore()
```

要检索"交互策略"选项卡高级选项所定义的市场营销人员的分数，请使用 getPredicateScore。

要检索由处理实际使用的市场营销人员的分数，请使用 getLearningScore。

返回值

getMarketerScore 方法返回一个整数，此整数用于定义市场营销人员的分数。

getOffer

getOffer 方法返回要进行处理的商品。

```
getOffer()
```

返回值

getOffer 方法返回 IOffer 对象，该对象定义要进行此处理的商品。

getOverrideValues

getOverrideValues 方法返回缺省商品或分数覆盖表中定义的覆盖。

```
getOverrideValues()
```

返回值

getOverrideValues 方法返回 IScoreOverride 对象。

getPredicate

getPredicate 方法返回缺省商品表、分数覆盖表或处理规则高级选项的谓词列所定义的谓词。

```
getPredicate()
```

返回值

getPredicate 方法返回一个字符串，此字符串定义缺省商品表、分数覆盖表或处理规则高级选项的谓词列所定义的谓词。

getPredicateScore

getPredicateScore 方法返回缺省商品表、分数覆盖表或处理规则高级选项的谓词列所设置的分数。

```
getPredicateScore()
```

返回值

getPredicateScore 方法返回一个双精度类型，用于定义缺省商品表、分数覆盖表或处理规则高级选项的谓词列所设置的分数。

getScore

getScore 方法将返回市场营销分数，该分数由 Campaign 中的交互策略或者由分数覆盖表进行定义。

```
getScore()
```

getScore 方法返回以下其中一个对象：

- 商品的市场营销分数，如 Campaign 中的"交互策略"选项卡中所定义（在 enableScoreOverrideLookup 属性设置为 false 的情况下）。
- 商品的分数，如 scoreOverrideTable 所定义（在 enableScoreOverrideLookup 属性设置为 true 的情况下）。

返回值

getScore 方法将返回一个表示商品分数的整数。

getTreatmentCode

getTreatmentCode 方法返回处理代码。

```
getTreatmentCode()
```

返回值

getTreatmentCode 方法返回用于定义处理代码的字符串。

setActualValueUsed

使用 setActualValueUsed 方法可定义在学习算法执行中各个阶段使用的值。

```
setActualValueUsed(string parmName, object value)
```

例如，如果您使用此方法来写入到联系和响应历史记录表，并修改现有样本报告，那么您可以在报告中包含学习算法中的数据。

- **parmName** - 一个字符串，用于定义您在设置的参数的名称。
- **value** - 一个对象，用于定义您在设置的参数的值。

返回值

无。

学习 API 示例

本节包含 ILearning 接口的实现样本。请注意，此实现只是一个样本，并非设计为用于生产环境。

此示例跟踪接受计数和联系计数，并使用某一特定商品的接受与联系之比来作为此商品的接受可能性等级。未显示的商品将获得较高的推荐优先级。至少具有一次联系的商品将根据递减的接受可能性评级来进行排序。

在此示例中，所有计数均保存在内存中。这是一种不现实的方案，因为运行时服务器将耗尽内存。在实际生产方案中，计数应持久存储在数据库中。

```
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);

        }
    }

}
```

第 12 章 IBM Interact WSDL

Interact 安装包括两个 WDSL（Web Service 描述语言）XML 文件，该文件描述可用的 Web service 以及如何访问这些 Web service。您可以查看 Interact 主目录中的这些文件，如以下示例所示。

在安装 Interact 之后，您可以在以下位置找到 Interact WSDL 文件：

- *<Interact_home>*/conf/InteractService.wsdl
- *<Interact_home>*/conf/InteractAdminService.wsdl

在各个软件发行版或修订包中，可能存在对 Interact WDSL 的更改。请参阅该发行版的 Interact 发行说明或自述文件以了解详细信息。

下面显示了 InteractService.wsdl 的一个副本。要确保您使用的是最新信息，请参阅随 Interact 安装的 WDSL 文件。

```
<?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>
```

第 13 章 Interact 运行时环境配置属性

本节描述 Interact 运行时环境的所有配置属性。

Interact | general

这些配置属性定义运行时环境的常规设置，包括缺省记录级别和语言环境设置。

log4jConfig

描述

包含 log4j properties 的文件的位置。此路径必须相对于 INTERACT_HOME 环境变量。INTERACT_HOME 是 Interact 安装目录的位置。

缺省值

./conf/interact_log4j.properties

asmUserForDefaultLocale

描述

asmUserForDefaultLocale 属性定义 IBM Marketing Software 用户，Interact 从该用户派生其语言环境设置。

语言环境设置定义在设计时显示什么语言，以及来自 Interact API 的报告消息采用什么语言。如果语言环境设置与机器操作系统设置不匹配，那么 Interact 仍可工作，但是设计时显示和报告消息可能会采用不同的语言。

缺省值

asm_admin

Interact | general | learningTablesDataSource

这些配置属性定义内置学习表的数据源设置。如果您要使用 Interact 内置学习，那么必须定义此数据源。

如果您要使用学习 API 创建您自己的学习实现，那么可以配置定制学习实现以使用 ILearningConfig 接口读取这些值。

jndiName

描述

使用此 jndiName 属性来识别在应用程序服务器（Websphere 或 WebLogic）中为由 Interact 运行时服务器访问的学习表定义的 Java 命名和目录接口 (JNDI) 数据源。

学习表通过 aci_lrntab ddl 文件创建，并且它们包含下列表（及其他表）：UACI_AttributeValue 和 UACI_OfferStats。

缺省值

未定义缺省值。

type

描述

由 Interact 运行时服务器访问的学习表所用数据源的数据库类型。

学习表通过 aci_lrntab ddl 文件创建，并且它们包含下列表（及其他表）：UACI_AttributeValue 和 UACI_OfferStats。

缺省值

SQLServer

有效值

SQLServer | DB2 | ORACLE

connectionRetryPeriod

描述

ConnectionRetryPeriod 属性指定当学习表失败时，Interact 自动重试数据库连接请求的时间长度（以秒计）。在报告数据库错误或失败之前，在此时间长度内，Interact 会自动尝试重新连接至数据库。如果将值设置为 0，那么 Interact 会重试无限多次；如果将值设置为 -1，那么不会重试。

学习表通过 aci_lrntab ddl 文件创建，并且它们包含下列表（及其他表）：UACI_AttributeValue 和 UACI_OfferStats。

缺省值

-1

connectionRetryDelay

描述

ConnectionRetryDelay 属性指定在学习表失败之后，Interact 在尝试重新连接至数据库之前等待的时间长度（以秒计）。如果将值设置为 -1，那么不会重试。

学习表通过 aci_lrntab ddl 文件创建，并且它们包含下列表（及其他表）：UACI_AttributeValue 和 UACI_OfferStats。

缺省值

-1

模式

描述

包含内置学习模块的表的模式名称。Interact 会在所有表名前面插入此属性的值，例如，UACI_IntChannel 将变成 schema.UACI_IntChannel。

不必定义模式。如果不定义模式，那么 Interact 会假设表的所有者与模式的所有者相同。应该设置此值以消除歧义。

缺省值

未定义缺省值。

Interact | general | prodUserDataSource

这些配置属性定义生产概要文件表的数据源设置。必须定义此数据源。这是在部署后运行交互式流程图时，运行时环境引用的数据源。

jndiName

描述

使用此 jndiName 属性来识别在应用程序服务器（Websphere 或 WebLogic）中为由 Interact 运行时服务器访问的客户表定义的 Java 命名和目录接口 (JNDI) 数据源。

缺省值

未定义缺省值。

type

描述

由 Interact 运行时服务器访问的客户表的数据库类型。

缺省值

SQLServer

有效值

SQLServer | DB2 | ORACLE

aliasPrefix

描述

AliasPrefix 属性指定 Interact 形成别名的方式；该别名在 Interact 使用维度表并写入由 Interact 运行时服务器访问的客户表中的新表时自动创建。

请注意，每个数据库都具有最大标识长度；请检查要使用的数据库的文档，以确保您设置的值不会超过数据库的最大标识长度。

缺省值

A

connectionRetryPeriod

描述

ConnectionRetryPeriod 属性指定当运行时客户表失败时，Interact 自动重试数据库连接请求的时间长度（以秒计）。在报告数据库错误或失败之前，在此时间长度内，Interact 会自动尝试重新连接至数据库。如果将值设置为 0，那么 Interact 会重试无限多次；如果将值设置为 -1，那么不会重试。

缺省值

-1

connectionRetryDelay

描述

ConnectionRetryDelay 属性指定在 Interact 运行时客户表失败之后，Interact 在尝试重新连接至数据库之前等待的时间长度（以秒计）。如果将值设置为 -1，那么不会重试。

缺省值

-1

模式

描述

包含概要文件数据表的模式名称。Interact 会在所有表名前面插入此属性的值，例如，UACI_IntChannel 将变成 schema.UACI_IntChannel。

不必定义模式。如果不定义模式，那么 Interact 会假设表的所有者与模式的所有者相同。应该设置此值以消除歧义。

使用 DB2 数据库时，模式名称必须是大写。

缺省值

未定义缺省值。

Interact | general | systemTablesDataSource

这些配置属性定义运行时环境的系统表的数据源设置。必须定义此数据源。

jndiName

描述

使用此 jndiName 属性来识别在应用程序服务器（Websphere 或 WebLogic）中为运行时环境表定义的 Java 命名和目录接口 (JNDI) 数据源。

运行时环境数据库是用 aci_runtime 和 aci_populate_runtime dll 脚本填充的数据库，并且包含下面举例描述的表和其他表：UACI_CHOfferAttrib 和 UACI_DefaultedStat。

缺省值

未定义缺省值。

type

描述

运行时环境系统表的数据库类型。

运行时环境数据库是用 aci_runtime 和 aci_populate_runtime dll 脚本填充的数据库，并且包含下面举例描述的表和其他表：UACI_CHOfferAttrib 和 UACI_DefaultedStat。

缺省值

SQLServer

有效值

SQLServer | DB2 | ORACLE

connectionRetryPeriod

描述

ConnectionRetryPeriod 属性指定当运行时系统表失败时，Interact 自动重试数据库连接请求的时间长度（以秒计）。在报告数据库错误或失败之前，在此时间长度内，Interact 会自动尝试重新连接至数据库。如果将值设置为 0，那么 Interact 会重试无限多次；如果将值设置为 -1，那么不会重试。

运行时环境数据库是用 aci_runtime 和 aci_populate_runtime dll 脚本填充的数据库，并且包含下面举例描述的表和其他表：UACI_CHOfferAttrib 和 UACI_DefaultedStat。

缺省值

-1

connectionRetryDelay

描述

ConnectionRetryDelay 属性指定在 Interact 运行时系统表失败之后，Interact 在尝试重新连接至数据库之前等待的时间长度（以秒计）。如果将值设置为 -1，那么不会重试。

运行时环境数据库是用 aci_runtime 和 aci_populate_runtime dll 脚本填充的数据库，并且包含下面举例描述的表和其他表：UACI_CHOfferAttrib 和 UACI_DefaultedStat。

缺省值

-1

模式

描述

包含运行时环境的表的模式名称。Interact 会在所有表名前面插入此属性的值，例如，UACI_IntChannel 将变成 schema.UACI_IntChannel。

不必定义模式。如果不定义模式，那么 Interact 会假设表的所有者与模式的所有者相同。应该设置此值以消除歧义。

缺省值

未定义缺省值。

Interact | general | systemTablesDataSource | loaderProperties

这些配置属性定义运行时环境系统表的数据库装入程序实用程序的设置。仅当您要使用数据库装入程序实用程序时，才需要定义这些属性。

databaseName

描述

数据库装入程序连接至的数据库的名称。

缺省值

未定义缺省值。

LoaderCommandForAppend

描述

LoaderCommandForAppend 参数指定发出来调用数据库装入实用程序以将记录追加到 Interact 内联系人和响应历史记录登台数据库表中的命令。需要设置此参数以启用联系人和响应历史记录数据的数据库装入程序实用程序。

此参数指定为数据库装入实用程序可执行文件或启动数据库装入实用程序的脚本的完整路径名。使用脚本允许您在调用装入实用程序之前执行其他设置。

大部分数据库装入实用程序都需要几个参数才能成功启动。这些参数可以包括指定从中装入的数据文件和控制文件以及要装入到的数据库和表。当命令运行时，这些标记将替换为指定的元素。

有关调用数据库装入实用程序时要使用的正确语法，请参阅数据库装入实用程序文档。

缺省情况下未定义此参数。

下表描述了可用于 LoaderCommandForAppend 的标记。

标记	描述
<CONTROLFILE>	用 Interact 根据在 LoaderControlFileTemplate 参数中指定的模板生成的临时控制文件的完整路径和文件名替换此标记。
<DATABASE>	用 Interact 将数据装入其中的数据源的名称替换此标记。此名称与该数据源的类别名称中使用的数据源名称相同。
<DATAFILE>	用在装入过程中由 Interact 创建的临时数据文件的完整路径和文件名替换此标记。此文件位于 Interact 临时目录 UNICA_ACTMPDIR 中。
<DBCOLUMNNUMBER>	用通常在数据库中的列替换此标记。
<FIELDLENGTH>	此标记将替换为装入数据库的字段的长度。
<FIELDNAME>	此标记将替换为装入数据库的字段的名称。
<FIELDNUMBER>	此标记将替换为装入数据库的字段的编号。
<FIELDTYPE>	此标记将被替换为字面值"CHAR()"。此字段的长度在 () 之间指定。如果数据库不了解字段类型 CHAR，那么可以手动指定字段类型的适当文本，并使用 <FIELDLENGTH> 标记。例如，对于 SQLSVR 和 SQL2000，应使用 "SQLCHAR(<FIELDLENGTH>)"
<NATIVETYPE>	用将此字段装入到的数据库的类型替换此标记。
<NUMFIELDS>	此标记将替换为表中字段的数目。

标记	描述
<PASSWORD>	此标记将替换为从当前流程图连接到数据源的数据库密码。
<TABLENAME>	用 Interact 将数据装入到的数据库表名替换此标记。
<USER>	此标记将替换为从当前流程图连接到数据源的数据库用户。

缺省值

未定义缺省值。

LoaderControlFileTemplateForAppend

描述

LoaderControlFileTemplateForAppend 属性指定先前已在 Interact 中配置的控制文件模板的完整路径和文件名。设置此参数时，Interact 会根据在此处指定的模板动态构建临时控制文件。此临时控制文件的路径和名称可用于 <CONTROLFILE> 标记（该标记可用于 LoaderCommandForAppend 属性）。

在数据库装入器实用程序方式下使用 Interact 之前，必须配置此参数指定的控制文件模板。该控制文件模板支持以下标记，当 Interact 创建临时控制文件时，会动态替换这些标记。

有关控制文件所需的正确语法，请参阅数据库装入器实用程序文档。可用于控制文件模板的标记与可用于 LoaderControlFileTemplate 属性的标记相同。

缺省情况下未定义此参数。

缺省值

未定义缺省值。

LoaderDelimiterForAppend

描述

LoaderDelimiterForAppend 属性指定 Interact 临时数据文件是固定宽度平面文件还是定界的平面文件，如果是定界的，那么还指定用作定界符的字符或字符集。

如果未定义值，那么 Interact 会将临时数据文件创建为固定宽度平面文件。

如果指定一个值，那么当调用装入器来填充未知为空的表时，将使用该值。Interact 会将临时数据文件创建为定界平面文件，并使用此属性的值作为定界符。

缺省情况下未定义此属性。

缺省值

有效值

字符（如果需要，可以括在双引号中）。

LoaderDelimiterAtEndForAppend

描述

一些外部装入实用程序需要数据文件是定界的，每行以定界符结尾。为满足此要求，请将 LoaderDelimiterAtEndForAppend 值设置为 TRUE，以便在调用装入器来填充未知为空的表时，Interact 在每一行的结尾处使用定界符。

缺省值

FALSE

有效值

TRUE | FALSE

LoaderUseLocaleDP

描述

LoaderUseLocaleDP 属性指定在 Interact 向数据库装入实用程序将要装入的文件写入数字值时，是否使用特定于语言环境的符号作为小数点。

将此值设置为 FALSE 可指定句点 (.) 将用作小数点。

将该值设置为 TRUE 可指定使用适合语言环境的小数点符号。

缺省值

FALSE

有效值

TRUE | FALSE

Interact | general | testRunDataSource

这些配置属性定义 Interact 设计环境的测试运行表的数据源设置。至少必须在其中一个运行时环境中定义此数据源。它们是执行交互式流程图的测试运行时使用的表。

jndiName

描述

使用此 jndiName 属性识别在应用程序服务器（Websphere 或 WebLogic）中为在执行交互式流程图测试运行时由设计环境访问的客户表定义的 Java 命名和目录接口 (JNDI) 数据源。

缺省值

未定义缺省值。

type

描述

在执行交互式流程图测试运行时由设计环境访问的客户表的数据库类型。

缺省值

SQLServer

有效值

SQLServer | DB2 | ORACLE

aliasPrefix

描述

AliasPrefix 属性指定 Interact 形成别名的方式，该别名在 Interact 使用维度表并写入在执行交互式流程图测试运行时由设计环境访问的客户表中的新表时自动创建。

请注意，每个数据库都具有最大标识长度；请检查要使用的数据库的文档，以确保您设置的值不会超过数据库的最大标识长度。

缺省值

A

connectionRetryPeriod

描述

ConnectionRetryPeriod 属性指定当测试运行表失败时，Interact 自动重试数据库连接请求的时间长度（以秒计）。在报告数据库错误或失败之前，在此时间长度内，Interact 会自动尝试重新连接至数据库。如果将值设置为 0，那么 Interact 会重试无限多次；如果将值设置为 -1，那么不会重试。

缺省值

-1

connectionRetryDelay

描述

ConnectionRetryDelay 属性指定在测试运行表失败之后，Interact 在尝试重新连接至数据库之前等待的时间长度（以秒计）。如果将值设置为 -1，那么不会重试。

缺省值

-1

模式

描述

包含交互式流程图测试运行的表的模式名称。Interact 会在所有表名前面插入此属性的值，例如，UACI_IntChannel 将变成 schema.UACI_IntChannel。

不必定义模式。如果不定义模式，那么 Interact 会假设表的所有者与模式的所有者相同。应该设置此值以消除歧义。

缺省值

未定义缺省值。

Interact | general | contactAndResponseHistoryDataSource

这些配置属性定义 Interact 跨会话响应跟踪所需联系人和响应历史记录数据源的连接设置。这些设置与联系人和响应历史记录模块不相关。

jndiName

描述

使用此 jndiName 属性来识别在应用程序服务器（WebSphere 或 WebLogic）中为 Interact 跨会话响应跟踪所需的联系和响应历史记录数据源定义的 Java 命名和目录接口 (JNDI) 数据源。

缺省值

type

描述

Interact 跨会话响应跟踪所需的联系人和响应历史记录数据源所用数据源的数据库类型。

缺省值

SQLServer

有效值

SQLServer | DB2 | ORACLE

connectionRetryPeriod

描述

ConnectionRetryPeriod 属性指定在 Interact 跨会话响应跟踪失败时，Interact 自动重试数据库连接请求的时间长度（以秒计）。在报告数据库错误或失败之前，在此时间长度内，Interact 会自动尝试重新连接至数据库。如果将值设置为 0，那么 Interact 会重试无限多次；如果将值设置为 -1，那么不会重试。

缺省值

-1

connectionRetryDelay

描述

ConnectionRetryDelay 属性指定在 Interact 跨会话响应跟踪失败之后，Interact 在尝试重新连接至数据库之前等待的时间长度（以秒计）。如果将值设置为 -1，那么不会重试。

缺省值

-1

模式

描述

包含 Interact 跨会话响应跟踪的表的模式名称。Interact 会在所有表名前面插入此属性的值，例如，UACI_IntChannel 将变成 schema.UACI_IntChannel。

不必定义模式。如果不定义模式，那么 Interact 会假设表的所有者与模式的所有者相同。应该设置此值以消除歧义。

缺省值

未定义缺省值。

Interact | general | idsByType

这些配置属性定义由联系人和响应历史记录模块使用的标识数的设置。

initialValue

描述

在使用 UACI_IDsByType 表生成标识时使用的初始标识值。

缺省值

1

有效值

大于 0 的任何值。

重试次数

描述

在使用 UACI_IDsByType 表生成标识时，在生成异常之前的重试次数。

缺省值

20

有效值

大于 0 的任何整数。

Interact | flowchart

本部分定义交互式流程图的配置设置。

defaultDateFormat

描述

由 Interact 用于进行日期到字符串及字符串到日期转换的缺省日期格式。

缺省值

yy/MM/dd

idleFlowchartThreadTimeoutInMinutes

描述

在释放专用于交互式流程图的线程之前，Interact 允许该线程处于空闲状态的分钟数。

缺省值

5

idleProcessBoxThreadTimeoutInMinutes

描述

在释放专用于交互式流程图进程的线程之前，Interact 允许该线程处于空闲状态的分钟数。

缺省值

5

maxSizeOfFlowchartEngineInboundQueue

描述

Interact 在队列中保留的流程图运行请求的最大数目。如果达到此请求数，那么 Interact 将会停止接收请求。

缺省值

1000

maxNumberOfFlowchartThreads

描述

专用于交互式流程图请求的线程的最大数目。

缺省值

25

maxNumberOfProcessBoxThreads

描述

专用于交互式流程图进程的线程的最大数目。

缺省值

50

maxNumberOfProcessBoxThreadsPerFlowchart

描述

专用于每个流程图实例的交互式流程图进程的线程的最大数目。

缺省值

3

minNumberOfFlowchartThreads

描述

专用于交互式流程图请求的线程的最小数目。

缺省值

10

minNumberOfProcessBoxThreads

描述

专用于交互式流程图进程的线程的最小数目。

缺省值

20

sessionVarPrefix

描述

会话变量的前缀。

缺省值

SessionVar

Interact | flowchart | ExternalCallouts | [ExternalCalloutName]

此部分定义您已使用外部调出 API 编写的定制外部调出的类设置。

class

描述

此外部调出表示的 Java 类的名称。

这是可通过 IBM 宏 EXTERNALCALLOUT 访问的 Java 类。

缺省值

未定义缺省值。

classpath

描述

此外部调出表示的 Java 类的类路径。类路径必须引用运行时环境服务器上的 JAR 文件。如果您正在使用服务器组并且所有运行时服务器都正在使用相同的 Marketing Platform，那么每个服务器都必须在相同的位置具有一个 JAR 文件副本。类路径必须包含 JAR 文件的绝对位置，用运行时环境服务器的操作系统的路径定界符分隔，例如，在 Windows 系统上使用分号 (;)，在 UNIX 系统上使用冒号 (:)。不接受包含类文件的目录。例如，在 UNIX 系统上：/path1/file1.jar:/path2/file2.jar。

此类路径必须少于 1024 个字符。可以在 .jar 文件中使用清单文件来指定其他 .jar 文件，因此，只有一个 .jar 文件必须出现在类路径中

这是可通过 IBM 宏 EXTERNALCALLOUT 访问的 Java 类。

缺省值

未定义缺省值。

Interact | flowchart | ExternalCallouts | [ExternalCalloutName] | Parameter Data | [parameterName]

此部分定义您已使用外部调出 API 编写的定制外部调出的参数设置。

值

描述

外部调出的类需要的任何参数的值。

缺省值

未定义缺省值。

示例

如果外部调出需要外部服务器的主机名，请创建名为 host 的参数类别并将 value 属性定义为服务器名称。

Interact | monitoring

此配置属性集可让您定义 JMX 监视设置。仅当您要使用 JMX 监视时，才需要配置这些属性。对于 Interact 设计环境的配置属性中的联系人和响应历史记录模块，要定义几个独立的 JMX 监视属性。

协议

描述

定义 Interact 消息传递服务的协议。

如果选择 JMXMP，那么必须在类路径中按顺序包括下列 JAR 文件：

`Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar`

缺省值

JMXMP

有效值

JMXMP | RMI

port

描述

消息传递服务的端口号。

缺省值

9998

enableSecurity

描述

一个布尔值，用于为 Interact 运行时服务器启用或禁用 JMXMP 消息传递服务安全性。如果设置为 `true`，那么必须提供用户名和密码才能访问 Interact 运行时 JMX 服务。此用户凭证由运行时服务器的 Marketing Platform 认证。Jconsole 不允许空密码登录。

如果协议是 RMI，那么此属性没有任何效果。对于 Campaign（Interact 设计时）上的 JMX，此属性没有任何效果。

缺省值

True

有效值

True | False

Interact | monitoring | activitySubscribers

此组配置属性用于为与远程订户相关的设置启用根节点，这些远程订户可接收 Interact 运行时环境中定期更新的基本绩效数据。

heartbeatPeriodInSecs

描述

每个运行时实例向订户发送更新的时间间隔（秒）。

缺省值

60

Interact | monitoring | activitySubscribers | (target)

(target)

描述

订户的设置的根节点。

URL

描述

此订户的 URL。此端点必须可以接受通过 HTTP 传输的 JSON 消息。

continuousErrorsForAbort

描述

运行时实例停止向此订户发送更多更新之前连续失败更新数。

缺省值

5

timeoutInMillis

描述

向此订户发送更新期间发送过程超时的时间（毫秒）。

缺省值

1000

有效值

已启用

描述

此订户是已启用还是已禁用。

缺省值

True

有效值

True 或 False

type

描述

此数据存储器的类型。选择此选项时，必须添加参数 **className**，且值为此实现类的标准名称。如果 JAR 文件的 URI 不在 Interact 运行时的类路径中，那么需要针对 **classPath** 添加此 URI。

缺省值

InteractLog

有效值

InteractLog、RelationalDB 和 Custom

jmxInclusionCycles

描述

向此订户发送详细 JMX 统计信息的 **heartbeatPeriodInSecs** 的乘数中的时间间隔。

缺省值

5

有效值

Interact | profile

此配置属性集控制几个可选的商品处理功能，包括商品禁止和分数覆盖。

enableScoreOverrideLookup

描述

如果设置为 True，那么 Interact 会在创建会话时从 scoreOverrideTable 装入分数覆盖数据。如果设置为 False，那么 Interact 不会在创建会话时装入市场营销分数覆盖数据。

如果设置为 true，那么还必须配置 Interact | profile | Audience Levels | (Audience Level) | scoreOverrideTable 属性。您只需要定义所需受众级别的 scoreOverrideTable 属性。让受众级别的 scoreOverrideTable 保留为空白会禁用该受众级别的分数覆盖表。

缺省值

False

有效值

True | False

enableOfferSuppressionLookup

描述

如果设置为 True，那么 Interact 会在创建会话时从 offerSuppressionTable 装入商品禁止数据。如果设置为 False，那么 Interact 不会在创建会话时装入商品禁止数据。

如果设置为 true，那么还必须配置 Interact | profile | Audience Levels | (Audience Level) | offerSuppressionTable 属性。您只需要定义所需受众级别的 enableOfferSuppressionLookup 属性。

缺省值

False

有效值

True | False

enableProfileLookup

描述

在 Interact 的新安装中，建议不要使用此属性。在 Interact 的升级安装中，此属性在第一次部署之前有效。

在交互式流程图中使用但未在交互式渠道中映射的装入行为。如果设置为 True，那么 Interact 会在创建会话时从 profileTable 装入概要文件数据。

如果设置为 true，那么还必须配置 Interact | profile | Audience Levels | (Audience Level) | profileTable 属性。

交互式渠道表映射向导中的**访问会话开始时将此数据装入到内存中**设置会覆盖此配置属性。

缺省值

False

有效值

True | False

defaultOfferUpdatePollPeriod

描述

在从缺省商品表更新高速缓存中的缺省商品之前，系统等待的秒数。如果设置为 -1，那么在运行时服务器启动时，将初始列表装入到高速缓存中之后，系统不会更新高速缓存中的缺省商品。

缺省值

-1

Interact | profile | Audience Levels | [AudienceLevelName]

此配置属性集可让您定义其他 Interact 功能需要的表名。如果您使用相关联的功能部件，那么只要求您定义表名。

新类别名称

描述

受众级别的名称。

scoreOverrideTable

描述

表的名称，此表包含此受众级别的分数覆盖信息。如果已将 enableScoreOverrideLookup 设置为 true，那么此属性适用。对于要为其启用分数覆盖表的受众级别，必须定义此属性。如果没有为此受众级别提供分数覆盖表，那么可以不定义此属性，即使 enableScoreOverrideLookup 设置为 true。

Interact 会在由 Interact 运行时服务器访问、由 prodUserDataSource 属性定义的客户表中查找此表。

如果已定义此数据源的 schema 属性，那么 Interact 会在此表名前面添加模式，例如，schema.UACI_ScoreOverride。如果您输入标准名称，例如，mySchema.UACI_ScoreOverride，那么 Interact 不会前置模式名称。

缺省值

UACI_ScoreOverride

offerSuppressionTable

描述

表的名称，此表包含此受众级别的商品禁止信息。对于要为其启用商品禁止表的受众级别，必须定义此属性。如果没有为此受众级别提供商品禁止表，那么可以不定义此属性。如果 enableOfferSuppressionLookup 设置为 true，那么必须将此属性设置为有效表。

Interact 会在由运行时服务器访问、由 prodUserDataSource 属性定义的客户表中查找此表。

缺省值

UACI_BlackList

contactHistoryTable

描述

此受众级别的联系人历史记录数据的登台表名。

此表存储在运行时环境表中 (systemTablesDataSource)。

如果已定义此数据源的 schema 属性，那么 Interact 会在此表名前面添加模式，例如，schema.UACI_CHStaging。如果您输入标准名称，例如，mySchema.UACI_CHStaging，那么 Interact 不会前置模式名称。

如果已禁用联系历史记录日志记录，那么不需要设置此属性。

缺省值

UACI_CHStaging

chOfferAttribTable

描述

此受众级别的联系历史记录商品属性表的名称。

此表存储在运行时环境表中 (systemTablesDataSource)。

如果已定义此数据源的 schema 属性，那么 Interact 会在此表名前面添加模式，例如，schema.UACI_CHOfferAttrib。如果您输入标准名称，例如，mySchema.UACI_CHOfferAttrib，那么 Interact 不会前置模式名称。

如果已禁用联系历史记录日志记录，那么不需要设置此属性。

缺省值

UACI_CHOfferAttrib

responseHistoryTable

描述

此受众级别的响应历史记录登台表的名称。

此表存储在运行时环境表中 (systemTablesDataSource)。

如果已定义此数据源的 schema 属性，那么 Interact 会在此表名前面添加模式，例如，schema.UACI_RHStaging。如果您输入标准名称，例如，mySchema.UACI_RHStaging，那么 Interact 不会前置模式名称。

如果已禁用响应历史记录日志记录，那么不需要设置此属性。

缺省值

UACI_RHStaging

crossSessionResponseTable

描述

此受众级别的表的名称，此表是响应跟踪功能可访问的联系和响应历史记录表中跨会话响应跟踪所必需。

如果已定义此数据源的 schema 属性，那么 Interact 会在此表名前面添加模式，例如，schema.UACI_XSessResponse。如果您输入标准名称，例如，mySchema.UACI_XSessResponse，那么 Interact 不会前置模式名称。

如果已禁用跨会话响应日志记录，那么不需要设置此属性。

缺省值

UACI_XSessResponse

userEventLoggingTable

描述

这是用于记录用户定义的事件活动的数据库表的名称。Interact 界面中"交互式渠道摘要"页面中"事件"选项卡上用户定义的事件。您在此处所指定的数据库表将存储诸如事件标识、名称以及自最近一次刷新事件活动高速缓存之后此受众级别发生此事件的次数等信息。

如果已定义此数据源的 schema 属性，那么 Interact 会在此表名前面添加模式，例如，schema.UACI_UserEventActivity。如果您输入标准名称，例如，mySchema.UACI_UserEventActivity，那么 Interact 不会前置模式名称。

缺省值

UACI_UserEventActivity

patternStateTable

描述

这是用于记录事件模式状态（例如，是否满足模式条件以及模式是否已过期或禁用等）的数据库表的名称。

如果已定义此数据源的 schema 属性，那么 Interact 会在此表名前面添加模式，例如，schema.UACI_EventPatternState。如果您输入标准名称，例如，mySchema.UACI_EventPatternState，那么 Interact 不会前置模式名称。

即使您不使用事件模式，每个受众级别也需要 patternStateTable。patternStateTable 基于所包括的 UACI_EventPatternState 的 DDL。以下是一个示例，其中受众标识包含两个部分：ComponentNum 和 ComponentStr。

```
CREATE TABLE UACI_EventPatternState_Composite
(
   UpdateTime bigint NOT NULL,
   State varbinary(4000),
   ComponentNum bigint NOT NULL,
   ComponentStr nvarchar(50) NOT NULL,
   CONSTRAINT PK_CustomerPatternState_Composite PRIMARY KEY
   (ComponentNum,ComponentStr,UpdateTime)
)
```

缺省值

UACI_EventPatternState

Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL

此配置属性集可让您定义其他 Interact 功能需要的表名。如果您使用相关联的功能部件，那么只要求您定义表名。

enableOffersByRawSQL

描述

如果设置为 True，那么 Interact 将为此受众级别启用 offersBySQL 功能，此功能使您可以配置要执行的 SQL 代码以在运行时创建一组所需的候选商品。如果是 False，那么 Interact 不使用 offersBySQL 功能。

如果将此属性设置为 true，那么可能还要配置 Interact | Interact | profile | Audience Levels | (Audience Level) | Offers by Raw SQL | SQL Template 属性以定义一个或多个 SQL 模板。

缺省值

False

有效值

True | False

cacheSize

描述

用于存储 OfferBySQL 查询结果的高速缓存的大小。请注意，如果查询结果对于大部分会话是唯一的，那么使用高速缓存可能具有负面影响。

缺省值

-1（关闭）

有效值

-1 | 值

cacheLifeInMinutes

描述

如果启用了高速缓存，那么这指示在系统清除高速缓存（以避免过时）之前的分钟数。

缺省值

-1（关闭）

有效值

-1 | 值

defaultSQLTemplate

描述

要使用的 SQL 模板的名称（如果未通过 API 调用来指定名称）。

缺省值

无

有效值

SQL 模板名称

name

配置类别

Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL | (SQL Templates)

描述

要分配给此 SQL 查询模板的名称。输入一个在将此 SQL 模板用于 API 调用中时有效的描述性名称。请注意，如果此处使用的名称与 offerBySQL 处理的 Interact List 流程框中定义的名称相同，那么将使用流程框中的 SQL，而非您在此处输入的 SQL。

缺省值

无

SQL

配置类别

Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL | (SQL Templates)

描述

包含要由此模板调用的 SQL 查询。SQL 查询可能包含对访问者的会话数据（概要文件）中部分变量名称的引用。例如，select * from MyOffers where category = ${preferredCategory} 将依赖于包含名为 preferredCategory 的变量的会话。

应将 SQL 配置为查询您在设计时期间创建的供此功能使用的特定商品表。请注意，此处不支持存储过程。

缺省值

无

Interact | profile | Audience Levels | [AudienceLevelName] | SQL Template

这些配置属性使您可以定义由 Interact 的 offersBySQL 功能所使用的一个或多个 SQL 查询模板。

name

描述

要分配给此 SQL 查询模板的名称。输入一个在将此 SQL 模板用于 API 调用中时有效的描述性名称。请注意，如果此处使用的名称与 offerBySQL 处理的 Interact List 流程框中定义的名称相同，那么将使用流程框中的 SQL，而非您在此处输入的 SQL。

缺省值

无

SQL

描述

包含要由此模板调用的 SQL 查询。SQL 查询可能包含对访问者的会话数据（概要文件）中部分变量名称的引用。例如，`select * from MyOffers where category = ${preferredCategory}` 将依赖于包含名为 `preferredCategory` 的变量的会话。

应将 SQL 配置为查询您在设计时期间创建的供此功能使用的特定商品表。请注意，此处不支持存储过程。

缺省值

无

Interact | profile | Audience Levels | [AudienceLevelName | Profile Data Services | [DataSource]

此配置属性集可让您定义其他 Interact 功能需要的表名。如果您使用相关联的功能部件，那么只要求您定义表名。Profile Data Services 类别提供了有关针对所有受众级别创建的内置数据源（也称为数据库）的信息，并且该数据源的优先级预配置为 100。但是，您可以选择对其进行修改或禁用。此类别还包含其他外部数据源的模板。单击名为**外部数据服务**的模板时，您可以完成此处描述的配置设置。

新类别名称

描述

（不适用于缺省"数据库"条目。）您正在定义的数据源的名称。您在此处输入的名称在相同受众级别的数据源中必须唯一。

缺省值

无

有效值

不允许任何文本字符串。

enabled

描述

如果设置为 `True`，那么会针对其分配给的受众级别启用此数据源。如果设置为 `False`，那么 Interact 不会使用此受众级别的数据源。

缺省值

True

有效值

True | False

className

描述

（不适用于缺省"数据库"条目。）用于实现 IInteractProfileDataService 的数据源类的标准名称。

缺省值

无。

有效值

提供标准类名的字符串。

classPath

描述

（不适用于缺省"数据库"条目。）提供用于装入此数据源实现类的路径的可选配置设置。如果您忽略该配置设置，那么缺省情况下，会使用所包含应用程序服务器的类路径。

缺省值

未显示，但是如果此处未提供任何值，那么缺省情况下，会使用所包含应用程序服务器的类路径。

有效值

提供类路径的字符串。

优先级

描述

此受众级别内该数据源的优先级。它必须在每个受众级别的所有数据源中是唯一值。（即，如果某个数据源的优先级设置为 100，那么此受众级别内没有任何其他数据源的优先级可能是 100。）

缺省值

对于缺省数据库，为 100，对于用户定义的数据源，为 200

有效值

不允许任何非负整数。

Interact | offerserving

这些配置属性定义类属学习配置属性。如果您正在使用内置学习来要调整学习实现，请使用设计环境的配置属性。

offerTieBreakMethod

描述

offerTieBreakMethod 属性定义两个商品的分数相等（相关）时商品呈现的行为。如果您将此属性设置为缺省值 Random，那么 Interact 将从具有相等分数的商品中呈现随机选项。如果您将此配置设置为更新的商品，那么当商品之间的分数相同时，Interact 将在更旧的商品（商品标识更低）之前提供更新的商品（如果具有更高的商品标识）。

注：

Interact 具有可选功能，允许管理员通过设置 percentRandomSelection 选项 (Campaign | partitions | [partition_number] | Interact | learning | percentRandomSelection) 来配置系统以按随机顺序（与分数无关）返回商品。仅当 percentRandomSelection 设置为 0（已禁用）时，才使用此处描述的 offerTieBreakMethod 属性。

缺省值

Random

有效值

Random | Newer Offer

optimizationType

描述

optimizationType 属性定义 Interact 是否使用学习引擎来协助商品分配。如果设置为 NoLearning，那么 Interact 不会使用学习。如果设置为 BuiltInLearning，那么 Interact 会使用通过 Interact 构建的 Bayesian 学习引擎。如果设置为 ExternalLearning，那么 Interact 会使用您提供的学习引擎。如果选择 ExternalLearning，那么您必须定义 externalLearningClass 和 externalLearningClassPath 属性。

缺省值

NoLearning

有效值

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

描述

运行时服务器在获取商品之前等待交互式流程图完成的最大毫秒数。

缺省值

5000

treatmentCodePrefix

描述

添加至处理码前面的前缀。

缺省值

未定义缺省值。

effectiveDateBehavior

描述

确定 Interact 是否应该使用商品的生效日期来过滤对访问者显示的商品。值包括：

- -1 告知 Interact 忽略商品的生效日期。

  0 告知 Interact 使用生效日期来过滤商品，以便在商品生效日期早于或等于当前日期时将商品提供给访问者。

  如果设置了 **effectiveDateGracePeriod** 值，那么还会应用宽限期以确定是否提供商品。
- 任何正整数告知 Interact 使用当前日期加上此属性的值来确定是否向访问者提供商品，以便在商品生效日期早于当前日期加此属性值时向访问者提供商品。

  如果设置了 **effectiveDateGracePeriod** 值，那么还会应用宽限期以确定是否提供商品。

缺省值

-1

effectiveDateGracePeriodOfferAttr

描述

在商品定义中指定用于指示生效日期宽限期的定制属性的名称。例如，您可以将此属性的值配置为 AltGracePeriod。然后，您可使用称为 AltGracePeriod 的定制属性来定义商品，该定制属性用于指定与 **effectiveDateBehavior** 属性配合使用的宽限期天数。

假定您创建一个新商品模板，其生效日期距离当前日期 10 天并且包括一个称为 AltGracePeriod 的定制属性。使用该模板创建商品时，如果将 AltGracePeriod 的值设置为 14 天，那么会向访问者提供商品，因为当前日期在商品有效日期的宽限期内。

缺省值

空

alwaysLogLearningAttributes

描述

指示 Interact 是否应将有关学习模块使用的访问者属性的信息写至日志文件。请注意，将此值设置为 true 可能会影响学习性能和日志文件大小。

缺省值

False

Interact | offerserving | Built-in Learning Config

这些配置属性定义内置学习的数据库写入设置。要调整学习实现，请使用设计环境的配置属性。

version

描述

您可以选择 1 或 2。版本 1 是基本配置版本，它不会使用参数来设置线程和记录限制。版本 2 是增强型配置版本，它允许您设置线程和记录参数以提高性能。达到这些参数限制时，这些参数会执行聚集和删除。

缺省值

1

insertRawStatsIntervalInMinutes

描述

Interact 学习模块在学习登台表中插入更多行之前等待的分钟数。可能需要根据学习模块正在环境中处理的数据量修改此时间。

缺省值

5

有效值

正整数

aggregateStatsIntervalInMinutes

描述

Interact 学习模块每次在学习登台表中聚集数据之后再次聚集数据之前等待的分钟数。可能需要根据学习模块正在环境中处理的数据量修改此时间。

缺省值

15

有效值

大于零的整数。

autoAdjustPercentage

描述

确定聚集运行尝试处理的数据百分比的值，它基于先前运行的度量值。缺省情况下，此值设置为零，这表示聚集器将处理所有登台记录，并且禁用此自动调整功能。

缺省值

0

有效值

0 与 100 之间的数字。

enableObservationModeOnly

描述

如果设置为 True，那么会启用学习方式，在该方式下 Interact 收集用于学习的数据，但不会将该数据用于建议或商品仲裁。这使您能够在启动方式下进行自学，直到您确定已为建议收集到足够的数据为止。

缺省值

False

有效值

True | False

excludeAbnormalAttribute

描述

确定是否要将那些属性标记为无效的设置。如果设置为 IncludeAttribute，那么将包括异常属性，而不会将它们标记为无效。如果设置为 ExcludeAttribute，那么将排除异常属性，并且会将它们标记为无效。

缺省值

IncludeAttribute

有效值

IncludeAttribute | ExcludeAttribute

Interact | offerserving | Built-in Learning Config | Parameter Data | [parameterName]

这些配置属性定义外部学习模块的任何参数。

numberOfThreads

描述

学习聚合器用来处理数据的线程最大数目。有效值是正整数，并且不应该超过在学习数据源中配置的最大连接数。此参数仅由聚集器版本 2 使用。

缺省值

10

maxLogTimeSpanInMin

描述

如果选择了聚集器版本 1，那么您可以采用迭代方式处理登台记录，以避免出现过大的数据库批次。在这种情况下，这些登台记录按区块进行处理；在单个聚集周期中反复迭代。此参数的值指定每次迭代中聚集器尝试处理的登台记录的最大持续时间。此时间范围基于与每个登台记录相关联的 LogTime 字段，并且仅处理其 LogTime 在最早的时间窗口内的记录。有效值是非负整数。如果值为 0，那么没有任何限制，这表示将在单次迭代中处理所有登台记录。

缺省值

0

maxRecords

描述

如果选择了聚集器版本 2，那么您可以采用迭代方式处理登台记录，以避免出现过大的数据库批次。在这种情况下，这些登台记录按区块进行处理；在单个聚

集周期中反复迭代。此参数的值指定每次迭代中聚集器尝试处理的最大登台记录数。有效值是非负整数。如果值为 0，那么没有任何限制，这表示将在单次迭代中处理所有登台记录。

缺省值

0

值

描述

内置学习模块的类所需的任何参数的值。

缺省值

未定义缺省值。

Interact | offerserving | External Learning Config

这些配置属性定义您使用学习 API 编写的外部学习模块的类设置。

class

描述

如果将 optimizationType 设置为 ExternalLearning，请将 externalLearningClass 设置为外部学习引擎的类名。

缺省值

未定义缺省值。

可用性

仅当将 optimizationType 设置为 ExternalLearning 时，此属性才适用。

classPath

描述

如果将 optimizationType 设置为 ExternalLearning，请将 externalLearningClass 设置为外部学习引擎的类路径。

类路径必须引用运行时环境服务器上的 JAR 文件。如果您正在使用服务器组并且所有运行时服务器都正在使用相同的 Marketing Platform，那么每个服务器都必须在相同的位置具有一个 JAR 文件副本。类路径必须包含 JAR 文件的绝对位置，用运行时环境服务器的操作系统的路径定界符分隔，例如，在 Windows 系统上使用分号 (;)，在 UNIX 系统上使用冒号 (:)。不接受包含类文件的目录。例如，在 UNIX 系统上：/path1/file1.jar:/path2/file2.jar。

此类路径必须少于 1024 个字符。可以在 .jar 文件中使用清单文件来指定其他 .jar 文件，因此，只有一个 .jar 文件必须出现在类路径中

缺省值

未定义缺省值。

可用性

仅当将 optimizationType 设置为 ExternalLearning 时，此属性才适用。

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

这些配置属性定义外部学习模块的任何参数。

值

描述

外部学习模块的类需要的任何参数的值。

缺省值

未定义缺省值。

示例

如果外部学习模块需要算法求解程序应用程序的路径，那么您应创建名为 `solverPath` 的参数类别并将 `value` 属性定义为该应用程序的路径。

Interact | offerserving | Constraints

这些配置属性定义针对商品服务过程设置的约束。

maxOfferAllocationInMemoryPerInstance

描述

商品块的大小。Interact 将商品池保留在内存中，从而在每次退回商品时，系统都不必查询数据库。每次退回商品时，都会调整池。池耗尽时，Interact 会获取另一个商品块以填充池。

缺省值

1000

有效值

大于 0 的整数。

maxDistributionPerIntervalPerInstanceFactor

描述

支持在运行时服务器中进行分发的运行时服务器给定商品分配的约束百分比。

缺省值

100

有效值

0 到 100 之间的整数。

constraintCleanupIntervalInDays

描述

清理 UACI_OfferCount 表中禁用计数的频率。值小于 1 会禁用此功能。

缺省值

7

有效值

大于 0 的整数。

Interact | services

此类别中的配置属性定义管理收集联系和响应历史记录数据及统计信息以报告和写入运行时环境系统表的所有服务的设置。

externalLoaderStagingDirectory

描述

此属性定义数据库装入实用程序的登台目录的位置。

缺省值

未定义缺省值。

有效值

相对于 Interact 安装目录的路径或登台目录的绝对路径。

如果启用数据库装入实用程序，那么必须将 contactHist 和 responstHist 类别中的 cacheType 属性设置为 External Loader File。

Interact | services | contactHist

此类别中的配置属性定义服务（此服务收集联系历史记录登台表的数据）的设置。

enableLog

描述

如果是 true，那么会启用服务来收集用于记录联系历史记录数据的数据。如果是 false，那么不会收集数据。

缺省值

True

有效值

True | False

cacheType

描述

定义为联系历史记录收集的数据是保留在内存 (Memory Cache) 还是文件 (External Loader file) 中。仅当您已将 Interact 配置为使用数据库装入器实用程序时，才能使用 External Loader file。

如果选择 Memory Cache，请使用 cache 类别设置。如果选择 External Loader File，请使用 fileCache 类别设置。

缺省值

Memory Cache

有效值

Memory Cache | External Loader File

Interact | services | contactHist | cache

此类别中的配置属性定义服务（此服务收集联系人历史记录登台表的数据）的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的联系人历史记录数据写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | contactHist | fileCache

此类别中的配置属性定义您要使用数据库装入程序实用程序时收集联系人历史记录数据的服务的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的联系人历史记录数据写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | defaultedStats

此类别中的配置属性定义服务（此服务收集有关使用交互点缺省字符串的次数的统计信息）的设置。

enableLog

描述

如果是 true，那么启用服务（此服务收集有关用于 UACI_DefaultedStat 表的交互点缺省字符串的次数的统计信息）。如果是 false，那么不会收集缺省字符串统计信息。

如果不使用 IBM 报告，那么可以将此属性设置为 false，因为不需要进行数据收集。

缺省值

True

有效值

True | False

Interact | services | defaultedStats | cache

此类别中的配置属性定义服务（此服务收集有关使用交互点缺省字符串的次数的统计信息）的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的缺省字符串统计信息写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | eligOpsStats

此类别中的配置属性定义服务（此服务写合格商品的统计信息）的设置。

enableLog

描述

如果是 true，那么会启用收集合格商品的统计信息的服务。如果是 false，那么不会收集合格商品统计信息。

如果不使用 IBM 报告，那么可以将此属性设置为 false，因为不需要进行数据收集。

缺省值

True

有效值

True | False

Interact | services | eligOpsStats | cache

此类别中的配置属性定义服务（此服务收集合格商品统计信息）的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的合格商品统计信息写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | eventActivity

此类别中的配置属性定义服务（此服务收集事件活动统计信息）的设置。

enableLog

描述

如果是 true，那么会启用收集事件活动统计信息的服务。如果是 false，那么不会收集事件统计信息。

如果不使用 IBM 报告，那么可以将此属性设置为 false，因为不需要进行数据收集。

缺省值

True

有效值

True | False

Interact | services | eventActivity | cache

此类别中的配置属性定义服务（此服务收集事件活动统计信息）的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的事件活动统计信息写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | eventPattern

eventPattern 类别中的配置属性定义服务（此服务收集事件模式活动统计信息）的设置。

persistUnknownUserStates

描述

确定数据库中是否已保留未知受众标识（访问者）的事件模式状态。缺省情况下，当会话结束时，如果受众标识为已知（即，可在概要文件数据源中找到访问者的概要文件），那么与访问者受众标识相关联的所有已更新事件模式的状态都存储在数据库中。

persistUnknownUserStates 属性确定受众标识为未知的情况下发生的情况。缺省情况下，此属性设置为 False，并且针对未知的受众标识，事件模式状态将在会话结束时被废弃。

如果您将此属性设置为 True，那么将持久存储未知用户（无法在已配置的概要文件数据服务中找到其概要文件）的事件模式状态。

缺省值

False

有效值

True | False

mergeUnknowUserInSessionStates

描述

确定如何保留未知受众标识（访问者）的事件模式状态。如果受众标识在会话过程中切换，那么 Interact 将尝试从数据库表装入新受众标识的已保存事件模式状态。当先前受众标识未知并且您将 mergeUnknowUserInSessionStates 属性设置为 True 时，属于同一会话中先前受众标识的用户事件活动将合并到新的受众标识。

缺省值

False

有效值

True | False

enableUserEventLog

描述

确定数据库中是否已记录用户事件活动。

缺省值

False

有效值

True | False

Interact | services | eventPattern | userEventCache

userEventCache 类别中的配置属性用于定义确定何时移动高速缓存中的事件活动以持久存储在数据库中的设置。

阈值（threshold）

描述

确定可存储在事件模式状态高速缓存中的最大事件模式状态数目。当达到该限制时，将从高速缓存中除去最近使用频率最低的状态。

缺省值

100

有效值

要保留在高速缓存中的所需事件模式状态的数目。

insertPeriodInSecs

描述

确定内存中已排队的用户事件活动的最大事件长度（以秒计）。当达到此属性所指定的时间限制时，会将这些活动持久存储到数据库中。

缺省值

3600（60 分钟）

有效值

所需秒数。

Interact | services | eventPattern | advancedPatterns

此类别中的配置属性控制是否启用与 Interact Advanced Patterns 的集成，并且它们定义用于与 Interact Advanced Patterns 的连接的超时时间间隔。

enableAdvancedPatterns

描述

如果为 true，那么会启用与 Interact Advanced Patterns 的集成。如果为 false，那么不会启用集成。如果先前启用了集成，那么 Interact 会使用从 Interact Advanced Patterns 接收到的最新模式状态。

缺省值

True

有效值

True | False

connectionTimeoutInMilliseconds

描述

从 Interact 实时环境到 Interact Advanced Patterns 进行 HTTP 连接所花的最长时间。如果请求超时，那么 Interact 会使用模式中最近一次保存的数据。

缺省值

30

readTimeoutInMilliseconds

描述

在 Interact 实时环境与 Interact Advanced Patterns 之间建立 HTTP 连接并且将请求发送到 Interact Advanced Patterns 以获取事件模式状态之后，可用于接收数据的最长时间。如果请求超时，那么 Interact 会使用模式中最近一次保存的数据。

缺省值

100

connectionPoolSize

描述

用于 Interact 实时环境与 Interact Advanced Patterns 之间通信的 HTTP 连接池的大小。

缺省值

10

Interact | services | eventPattern | advancedPatterns | autoReconnect

此类别中的配置属性指定与 Interact Advanced Patterns 的集成中的自动重新连接功能的参数。

启用

描述

如果 Interact 实时环境与 Interact Advanced Patterns 之间发生连接问题，确定系统是否自动重新连接。缺省值为 **True** 启用此功能。

缺省值

True

有效值

True | False

durationInMinutes

描述

此属性指定时间间隔（以分钟计），系统将在该时间间隔内评估 Interact 实时环境和 Interact Advanced Patterns 之间的重复连接问题。

缺省值

10

numberOfFailuresBeforeDisconnect

描述

此属性指定系统与 Interact Advanced Patterns 自动断开连接之前指定时间段内所允许的连接失败次数。

缺省值

3

consecutiveFailuresBeforeDisconnect

描述

确定自动重新连接功能是否仅评估 Interact 实时环境与 Interact Advanced Patterns 之间的连续连接失败次数。如果将此值设置为 **False**，那么会评估指定时间间隔内的所有失败。

缺省值

True

sleepBeforeReconnectDurationInMinutes

描述

系统在断开连接（由于发生了此类别中其他属性中指定的重复失败次数）之后，重新连接之前将等待此属性中指定的分钟数。

缺省值

5

sendNotificationAfterDisconnect

描述

此属性确定系统在发生连接失败的情况下是否发送电子邮件通知。通知消息包括所发生的连接失败的 Interact 实时实例名称以及在重新连接之前的时间长度（如 **sleepBeforeReconnectDurationInMinutes** 属性中所指定）。缺省值为 **True** 表示将发送通知。

缺省值

True

Interact | services | customLogger

此类别中的配置属性定义服务（此服务收集定制数据以写入表（使用 UACICustomLoggerTableName 事件参数的事件））的设置。

enableLog

描述

如果是 true，那么启用定制记录到表功能。如果是 false，那么 UACICustomLoggerTableName 事件参数没有任何效果。

缺省值

True

有效值

True | False

Interact | services | customLogger | cache

此类别中的配置属性定义服务（此服务将定制数据收集到表中（使用 UACICustomLoggerTableName 事件参数的事件））的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的定制数据写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | responseHist

此类别中的配置属性定义服务（此服务向响应历史记录登台表写入数据）的设置。

enableLog

描述

如果是 true，那么将启用向响应历史记录登台表写入数据的服务。如果是 false，那么不会向响应历史记录登台表中写入数据。

响应历史记录登台表由受众级别的 responseHistoryTable 属性来定义。缺省值是 UACI_RHStaging。

缺省值

True

有效值

True | False

cacheType

描述

定义高速缓存是保留在内存中还是文件中。仅当您已将 Interact 配置为使用数据库装入器实用程序时，才能使用 External Loader file。

如果选择 Memory Cache，请使用 cache 类别设置。如果选择 External Loader File，请使用 fileCache 类别设置。

缺省值

Memory Cache

有效值

Memory Cache | External Loader File

actionOnOrphan

描述

此设置确定如何处理没有对应联系事件的响应事件。如果设置为 NoAction，那么将像发布了对应联系事件一样处理响应事件。如果设置为 Warning，那么将像发布了对应联系事件一样处理响应事件，但是会将一条警告消息写入 interact.log。如果设置为 Skip，那么不会处理响应事件，并且会将一条错误消息写入 interact.log。无论是否启用了响应历史记录日志记录，您在此处选择的设置都会生效。

缺省值

NoAction

有效值

NoAction | Warning | Skip

Interact | services | responseHist | cache

此类别中的配置属性定义服务（此服务收集响应历史记录数据）的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的响应历史记录数据写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | response Hist | responseTypeCodes

此类别中的配置属性定义响应历史记录服务的设置。

新类别名称

描述 响应类型代码的名称。

代码

描述

响应类型的定制代码。

缺省值

UA_UsrResponseType 表中添加的定制节点。

操作

描述

与定制响应类型代码对应的操作。

为发布的事件定义的操作将覆盖此处定义的操作。因此，如果未使用 responseTypeCode 发布 logAccept 事件，那么会将此事件视为接受事件。如果使用此配置中存在的 responseTypeCode 发布 logAccept 事件，那么会使用配置的操作确定其是否为接受事件。如果使用此配置中不存在的 responseTypeCode 发布 logAccept 事件，那么不会将此事件视为接受事件。将事件视为接受事件时，如果启用了学习，那么会相应更新学习统计信息。如果具有一个商品表达式规则基于此商品的接受情况，那么会评估这些规则。

缺省值

无

有效值

LogAccept | LogReject | None

Interact | services | responseHist | fileCache

此类别中的配置属性定义要使用数据库装入程序实用程序时收集响应历史记录数据的服务的高速缓存设置。

阈值（threshold）

描述

在 Interact 将记录写入数据库之前累积的记录数。

responseHist - 由受众级别的 responseHistoryTable 属性定义的表。缺省值是 UACI_RHStaging。

缺省值

100

insertPeriodInSecs

描述

强制写入数据库之间的秒数。

缺省值

3600

Interact | services | crossSessionResponse

此类别中的配置属性定义 crossSessionResponse 服务和 xsession 进程的常规设置。仅当您要使用 Interact 跨会话响应跟踪时，才需要配置这些设置。

enableLog

描述

如果是 true，那么启用 crossSessionResponse 服务并允许 Interact 将数据写入跨会话响应跟踪登台表。如果是 false，那么会禁用 crossSessionResponse 服务。

缺省值

False

xsessionProcessIntervalInSecs

描述

运行 xsession 进程之间的秒数。此进程会将数据从跨会话响应跟踪登台表移至响应历史记录登台表和内置学习模块。

缺省值

180

有效值

大于零的整数

purgeOrphanResponseThresholdInMinutes

描述

在标记与联系人和响应历史记录表中的联系人不匹配的任何响应之前，crossSessionResponse 服务等待的分钟数。

如果响应在联系人和响应历史记录表中没有匹配项，那么在 purgeOrphanResponseThresholdInMinutes 分钟后，Interact 会在 xSessResponse 登台表 Mark 列中用值 -1 标记该响应。然后可以手动匹配或删除这些响应。

缺省值

180

Interact | services | crossSessionResponse | cache

此类别中的配置属性定义服务（此服务收集跨会话响应数据）的高速缓存设置。

阈值（threshold）

描述

在 flushCacheToDB 服务将收集的跨会话响应数据写入数据库之前累积的记录数。

缺省值

100

insertPeriodInSecs

描述

强制写入 XSessResponse 表之间的秒数。

缺省值

3600

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

此部分中的属性定义跨会话响应跟踪如何使处理码与联系人和响应历史记录匹配。

SQL

描述

此属性定义 Interact 是否使用在 OverrideSQL 属性中定义的系统生成的 SQL 或定制 SQL。

缺省值

Use System Generated SQL

有效值

Use System Generated SQL | Override SQL

OverrideSQL

描述

如果不使用缺省 SQL 命令来将处理码与联系人和响应历史记录相匹配，请在此处输入 SQL 或存储过程。

如果将 SQL 设置为 Use System Generated SQL，那么会忽略此值。

缺省值

useStoredProcedure

描述

如果设置为 true，那么 OverrideSQL 必须包含对存储过程（其将处理码与联系人和响应历史记录相匹配）的引用。

如果设置为 false，那么 OverrideSQL（如果使用）必须是 SQL 查询。

缺省值

false

有效值

true | false

类型

描述

在运行时环境表内 UACI_TrackingType 表中定义的相关联的 TrackingCodeType。除非您修改 UACI_TrackingType 表，否则 Type 必须是 1。

缺省值

1

有效值

在 UACI_TrackingType 表中定义的整数。

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

此部分中的属性定义跨会话响应跟踪如何使商品代码与联系人和响应历史记录匹配。

SQL

描述

此属性定义 Interact 是否使用在 OverrideSQL 属性中定义的系统生成的 SQL 或定制 SQL。

缺省值

Use System Generated SQL

有效值

Use System Generated SQL | Override SQL

OverrideSQL

描述

如果不使用缺省 SQL 命令来将商品代码与联系人和响应历史记录相匹配，请在此处输入 SQL 或存储过程。

如果将 SQL 设置为 Use System Generated SQL，那么会忽略此值。

缺省值

useStoredProcedure

描述

如果设置为 True，那么 OverrideSQL 必须包含对存储过程（其将商品代码与联系人和响应历史记录相匹配）的引用。

如果设置为 false，那么 OverrideSQL（如果使用）必须是 SQL 查询。

缺省值

false

有效值

true | false

类型

描述

在运行时环境表内 UACI_TrackingType 表中定义的相关联的 TrackingCodeType。除非您修改 UACI_TrackingType 表，否则 Type 必须是 2。

缺省值

2

有效值

在 UACI_TrackingType 表中定义的整数。

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

此部分中的属性定义跨会话响应跟踪如何使用户定义的备用代码与联系和响应历史记录匹配。

名称

描述

此属性定义备用代码的名称。此名称必须与运行时环境表的 UACI_TrackingType 表中的 Name 值匹配。

缺省值

OverrideSQL

描述

SQL 命令或存储过程，用于通过商品代码或处理码将备用代码与联系人和响应历史记录相匹配。

缺省值

useStoredProcedure

描述

如果设置为 true，那么 OverrideSQL 必须包含对存储过程（其将备用代码与联系人和响应历史记录相匹配）的引用。

如果设置为 false，那么 OverrideSQL（如果使用）必须是 SQL 查询。

缺省值

false

有效值

true | false

类型

描述

在运行时环境表内 UACI_TrackingType 表中定义的相关联的 TrackingCodeType。

缺省值

3

有效值

在 UACI_TrackingType 表中定义的整数。

Interact | services | threadManagement | contactAndResponseHist

此类别中的配置属性定义服务（这些服务收集联系人和响应历史记录登台表的数据）的线程管理设置。

corePoolSize

描述

要保留在池中用于收集联系人和响应历史记录数据的线程数（即使它们是空闲的）。

缺省值

5

maxPoolSize

描述

要保留在池中用于收集联系人和响应历史记录数据的线程的最大数目。

缺省值

5

keepAliveTimeSecs

描述

当线程数大于核心时，这是在结束收集联系人和响应历史记录数据之前，过量的空闲线程等待新任务的最长时间。

缺省值

5

queueCapacity

描述

由用于收集联系人和响应历史记录数据的线程池使用的队列大小。

缺省值

1000

termWaitSecs

描述

在运行时服务器关闭时，这是等待服务线程完成收集联系人和响应历史记录数据的秒数。

缺省值

5

Interact | services | threadManagement | allOtherServices

此类别中的配置属性定义收集商品合格性统计信息、事件活动统计信息、缺省字符串使用情况统计信息和定制记录到表数据的服务的线程管理设置。

corePoolSize

描述

对于收集商品合格性统计信息、事件活动统计信息、缺省字符串使用情况统计信息和定制记录到表数据的服务，要保留在池中的线程数（即使它们是空闲的）。

缺省值

5

maxPoolSize

描述

对于收集商品合格性统计信息、事件活动统计信息、缺省字符串使用情况统计信息和定制记录到表数据的服务，要保留在池中的线程的最大数目。

缺省值

5

keepAliveTimeSecs

描述

当线程数大于核心时，这是在终止收集商品合格性统计信息、事件活动统计信息、缺省字符串使用情况统计信息和定制记录到表数据的服务之前，过量的空闲线程等待新任务的最长时间。

缺省值

5

queueCapacity

描述

服务的线程池使用的队列大小（这些服务收集商品合格性统计信息、事件活动统计信息、缺省字符串使用情况统计信息和定制记录到表数据）。

缺省值

1000

termWaitSecs

描述

在运行时服务器关闭时，这是等待服务（这些服务收集商品合格性统计信息、事件活动统计信息、缺省字符串使用情况统计信息和定制记录到表数据）的服务线程完成的秒数。

缺省值

5

Interact | services | threadManagement | flushCacheToDB

此类别中的配置属性定义将高速缓存中的已收集数据写入运行时环境数据库表的线程的线程管理设置。

corePoolSize

描述

对于将高速缓存的数据写入数据存储的已调度线程，要保留在池中的线程数。

缺省值

5

maxPoolSize

描述

对于将高速缓存的数据写入数据存储的已调度线程，要保留在池中的线程的最大数目。

缺省值

5

keepAliveTimeSecs

描述

当线程数大于核心时，这是在终止将高速缓存的数据写入数据存储的已调度线程之前，过量的空闲线程等待新任务的最长时间。

缺省值

5

queueCapacity

描述

对于已高速缓存数据写入数据存储的已调度线程，线程池使用的队列大小。

缺省值

1000

termWaitSecs

描述

在运行时服务器关闭时，这是等待服务线程完成将已高速缓存数据写入数据存储的已调度线程的秒数。

缺省值

5

Interact | services | threadManagement | eventHandling

此类别中的配置属性定义服务（这些服务收集用于处理事件的数据）的线程管理设置。

corePoolSize

描述

要保留在池中用于收集事件处理数据的线程数（即使它们是空闲的）。

缺省值

1

maxPoolSize

描述

对于用于收集事件处理数据的服务，要保留在池中的最大线程数。

缺省值

5

keepAliveTimeSecs

描述

当线程数大于核心时，这是在结束收集事件处理数据之前，过量的空闲线程等待新任务的最长时间。

缺省值

5

queueCapacity

描述

由用于收集事件处理数据的线程池使用的队列大小。

缺省值

1000

termWaitSecs

描述

在运行时服务器关闭时，这是等待针对用于收集事件处理数据的服务完成服务线程的秒数。

缺省值

5

Interact | services | configurationMonitor

此类别中的配置属性允许您启用或禁用与 Interact Advanced Patterns 的集成，而不必重新启动 Interact 实时环境，并且它们定义用于轮询启用该集成的属性值的时间间隔。

启用

描述

如果为 `true`，启用刷新 **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns** 属性值的服务。如果为 `false`，在更改 **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns** 属性的值时您必须重新启动 Interact 实时。

缺省值

False

有效值

True | False

refreshIntervalInMinutes

描述

定义轮询 **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns** 属性值的时间间隔。

缺省值

5

Interact | cacheManagement

此配置属性集合定义设置，用于选择和配置可用来提高 Interact 的性能的受支持高速缓存管理器（例如，构建在 Interact 安装中的 Ehcache、作为可选附加组件的 WebSphere eXtreme Scale 高速缓存或者其他外部高速缓存系统）。

使用 **Interact | cacheManagement | Cache Managers** 配置属性来配置要使用的高速缓存管理器。使用 **Interact | cacheManagement | caches** 配置属性来指定 Interact 应该用来提高性能的高速缓存管理器。

Interact | cacheManagement | Cache Managers

Cache Managers 类别指定您计划与 Interact 配合使用的高速缓存管理解决方案的参数。

Interact | cacheManagement | Cache Managers | EHCache

EHCache 类别指定 EHCache 高速缓存管理解决方案的参数，以便您可对其进行定制以提高 Interact 的性能。

Interact | Cache Managers | EHCache | Parameter Data

此类别中的配置属性控制 EHCache 高速缓存管理系统如何工作以提高 Interact 的性能。

cacheType

描述

可配置服务器组中的 Interact 运行时服务器以使用多点广播地址来共享高速缓存数据。这称为分布式高速缓存。cacheType 参数指定是在**本地**（独立）方式还是**分布式**（与运行时服务器组一样）下使用内置 EHCache 高速缓存机制。

注：

如果您选择**分布式**作为 cacheType，那么共享高速缓存的所有服务器都必须是同一个服务器组的组成部分。还必须启用多点广播才能在服务器组的所有成员之间工作。

缺省值

Local

有效值

Local | Distributed

multicastIPAddress

描述

如果您指定 **cacheType** 参数为"分布式"，那么要将高速缓存配置为通过多点广播在 Interact 运行时服务器组的所有成员之间运行。multicastIPAddress 值是服务器组的所有 Interact 服务器用于侦听的 IP 地址。

该 IP 地址必须在服务器组之间唯一。

缺省值

230.0.0.1

multicastPort

描述

如果您指定 **cacheType** 参数为"分布式"，那么 **multicastPort** 参数用于指示服务器组的所有 Interact 服务器用于侦听的端口。

缺省值

6363

overflowToDisk

描述

EHCache 高速缓存管理器使用可用内存来管理会话信息。对于大型概要文件导致会话大小很大的环境，内存中支持的会话数可能不足以支持客户场景。遇到这种情况时，EHCache 具有一项可选功能可允许大小超过内存存储能力的高速缓存信息改为临时写入硬盘驱动器中。

如果您将 **overflowToDisk** 属性设置为"yes"，那么每台 Java 虚拟机 (JVM) 能够处理的并行会话比内存单独所允许处理的更多。

缺省值

否

有效值

No | Yes

diskStore

描述

当 **overflowToDisk** 配置属性设置为 Yes 时，此配置属性指定磁盘目录，该目录将存放从内存中溢出的高速缓存条目。如果此配置属性不存在或者其值无效，那么将在操作系统的缺省临时目录中自动创建磁盘目录。

缺省值

无

有效值

一个目录，托管 Interact 运行时的 Web 应用程序对该目录具有写特权。

(Parameter)

描述

可用来创建要与高速缓存管理器配合使用的定制参数的模板。可设置任何参数名称以及它必须具有的值。

要创建定制参数，请单击 ***(Parameter)*** 并填写要分配给该参数的名称和值。当您单击**保存更改**时，会将您所创建的参数添加到"参数数据"类别中的列表。

缺省值

无

Interact | cacheManagement | Cache Managers | Extreme Scale

Extreme Scale 类别指定供适配器使用 WebSphere eXtreme Scale 高速缓存管理解决方案的参数，以便您可对其进行定制以提高 Interact 的性能。

ClassName

描述

将 Interact 连接到 WebSphere eXtreme Scale 服务器的类的标准名称。该名称必须是 `com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`。

缺省值

`com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`

ClassPath

描述

文件 `interact_wxs_adapter.jar` 所在位置的 URI，例如，`file:///IBM/IMS/Interact/lib/interact_wxs_adapter.jar` 或 `file:///C:/IBM/IMS/Interact/lib/interact_wxs_adapter.jar`。但是，如果此 JAR 文件已经包括在托管应用程序服务器的类路径中，那么应该将此字段保留为空白。

缺省值

空

Interact | Cache Managers | Extreme Scale | Parameter Data

此类别中的配置属性控制可选择性地随 Interact 安装一起提供的 WebSphere eXtreme Scale 适配器。必须将充当客户机的每台 Interact 运行时服务器的这些设置配置为 eXtreme Scale 服务器网格。

catalogPropertyFile

描述

用于启动 WebSphere eXtreme Scale 目录服务器的属性文件所在的位置的 URI。如果 Extreme Scale 适配器用于启动目录服务器，那么必须设置此属性。否则，不会使用该属性。

缺省值

`file:///C:/depot/Interact/dev/main/extremescale/config/catalogServer.props`

containerPropertyFile

描述

用于启动 WebSphere eXtreme Scale 容器实例的属性文件所在的位置的 URI。如果所包括的服务器组件用于启动 WebSphere eXtreme Scale 容器服务器，那么必须设置此属性。否则，不会使用该属性。

缺省值

```
file:///C:/depot/Interact/dev/main/extremescale/config/
containerServer.props
```

deploymentPolicyFile

描述

用于启动 WebSphere eXtreme Scale 目录服务器的部署策略文件所在的位置的 URI。如果所包括的服务器组件用于启动 WebSphere eXtreme Scale 目录服务器，那么必须设置此属性。否则，不会使用该属性。

缺省值

```
file:///C:/depot/Interact/dev/main/extremescale/config/
deployment.xml
```

objectGridConfigFile

描述

用于启动 WebSphere eXtreme Scale 目录服务器以及与 Interact 运行时服务器在同一 Java 虚拟机 (JVM) 中运行的近高速缓存组件的对象网格配置文件所在的位置的 URI。

缺省值

```
file:///C:/depot/Interact/dev/main/extremescale/config/
objectgrid.xml
```

gridName

描述

保存所有 Interact 高速缓存的 WebSphere eXtreme Scale 网格的名称。

缺省值

InteractGrid

catalogURLs

描述

一个 URL，包含主机名或 IP 地址以及 WebSphere eXtreme Scale 目录服务器在侦听连接时使用的端口。

缺省值

无

(Parameter)

描述

可用来创建要与高速缓存管理器配合使用的定制参数的模板。可设置任何参数名称以及它必须具有的值。

要创建定制参数，请单击 ***(Parameter)*** 并填写要分配给该参数的名称和值。当您单击**保存更改**时，会将您所创建的参数添加到"参数数据"类别中的列表。

缺省值

无

Interact | caches

使用此配置属性集合来指定您要用于提高 Interact 的性能的受支持高速缓存管理器（例如，Ehcache 或 WebSphere eXtreme Scale 高速缓存），并且为您要配置的运行时服务器配置特定高速缓存属性。

这包括用于存储会话数据、事件模式状态和细分结果的高速缓存。通过调整这些设置，您可以指定用于每种高速缓存的高速缓存解决方案，并且可指定用于控制高速缓存工作方式的个别设置。

Interact | cacheManagement | caches | InteractCache

InteractCache 类别为所有会话对象配置高速缓存，其中包括概要文件数据、细分结果、最近交付的处理、通过 API 方法传递的参数以及 Interact 运行时所使用的其他对象。

Interact 需要 InteractCache 类别才能正常工作。

对于 **Interact | cacheManagement | Caches** 中不支持的设置，还可以通过外部 EHCache 配置来配置 InteractCache 类别。如果使用 EHCache，那么必须确保正确配置 InteractCache。

CacheManagerName

描述

处理 Interact 高速缓存的高速缓存管理器的名称。您在此处输入的值必须是在 **Interact | cacheManagement | Cache Managers** 配置属性中定义的其中一个高速缓存管理器，例如 `EHCache` 或 `Extreme Scale`。

缺省值

EHCache

有效值

在 **Interact | cacheManagement | Cache Managers** 配置属性中定义的任何高速缓存管理器。

maxEntriesInCache

描述

要存储在此高速缓存中的最大会话数据对象数。当达到最大会话数据对象数并且需要存储更多会话的数据时，会删除最近最少使用的对象。

缺省值

100000

有效值

大于 0 的整数。

timeoutInSecs

描述

自使用或更新会话数据对象以来经过的时间（以秒计），用于确定从高速缓存移除该对象的时间。

注：如果已从 9.1 之前的版本升级，那么将需要重新配置 timeoutInSecs 属性，因为此属性已移动。

缺省值

300

有效值

大于 0 的整数。

Interact | Caches | Interact Cache | Parameter Data

此类别中的配置属性控制 Interact 安装自动使用的 Interact 高速缓存。必须分别为每台 Interact 运行时服务器配置这些设置。

asyncIntervalMillis

描述

高速缓存管理器 EHCache 在其将任何更改复制到其他 Interact 运行时实例之前应等待的时间（以毫秒计）。如果该值不是正数，那么会同步复制这些更改。

缺省情况下，不会创建此配置属性。如果您创建了此属性，那么仅当 EHCache 为高速缓存管理器并且 ehCache **cacheType** 属性设置为 distributed 时才使用此属性。

缺省值

无。

(Parameter)

描述

可用来创建要与 Intearct 高速缓存配合使用的定制参数的模板。可设置任何参数名称以及它必须具有的值。

要创建定制参数，请单击 ***(Parameter)*** 并填写要分配给该参数的名称和值。当您单击**保存更改**时，会将您所创建的参数添加到"参数数据"类别中的列表。

缺省值

无

Interact | cacheManagement | caches | PatternStateCache

PatternStateCache 类别用于托管事件模式和实时商品禁止规则的状态。缺省情况下，会将此高速缓存配置为直读和直写高速缓存，以便 Interact 尝试首先对事件模式和商品禁止数据使用高速缓存。如果所请求的条目在高速缓存中不存在，那么高速缓存实现将通过 JNDI 配置或者直接使用 JDBC 连接从数据源装入该条目。

要使用 JNDI 连接，Interact 会连接到现有数据源提供程序，该数据源提供程序已通过所指定服务器使用 JNDI 名称、URL 等等定义。对于 JDBC 连接，必须提供一组包括 JDBC 驱动程序类名、数据库 URL 和认证信息的 JDBC 设置。

请注意，如果您定义多个 JNDI 和 JDBC 源，那么会使用第一个启用的 JNDI 源；如果未启用任何 JNDI 源，那么会使用第一个启用的 JDBC 源。

Interact 需要 PatternStateCache 类别才能正常工作。

对于 **Interact | cacheManagement | Caches** 中不支持的设置，还可以通过外部 EHCache 配置来配置 PatternStateCache 类别。如果使用 EHCache，那么必须确保正确配置 PatternStateCache。

CacheManagerName

描述

处理 Interact 模式状态高速缓存的高速缓存管理器的名称。您在此处输入的值必须是在 **Interact | cacheManagement | Cache Managers** 配置属性中定义的其中一个高速缓存管理器，例如 `EHCache` 或 `Extreme Scale`。

缺省值

EHCache

有效值

在 **Interact | cacheManagement | Cache Managers** 配置属性中定义的任何高速缓存管理器。

maxEntriesInCache

描述

要存储在此高速缓存中的最大事件模式状态数。当达到最大事件模式状态数并且需要存储更多事件模式状态的数据时，会删除最近最少使用的对象。

缺省值

100000

有效值

大于 0 的整数。

timeoutInSecs

描述

指定事件模式状态对象在事件模式状态高速缓存中超时的时间量（以秒计）。当此类状态对象在高速缓存中持续处于空闲状态的时间达到 `timeoutInSecs` 秒时，会根据 least-recently-used 规则将它从高速缓存中逐出。请注意，此属性的值应该比 `sessionTimeoutInSecs` 属性中定义的更大。

注：如果已从 9.1 之前的版本升级，那么将需要重新配置 `timeoutInSecs` 属性，因为此属性已移动。

缺省值

300

有效值

大于 0 的整数。

***Interact | Caches | PatternStateCache | Parameter Data*:**

此类别中的配置属性控制用于托管事件模式和实时商品禁止规则的状态的"模式状态高速缓存"。

(Parameter)

描述

可用来创建要与"模式状态高速缓存"配合使用的定制参数的模板。可设置任何参数名称以及它必须具有的值。

要创建定制参数，请单击 ***(Parameter)*** 并填写要分配给该参数的名称和值。当您单击**保存更改**时，会将您所创建的参数添加到"参数数据"类别中的列表。

缺省值

无

***Interact | cacheManagement | caches | PatternStateCache | loaderWriter*:**

loaderWriter 类别包含与外部存储库交互的装入程序的配置以检索和持久存储事件模式。

className

描述

此装入程序的标准类名。此类必须符合所选高速缓存管理器的要求。

缺省值

`com.unicacorp.interact.cache.ehcache.loaderwriter.PatternStateEHCacheLoaderWriter`

有效值

标准类名。

classPath

描述

装入程序的类文件的路径。如果您将此值保留为空白或所输入的内容无效，那么会使用用于运行 Interact 的类路径。

缺省值

无

有效值

有效类路径。

writeMode

描述

指定写程序的模式，以持久存储高速缓存中的新建或更新的事件模式状态。有效选项为：

- WRITE_THROUGH。每当存在新条目或者更新现有条目时，都会立即将该条目写入存储库。
- WRITE_BEHIND。高速缓存管理器等待一段时间以收集大量更改，然后将这些更改以批处理方式持久存储到存储库。

缺省值

WRITE_THROUGH

有效值

WRITE_THROUGH 或 WRITE_BEHIND。

batchSize

描述

写程序将以批处理方式持久存储的最大事件模式状态对象数。仅当 **writeMode** 设置为 WRITE_BEHIND 时，才会使用此属性。

缺省值

100

有效值

整数值。

maxDelayInSecs

描述

高速缓存管理器在持久存储事件模式状态对象之前等待的最长时间（以秒计）。仅当 **writeMode** 设置为 WRITE_BEHIND 时，才会使用此属性。

缺省值

5

有效值

整数值。

Interact | *Caches* | *PatternStateCache* | *loaderWriter* | *Parameter Data*：

此类别中的配置属性控制"模式状态高速缓存"装入程序。

(Parameter)

描述

可用来创建要与"模式状态高速缓存"装入程序配合使用的定制参数的模板。可设置任何参数名称以及它必须具有的值。

要创建定制参数，请单击 ***(Parameter)*** 并填写要分配给该参数的名称和值。当您单击**保存更改**时，会将您所创建的参数添加到"参数数据"类别中的列表。

缺省值

无

Interact | *cacheManagement* | *caches* | *PatternStateCache* | *loaderWriter* | *jndiSettings*:

jndiSettings 类别包含装入程序将用来与支持数据库进行通信的 JNDI 数据源的配置。要创建新的 JNDI 设置集合，请展开 **jdniSettings** 类别并单击 ***(jndiSetting)*** 属性。

(jndiSettings)

注：使用 WebSphere Application Server 时，loaderWriter 不会与 **jndiSettings** 连接。

描述

当单击此类别时，会出现一个表单。要定义 JNDI 数据源，请填写以下值：

- **新类别名称**，要用于识别此 JNDI 连接的名称。
- **已启用**，让您指示是否希望此 JNDI 连接可供使用。将新连接的此值设置为 `True`。
- **jdniName**，设置数据源时已定义的数据源中的 JNDI 名称。
- **providerUrl**，用于查找此 JNDI 数据源的 URL。如果您将此字段保留为空白，那么会使用托管 Interact 运行时的 Web 应用程序的 URL。
- **初始上下文工厂**，用于连接到 JNDI 提供程序的初始上下文工厂类的标准类名。如果托管 Interact 运行时的 Web 应用程序用作 **providerUrl**，那么会将此字段保留为空白。

缺省值

无。

Interact | *cacheManagement* | *caches* | *PatternStateCache* | *loaderWriter* | *jdbcSettings*:

jdbcSettings 类别包含装入程序将用来与支持数据库进行通信的 JDBC 连接的配置。要创建新的 JDBC 设置集合，请展开 **jdbcSettings** 类别并单击 ***(jdbcSetting)*** 属性。

(jdbcSettings)

描述

当单击此类别时，会出现一个表单。要定义 JDBC 数据源，请填写以下值：

- **新类别名称**，要用于识别此 JDBC 连接的名称。
- **已启用**，让您指示是否希望此 JDBC 连接可供使用。将新连接的此值设置为 `True`。
- **driverClassName**，JDBC 驱动程序的标准类名。此类必须存在于为启动托管高速缓存服务器而配置的类路径中。
- **databaseUrl**，用于查找此 JDBC 数据源的 URL。
- **asmUser**，IBM Marketing Software 用户的名称，该名称已通过用于连接到此 JDBC 连接中的数据库的凭证进行配置。
- **asmDataSource**，IBM Marketing Software 数据源的名称，该名称已通过用于连接到此 JDBC 连接中的数据库的凭证配置。
- **maxConnection**，允许对此 JDBC 连接中的数据库进行的最大并发连接数。

缺省值

无。

Interact | triggeredMessage

此类别中的配置属性定义所有触发式消息和商品渠道交付的设置。

backendProcessIntervalMin

描述

此属性定义后端线程装入和处理延迟的商品交付的时间段（以分钟计）。该值必须为整数。如果值为 0 或负数，那么表示后端进程处于禁用状态。

有效值

正整数

autoLogContactAfterDelivery

描述

如果此属性设置为 true，那么将在分派此商品或者此商品已排队进行延迟交付之后立即自动发布一个联系事件。如果此属性设置为 false，那么不会针对出站商品自动发布联系事件。这是缺省行为。

注：

- 当触发出站消息时，如果您想要在联系历史记录中捕获其他属性，那么可以在联系历史记录中的列中添加其他定制属性。发布将触发出站消息的事件时，您可以将 `postEvent` 方法中的属性值作为名称值参数来进行传递
- 要将商品参数化到出站渠道，您可以分配相关联类别中的商品，部署渠道，将商品参数化，然后在触发的消息中选择**自动选择下一个最佳商品**。

有效值

True | False

waitForFlowchart

描述

此属性确定流程图是否应该等待当前正在运行的细分市场完成，以及超出等待时间之后流程图的行为。

DoNotWait：无论细分市场当前是否处于运行状态，都将开始处理触发式的消息。但是，如果在合格规则和/或选择为商品选择方法的 NextBestOffer 中使用了细分市场，那么 TM 执行将仍然等待。

OptionalWait：在当前处于运行状态的细分市场完成或发生超时之前，触发式消息的处理过程将一直等待。如果等待超时，那么将记录警告，并且将继续处理此触发式消息。这是缺省设置。

MandatoryWait：在当前处于运行状态的细分市场完成或发生超时之前，触发式消息的处理过程将一直等待。如果等待超时，那么将记录错误，并且此触发式消息的处理将异常中止。

有效值

DoNotWait | OptionalWait | MandatoryWait

Interact | triggeredMessage | offerSelection

此类别中的配置属性定义触发式消息中商品选择的设置。

maxCandidateOffers

描述

此属性定义引擎为了获得最佳商品以进行交付而返回的合格商品的最大数目。可能无法根据所选渠道发送返回的这些合格商品中的任何商品。候选商品越多，这种情况越少发生。但是，更多候选商品可能会增加处理时间。

有效值

正整数

defaultCellCode

描述

如果交付的商品是评估战略规则或表驱动记录的结果，那么将存在与该商品相关联的目标单元，并且在所有相关日志记录中都将使用此单元的信息。但是，如果使用特定商品的列表作为商品选择的输入，那么不会提供目标单元。在这种情况下，将使用此配置设置的值。必须确保在部署中包含此目标单元及其营销活动。要实现此目的，最简单的方法是将此单元添加到已部署策略中。

Interact | triggeredMessage | dispatchers

此类别中的配置属性定义触发式消息中所有分派器的设置。

dispatchingThreads

描述

此属性定义引擎用来以异步方式调用分派器的线程数。如果值为 0 或负数，那么表示分派器调用是同步调用。缺省值为 0。

有效值

整数

Interact | triggeredMessage | dispatchers | <dispatcherName>

此类别中的配置属性定义触发式消息中特定分派器的设置。

category name

描述

此属性定义此分派器的名称。所有分派器的名称都必须唯一。

type

描述

此属性定义分派器类型。

有效值

InMemoryQueue | JMSQueue | Custom

注：如果您使用 JMSQueue 或 Custom 将 Interact 与 IBM MQ 集成，那么 Interact 运行时必须位于安装了 JDK 1.7 的应用程序服务器上。对于 WebSphere 和 WebLogic，建议您使用提供的最新 JDK 修订包版本。

JMSQueue 仅支持 WebLogic。如果您使用 WebSphere Application Server，那么无法使用 JMSQueue。

className

描述

此属性定义此分派器实现的标准类名。如果类型为 InMemoryQueue，那么该值应该为空。如果类型为 custom，那么此设置的值必须为 `com.unicacorp.interact.eventhandler.triggeredmessage.dispatchers.IBMMQDispatcher`。

classPath

描述

此属性定义 JAR 文件的 URL，此文件包含此分派器的实现。

如果类型为 custom，那么此设置的值必须为 `file://<Interact_HOME>/lib/interact_ibmmqdispatcher.jar;file://<Interact_HOME>/lib/com.ibm.mq.allclient.jar;file://<Interact_HOME>/lib/jms.jar`

Interact | triggeredMessage | dispatchers | <dispatcherName> | Parameter Data

此类别中的配置属性定义触发式消息中特定分派器的参数。

您可以在 3 种类型的分派器之间进行选择。InMemoryQueue 是 Interact 的内部分派器。Custom 用于 IBM MQ。JMSQueue 用于通过 JNDI 连接到 JMS 提供程序。

category name

描述

此属性定义此参数的名称。此名称在该分派器的所有参数中必须唯一。

值

描述

此属性以"名称/值"对格式定义此分派器需要的参数。

注：触发器消息的所有参数都区分大小写，并且应该按此处显示的方式输入。

如果类型为 InMemoryQueue，那么支持以下参数。

- queueCapacity：可选。这是可以在队列中等待进行分派的最大商品数。如果指定了此属性，那么它必须是正整数。如果未指定此属性或者此属性无效，那么将使用缺省值 (1000)。

如果类型为 Custom，那么支持以下参数。

- providerUrl：<hostname>:port（区分大小写）
- queueManager：在 IBM MQ 服务器上创建的队列管理器的名称。
- messageQueueName：在 IBM MQ 服务器上创建的消息队列的名称。

- enableConsumer：此属性必须设置为 true。
- asmUserforMQAuth：用于登录服务器的用户名。如果服务器强制执行认证，那么此参数是必需的。否则，不应指定此参数。
- authDS：与用于登录服务器的用户名相关联的密码。如果服务器强制执行认证，那么此参数是必需的。否则，不应指定此参数。

如果类型为 JMSQueue，那么支持以下参数。

- providerUrl：JNDI 提供程序的 URL（区分大小写）。
- connectionFactoryJNDI：JMS 连接工厂的 JNDI 名称。
- messageQueueJNDI：JMS 队列的 JNDI 名称，系统向此队列发送并从中检索触发式消息。
- enableConsumer：指示是否应该在 Interact 中启动这些触发式消息的使用者。此属性必须设置为 true。如果未指定此属性，那么将使用缺省值 (false)。
- initialContextFactory：JNDI 初始上下文工厂类的标准名称。如果您使用 WebLogic，那么此参数的值应该为 weblogic.jndi.WLInitialContextFactory。

Interact | triggeredMessage | gateways | <gatewayName>

此类别中的配置属性定义触发式消息中特定网关的设置。

Interact 不支持同一网关的多个实例。所有网关配置文件应该可从每个 Interact 运行时节点访问。在分布式设置中，确保网关文件保存在共享位置。

category name

描述

此属性定义此网关的名称。所有网关的名称都必须唯一。

className

描述

此属性定义此网关实现的标准类名。

classPath

描述

此属性定义 JAR 文件的 URI，此文件包含此网关的实现。如果将此属性保留为空白，那么将使用托管 Interact 应用程序的类路径。

例如，在 Windows 系统中，如果网关 JAR 文件位于 C:\IBM\EMM\EmailGateway\IBM_Interact_OMO_OutboundGateway_Silverpop_1.0\lib\OMO_OutboundGateway_Silverpop.jar 目录中，那么 classPath 应该为 file:///C:/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar。在 Unix 系统中，如果网关 jar 文件位于 /opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar 目录中，那么 classpath 应该为 file:///

opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar。

Interact | triggeredMessage | gateways | <gatewayName> | Parameter Data

此类别中的配置属性定义触发式消息中特定网关的参数。

category name

描述

此属性定义此参数的名称。该网关的所有参数的名称都必须唯一。

值

描述

此属性以"名称/值"对格式定义此网关需要的参数。对于所有网关，支持以下参数。

注：触发器消息的所有参数都区分大小写，并且应该按此处显示的方式输入。

- validationTimeoutMillis：通过此网关验证商品时，发生超时的持续时间（以毫秒计）。缺省值为 500。
- deliveryTimeoutMillis：使用此网关交付商品时，发生超时的持续时间（以毫秒计）。缺省值为 1000。

Interact | triggeredMessage | channels

此类别中的配置属性定义触发式消息中所有渠道的设置。

type

描述

此属性定义与特定网关相关的设置的根节点。Default 使用内置渠道选择器，此选择器基于触发式消息 UI 中定义的渠道列表。如果选择了 default，那么应该将 className 和 classPath 值保留为空白。Custom 使用 IChannelSelector 的客户实现。

有效值

Default | Custom

className

描述

此属性定义渠道选择器的客户实现的标准类名。如果类型为 Custom，那么此设置是必需的。

classPath

描述

此属性定义 JAR 文件的 URL，此文件包含渠道选择器的客户实现。如果将此属性保留为空白，那么将使用托管 Interact 应用程序的类路径。

Interact | triggeredMessage | channels | Parameter Data

此类别中的配置属性定义触发式消息中特定渠道的参数。

category name

描述

此属性定义此参数的名称。该渠道的所有参数的名称都必须唯一。

值

描述

此属性以"名称/值"对格式定义渠道选择器需要的参数。

如果您使用**客户首选渠道**作为渠道，那么必须创建

Interact | triggeredMessage | channels | <channelName>

此类别中的配置属性定义触发式消息中特定渠道的设置。

category name

描述

此属性定义通过其发送商品的渠道的名称。它应该与 **Campaign | partitions | <partition[N]> | Interact | outboundChannels** 下的设计时中定义的那些名称匹配。

Interact | triggeredMessage | channels | <channelName> | <handlerName>

此类别中的配置属性定义触发式消息中用于发送商品的特定处理程序的设置。

category name

描述

此属性定义渠道将用来发送商品的处理程序的名称。

dispatcher

描述

此属性定义分派器的名称，此处理程序通过该分派器将商品发送到网关。它必须是 **interact | triggeredMessage | dispatchers** 下定义的其中一个分派器。

gateway

描述

此属性定义网关的名称，此处理程序最终将商品发送到该网关。它必须是 **interact | triggeredMessage | gateways** 下定义的其中一个网关。

mode

描述

此属性定义此处理程序的使用方式。如果选择了 Failover，那么仅当此渠道中定义的所有具有较高优先级的处理程序未能发送商品时，才会使用此处理程序。如果选择了 Addon，那么无论其他处理程序是否已成功发送商品，都会使用此处理程序。

priority

描述

此属性定义此处理程序的优先级。引擎首先尝试使用具有最高优先级的处理程序来发送商品。

有效值

任何整数

缺省值

100

Interact | activityOrchestrator

活动编排器类别指定 Interact 入站网关活动的接收器和网关。

使用 **Interact | activityOrchestrator | receivers** 配置属性来配置 Interact 接收器。使用 **Interact | activityOrchestrator | gateways** 配置属性来配置要在 Interact 中使用的网关。

Interact | activityOrchestrator | receivers

活动编排器接收器类别指定 Interact 入站网关活动的事件接收器。

类别名称

描述

接收器的名称。

类型

描述 接收器的类型。您可以在 IBM MQ 和定制之间进行选择。定制需要您使用 iReceiver 的实现。

已启用

描述 选择 True 以启用接收器，或选择 false 以禁用接收器。

className

描述 此属性定义此接收器实现的标准类名。此属性仅在类型为定制时使用。

classPath

描述 此属性定义 JAR 文件的 URI，此文件包含此接收器的实现。如果将此属性保留为空白，那么将使用托管 Interact 应用程序的类路径。此属性仅在类型为定制时使用。

Interact | activityOrchestrator | receivers | Parameter Data

您可以添加接收器参数（例如，queueManager 和 messageQueueNameyi）以定义接收器队列。

Interact | activityOrchestrator | gateways

活动编排器网关类别指定 Interact 入站网关活动的网关。

类别名称

描述

网关的名称。

className

描述 此属性定义此网关实现的标准类名。

classPath

描述 此属性定义 JAR 文件的 URI，此文件包含此网关的实现。如果将此属性保留为空白，那么将使用托管 Interact 应用程序的类路径。此属性仅在类型为定制时使用。

Interact | activityOrchestrator | gateways | Parameter Data

您可以为网关配置文件（例如，`OMO-conf_inbound_UBX_interactEventNameMapping` 和 `OMO-conf_inbound_UBX_interactEventPayloadMapping`）添加网关参数。

Interact | ETL | patternStateETL

此类别中的配置属性定义 ETL 过程的设置。

新类别名称

描述

提供一个可唯一识别此配置的名称。请注意，运行独立 ETL 过程时，必须提供此确切名称。为了便于在命令行中指定此名称，您可能要避免使用含有空格或标点符号的名称（例如 `ETLProfile1`）。

runOnceADay

描述

决定此配置中的独立 ETL 过程是否应该每天运行一次。有效答案为 **Yes** 或 **No**。如果您在这里的答案为 **No**，那么 **processSleepIntervalInMinutes** 会决定该过程的运行计划安排。

preferredStartTime

描述

独立 ETL 过程应该启动的首选时间。请以格式 HH:MM:SS AM/PM 来指定时间，例如 `01:00:00 AM`。

preferredEndTime

描述

独立 ETL 过程应该停止的首选时间。请以格式 HH:MM:SS AM/PM 来指定时间，例如 08:00:00 AM。

processSleepIntervalInMinutes

描述

如果尚未将独立 ETL 过程配置成每天运行一次（如 **runOnceADay** 属性中指定），那么此属性指定 ETL 过程运行之间的间隔。例如，如果在这里指定 15，那么独立 ETL 过程在停止运行之后，将等待 15 分钟，然后再次启动。

maxJDBCInsertBatchSize

描述

落实查询之前的 JDBC 批处理的记录的最大数目。缺省情况下，将此属性设置为 5000。请注意，这不是 ETL 在一次迭代中所处理的最大记录数。在每个迭代期间，ETL 会处理 UACI_EVENTPATTERNSTATE 表中所有可用的记录。但是，会将所有这些记录分为 **maxJDBCInsertSize** 个组块。

maxJDBCFetchBatchSize

描述

要从登台数据库中访存的 JDBC 批处理的记录的最大数目。

可能需要提高此值以调整 ETL 的性能。

communicationPort

描述

独立 ETL 过程用于侦听停止请求的网络端口。在常规情况下，不应更改此属性的缺省值。

queueLength

描述

用于性能调整的值。模式状态数据的集合将访存并转换成对象，这些对象会添加到要处理和写入数据库的队列中。此属性控制队列的大小。

completionNotificationScript

描述

指定 ETL 过程完成后要运行的脚本的绝对路径。如果指定脚本，那么会将三个自变量传递至完成通知脚本：开始时间、结束时间和处理的事件模式记录总数。开始时间和结束时间是表示自从 1970 年以来已经过毫秒数的数字值。

Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS

此类别中的配置属性定义 ETL 运行时 DS 的设置。

type

描述

您要定义的数据源支持的数据库类型的列表。

dsname

描述

数据源的 JNDI 名称。此名称还必须用在用户的数据源配置中，以确保该用户有权访问目标及运行时数据源。

driver

描述

要使用的 JDBC 驱动程序的名称，例如下列任何一项：

Oracle：oracle.jdbc.OracleDriver

Microsoft SQL Server： com.microsoft.sqlserver.jdbc.SQLServerDriver

IBM DB2：com.ibm.db2.jcc.DB2Driver

serverURL

描述

数据源 URL，例如下列任何一项：

Oracle：jdbc:oracle:thin:@<your_db_host>:<your_db_port>:<your_db_service_name>

Microsoft SQL Server：jdbc:sqlserver:// <your_db_host>:<your_db_port>;databaseName= <your_db_name>

IBM DB2：jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>

connectionpoolSize

描述

这是用于指示连接池大小的值，提供该值的目的是为了调整性能。根据可用的数据库连接，同时读取和转换模式状态数据。如果增加连接池大小，那么将允许更多并发数据库连接，但受内存和数据库读/写功能的限制影响。例如，如果将此值设置为 4，那么将有 4 个作业同时运行。如果具有大量数据，那么可能需要将此值增加到诸如 10 或 20 之类的数字，只要有充足的内存和足够数据库性能即可。

模式

描述

此配置正在连接至的数据库模式的名称。

connectionRetryPeriod

描述

ConnectionRetryPeriod 属性指定 Interact 在失败时自动重试数据库连接请求的时间量（以秒计）。在报告数据库错误或失败之前，在此时间长度内，Inter-

act 会自动尝试重新连接至数据库。如果将值设置为 0，那么 Interact 会重试无限多次；如果将值设置为 -1，那么不会重试。

connectionRetryDelay

描述

ConnectionRetryDelay 属性指定 Interact 在失败之后尝试重新连接至数据库之前等待的时间长度（以秒计）。如果将值设置为 -1，那么不会重试。

Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS

此类别中的配置属性定义 ETL 目标 DS 的设置。

type

描述

您要定义的数据源支持的数据库类型的列表。

dsname

描述

数据源的 JNDI 名称。此名称还必须用在用户的数据源配置中，以确保该用户有权访问目标及运行时数据源。

driver

描述

要使用的 JDBC 驱动程序的名称，例如下列任何一项：

Oracle：oracle.jdbc.OracleDriver

Microsoft SQL Server： com.microsoft.sqlserver.jdbc.SQLServerDriver

IBM DB2：com.ibm.db2.jcc.DB2Driver

serverURL

描述

数据源 URL，例如下列任何一项：

Oracle：jdbc:oracle:thin:@<your_db_host>:<your_db_port>:<your_db_service_name>

Microsoft SQL Server：jdbc:sqlserver:// <your_db_host>:<your_db_port>;databaseName= <your_db_name>

IBM DB2：jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>

connectionpoolSize

描述

这是用于指示连接池大小的值，提供该值的目的是为了调整性能。根据可用的数据库连接，同时读取和转换模式状态数据。如果增加连接池大小，那么将允许更多并发数据库连接，但受内存和数据库读/写功能的限制影响。例如，如

果将此值设置为 4，那么将有 4 个作业同时运行。如果具有大量数据，那么可能需要将此值增加到诸如 10 或 20 之类的数字，只要有充足的内存和足够数据库性能即可。

模式

描述

此配置正在连接至的数据库模式的名称。

connectionRetryPeriod

描述

ConnectionRetryPeriod 属性指定 Interact 在失败时自动重试数据库连接请求的时间量（以秒计）。在报告数据库错误或失败之前，在此时间长度内，Interact 会自动尝试重新连接至数据库。如果将值设置为 0，那么 Interact 会重试无限多次；如果将值设置为 -1，那么不会重试。

connectionRetryDelay

描述

ConnectionRetryDelay 属性指定 Interact 在失败之后尝试重新连接至数据库之前等待的时间长度（以秒计）。如果将值设置为 -1，那么不会重试。

Interact | ETL | patternStateETL | <patternStateETLName> | Report

此类别中的配置属性定义 ETL 报告聚集过程的设置。

启用

描述 启用或禁用与 ETL 的报告集成。缺省情况下，此属性设置为 disable。

如果设置为 disable，那么此属性会禁止对 UARI_DELTA_PATTERNS 表进行更新。它不会完全禁止报告。

注： 要禁用与 ETL 的报告集成，还必须修改触发器 TR_AGGREGATE_DELTA_PATTERNS，才能对 UACI_ETLPATTERNSTATERUN 登台表进行禁用。

retryAttemptsIfAggregationRunning

描述 如果设置了锁定标志，那么此属性是 ETL 尝试检查报告聚集是否已完成的次数。缺省情况下，此属性设置为 3。

sleepBeforeRetryDurationInMinutes

描述 连续尝试之间的休眠时间（以分钟计）。缺省情况下，此属性设置为 5。

aggregationRunningCheckSql

描述 此属性允许您定义定制 SQL，可以运行该 SQL 以查看是否设置了报告聚集锁定标志。缺省情况下，此属性为空。

未设置此属性时，ETL 会运行以下 SQL 来获取锁定标志。

```
select count(1) AS ACTIVERUNS from uari_pattern_lock where islock='Y'
=> If ACTIVERUNS is > 0, lock is set
```

aggregationRunningCheck

描述 允许或禁止在运行 ETL 之前检查报告聚集是否在运行。缺省情况下，此属性设置为 enable。

第 14 章 Interact 设计环境配置属性

本节描述 Interact 设计环境的所有配置属性。

Campaign | partitions | partition[n] | reports

Campaign | partitions | partition[n] | reports 属性定义用于报告的不同类型的文件夹。

offerAnalysisTabCachedFolder

描述

offerAnalysisTabCachedFolder 属性指定一个文件夹的位置，该文件夹包含当您通过单击导航窗格上的"分析"链接来访问"分析"选项卡时，在"分析"选项卡上列示的突发（扩展）商品报告的规范。路径使用 XPath 表示法来指定。

缺省值

```
/content/folder[@name='Affinium Campaign - Object Specific Reports']/
folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

描述

segmentAnalysisTabOnDemandFolder 属性指定一个文件夹的位置，该文件夹包含的是在段的"分析"选项卡上列示的段报告。路径使用 XPath 表示法来指定。

缺省值

```
/content/folder[@name='Affinium Campaign - Object Specific Reports']/
folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

描述

offerAnalysisTabOnDemandFolder 属性指定一个文件夹的位置，该文件夹包含商品的"分析"选项卡上列示的商品报告。路径使用 XPath 表示法来指定。

缺省值

```
/content/folder[@name='Affinium Campaign - Object Specific Reports']/
folder[@name='offer']
```

segmentAnalysisTabCachedFolder

描述

segmentAnalysisTabCachedFolder 属性指定一个文件夹的位置，该文件夹包含当您通过单击导航窗格上的"分析"链接来访问"分析"选项卡时，在"分析"选项卡上列示的突发（扩展）细分市场报告的规范。路径使用 XPath 表示法来指定。

缺省值

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='segment']

analysisSectionFolder

描述

analysisSectionFolder 属性指定存储报告规范的根文件夹的位置。路径使用 XPath 表示法来指定。

缺省值

/content/folder[@name='Affinium Campaign']

campaignAnalysisTabOnDemandFolder

描述

campaignAnalysisTabOnDemandFolder 属性指定一个文件夹的位置，该文件夹包含的是在营销活动的"分析"选项卡上列示的营销活动报告。路径使用 XPath 表示法来指定。

缺省值

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='campaign']

campaignAnalysisTabCachedFolder

描述

campaignAnalysisTabCachedFolder 属性指定一个文件夹的位置，该文件夹包含当您通过单击导航窗格上的"分析"链接来访问"分析"选项卡时，在"分析"选项卡上列示的突发（扩展）营销活动报告的规范。路径使用 XPath 表示法来指定。

缺省值

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='campaign']/folder[@name='cached']

campaignAnalysisTabEmessageOnDemandFolder

描述

campaignAnalysisTabEmessageOnDemandFolder 属性指定一个文件夹的位置，该文件夹包含的是在营销活动的"分析"选项卡上列示的 eMessage 报告。路径使用 XPath 表示法来指定。

缺省值

/content/folder[@name='Affinium Campaign']/folder[@name='eMessage Reports']

campaignAnalysisTabInteractOnDemandFolder

描述

Interact 报告的报告服务器文件夹字符串。

缺省值

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

可用性

只有在安装了 Interact 的情况下，此属性才适用。

interactiveChannelAnalysisTabOnDemandFolder

描述

交互式渠道"分析"选项卡报告的报告服务器文件夹字符串。

缺省值

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking

这些配置属性定义 Interact 联系和响应历史记录模块的设置。

isEnabled

描述

如果设置为 yes，那么会启用 Interact 联系和响应历史记录模块，该模块会将 Interact 联系和响应历史记录从 Interact 运行时中的登台表复制到 Campaign 联系和响应历史记录表。属性 interactInstalled 也必须设置为 yes。

缺省值

no

有效值

yes | no

可用性

只有在安装了 Interact 的情况下，此属性才适用。

runOnceADay

描述

指定是否一天运行"联系和响应历史记录 ETL"一次。如果将此属性设置为 Yes，那么 ETL 会在由 preferredStartTime 和 preferredEndTime 指定的调度时间间隔期间运行。

如果 ETL 的执行时间超过 24 小时，并因此错过了第二天的启动时间，那么会跳过该天，而在第三天的调度时间运行。例如，如果 ETL 配置为在 1AM 至 3AM 之间运行，并且流程在星期一 1AM 启动，在星期二 2AM 完成，那么会跳过原来调度为星期二 1AM 的下一次运行，而下一次 ETL 将在星期三 1AM 启动。

ETL 调度不考虑夏令时更改。例如，如果 ETL 调度为在 1AM 和 3AM 之间运行，当夏令时更改发生时，它将在 12AM 或 2AM 运行。

缺省值

否

可用性

只有在安装了 Interact 的情况下，此属性才适用。

processSleepIntervalInMinutes

描述

Interact 联系和响应历史记录模块在将数据从 Interact 运行时登台表复制到 Campaign 联系和响应历史记录表之间等待的分钟数。

缺省值

60

有效值

大于零的任何整数。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

preferredStartTime

描述

启动每日 ETL 流程的首选时间。与 preferredEndTime 属性配合使用时，此属性会设置运行 ETL 的首选时间间隔期间。ETL 将在指定的时间间隔期间启动，并且最多处理使用 `maxJDBCFetchBatchSize` 指定的记录数。格式是 HH:mm:ss AM 或 PM，使用 12 小时制时钟。

缺省值

12:00:00 AM

可用性

只有在安装了 Interact 的情况下，此属性才适用。

preferredEndTime

描述

完成每日 ETL 流程的首选时间。与 preferredStartTime 属性配合使用时，此属性会设置要运行 ETL 的首选时间间隔期间。ETL 将在指定的时间间隔期间启动，并且最多处理使用 `maxJDBCFetchBatchSize` 指定的记录数。格式是 HH:mm:ss AM 或 PM，使用 12 小时制时钟。

缺省值

2:00:00 AM

可用性

只有在安装了 Interact 的情况下，此属性才适用。

purgeOrphanResponseThresholdInMinutes

描述

Interact 联系和响应历史记录模块在清除没有相应联系的响应之前等待的分钟数。在不记录联系的情况下，这可避免记录响应。

缺省值

180

有效值

大于零的任何整数。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

maxJDBCInsertBatchSize

描述

落实查询之前的 JDBC 批处理的记录的最大数目。这并不是 Interact 联系和响应历史记录模块在一个迭代中处理的记录的最大数目。在每个迭代期间，Interact 联系和响应历史记录模块会处理登台表中所有可用的记录。但是，会将所有这些记录分为 maxJDBCInsertSize 个组块。

缺省值

1000

有效值

大于零的任何整数。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

maxJDBCFetchBatchSize

描述

要从登台数据库中访存的 JDBC 批处理的记录的最大数目。可能需要提高此值以调整联系和响应历史记录模块的性能。

例如，要一天处理 250 万个联系历史记录，应该将 maxJDBCFetchBatchSize 设置为大于 2.5M 的数，以便可处理一天的所有记录。

然后可以将 maxJDBCFetchChunkSize 和 maxJDBCInsertBatchSize 设置为较小的值（在此示例中，分别可以是 50,000 和 10,000）。还可以处理下一天的一些记录，但会将这些记录保留到下一天。

缺省值

1000

有效值

大于 0 的任何整数

maxJDBCFetchChunkSize

描述

在 ETL（抽取、变换、装入）期间所读取数据的 JDBC 组块大小的最大数目。在某些情况下，大于插入大小的组块大小可提高 ETL 流程的速度。

缺省值

1000

有效值

大于 0 的任何整数

deleteProcessedRecords

描述

指定在联系历史记录和响应历史记录得到处理之后是否予以保留。

缺省值

是

completionNotificationScript

描述

指定 ETL 完成后要运行的脚本的绝对路径。如果指定脚本，那么会将 5 个参数传递至完成通知脚本：开始时间、结束时间、处理的联系历史记录总数以及处理的响应历史记录总数和状态。开始时间和结束时间是表示自从 1970 年以来已经过毫秒数的数字值。状态参数指示 ETL 作业是成功还是失败。0 指示 ETL 作业成功。1 指示失败，且 ETL 作业中存在一些错误。

缺省值

无

fetchSize

描述

当从登台表中检索记录时，允许您设置 JDBC fetchSize。

特别地，在 Oracle 数据库上，将该设置调整为 JDBC 应该通过每个网络来回检索的记录数。对于 100K 或以上的大型批处理，请尝试 10000。请小心，在这里不要使用太大的值，因为这会对内存使用产生影响，并且增益将很小（如果不是忽略不计）。

缺省值

无

daysBackInHistoryToLookupContact

描述

将在回应历史记录查询中搜索到的记录限制为在过去指定天数内的记录。对于具有大量回应历史记录的数据库，这样可以将搜索时间段限制为指定的天数内，从而减少处理查询所需的时间。

缺省值 0 指示搜索到了所有记录。

缺省值

0（零）

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

这些配置属性定义 Interact 联系和响应历史记录模块的数据源。

jndiName

描述

使用 systemTablesDataSource 属性来识别在应用程序服务器（Websphere 或 WebLogic）中为 Interact 运行时表定义的 Java 命名和目录接口 (JNDI) 数据源。

Interact 运行时数据库是用 aci_runtime 和 aci_populate_runtime dll 脚本填充的数据库，并且包含下面举例描述的表和其他表：UACI_CHOfferAttrib 和 UACI_DefaultedStat。

缺省值

未定义缺省值。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

databaseType

描述

Interact 运行时数据源的数据库类型。

缺省值

SQLServer

有效值

SQLServer | Oracle | DB2

可用性

只有在安装了 Interact 的情况下，此属性才适用。

schemaName

描述

包含联系和响应历史记录模块登台表的模式的名称。这应该与运行时环境表相同。

不必定义模式。

缺省值

未定义缺省值。

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

这些配置属性定义营销活动中的联系类型（映射至"联系"以供报告或学习之用）。

已联系

描述

分配给商品联系的 Campaign 系统表中 UA_DtlContactHist 表 ContactStatusID 列的值。该值必须是 UA_ContactStatus 表中的有效条目。请参阅《Campaign 管理员指南》以获取有关添加联系类型的详细信息。

缺省值

2

有效值

大于零的整数。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

这些配置属性定义接收或拒绝的响应以用于报告和学习。

接受

描述

分配给已接受商品的 Campaign 系统表中 UA_ResponseHistory 表的 ResponseTypeID 列的值。该值必须是 UA_UsrResponseType 表中的有效条目。应该为 CountsAsResponse 列分配值 1（响应）。

请参阅《Campaign 管理员指南》以获取有关添加响应类型的详细信息。

缺省值

3

有效值

大于零的整数。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

拒绝

描述

分配给已拒绝商品的 Campaign 系统表中 UA_ResponseHistory 表的 ResponseTypeID 列的值。该值必须是 UA_UsrResponseType 表中的有效条目。应该为 CountsAsResponse 列分配值 2（拒绝）。请参阅《Campaign 管理员指南》以获取有关添加响应类型的详细信息。

缺省值

8

有效值

大于零的任何整数。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | report

与 Cognos 集成时，这些配置属性定义报告名称。

interactiveCellPerformanceByOfferReportName

描述

"按商品分类的交互式单元绩效"报告的名称。此名称必须与 Cognos 服务器上此报告的名称相匹配。

缺省值

按商品分组的交互式单元绩效

treatmentRuleInventoryReportName

描述

"处理规则库存"报告的名称。此名称必须与 Cognos 服务器上此报告的名称相匹配。

缺省值

渠道处理规则库存

deploymentHistoryReportName

描述

"部署历史记录报告"报告的名称。此名称必须与 Cognos 服务器上此报告的名称相匹配

缺省值

渠道部署历史记录

Campaign | partitions | partition[n] | Interact | learning

这些配置属性可让您调整内置学习模块。

confidenceLevel

描述

指出从探索切换到采用之前，您希望学习实用程序具有的信心的百分比。值 0 可有效地关闭探索。

仅当将 Interact 运行时的 `Interact > offerserving > optimizationType` 属性设置为 `BuiltInLearning` 时，此属性才适用。

缺省值

95

有效值

0 和 95 之间可以除以 5 的整数或 99。

validateonDeployment

描述

如果设置为 No，那么在部署时 Interact 不会验证学习模块。如果设置为 yes，那么部署时 Interact 会验证学习模块。

缺省值

No

有效值

Yes | No

maxAttributeNames

描述

Interact 学习实用程序监视的学习属性的最大数目。

仅当将 Interact 运行时的 Interact > offerserving > optimizationType 属性设置为 BuiltInLearning 时，此属性才适用。

缺省值

10

有效值

任何整数。

maxAttributeValues

描述

Interact 学习模块针对每个学习属性跟踪的值的最大数目。

仅当将 Interact 运行时的 Interact > offerserving > optimizationType 属性设置为 BuiltInLearning 时，此属性才适用。

缺省值

5

otherAttributeValue

描述

属性值的缺省名称，用于表示 maxAttributeValues 之外的所有属性值。

仅当将 Interact 运行时的 Interact > offerserving > optimizationType 属性设置为 BuiltInLearning 时，此属性才适用。

缺省值

Other

有效值

字符串或数字。

示例

如果将 maxAttributeValues 设置为 3 并且将 otherAttributeValue 设置为其他值，那么学习模块会跟踪前三个值。会将所有其他值分配给其他类别。例

如，如果您要跟踪访问者属性头发颜色，并且前五个访问者的头发颜色为黑色、棕色、金色、红色和灰色，那么学习实用程序会跟踪黑色、棕色和金色这三种头发颜色。会将红色和灰色这两种颜色分组到 otherAttributeValue（其他属性）下面。

percentRandomSelection

描述

学习模块显示随机商品的时间的百分比。例如，如果将 percentRandomSelection 设置为 5，那么表示学习模块显示随机商品的时间是 5% （每 100 个建议中的 5 个），与分数无关。启用 percentRandomSelection 会覆盖 offerTieBreakMethod 配置属性。启用 percentRandomSelection 时，会设置此属性，不管学习是否开启或是使用内置学习还是外部学习。

缺省值

5

有效值

从 0 一直到 100 的任意整数（禁用 percentRandomSelection 功能）。

recencyWeightingFactor

描述

由 recencyWeightingPeriod 定义的数据集百分比的小数表示。例如，缺省值 .15 表示学习实用程序使用的数据中，15% 来自 recencyWeightingPeriod。

仅当将 Interact 运行时的 Interact > offerserving > optimizationType 属性设置为 BuiltInLearning 时，此属性才适用。

缺省值

0.15

有效值

小于 1 的小数值。

recencyWeightingPeriod

描述

由学习模块授予权重的 recencyWeightingFactor 百分比的数据大小（以小时计）。例如，缺省值 120 表示学习模块使用的数据的 recencyWeightingFactor 来自前 120 小时。

仅当将 optimizationType 设置为 builtInLearning 时，此属性才适用。

缺省值

120

minPresentCountThreshold

描述

必须显示商品的最少次数，才能在计算中使用其数据，随后学习模块进入探索模式。

缺省值

0

有效值

大于或等于 0 的整数。

enablePruning

描述

如果设置为 Yes，那么 Interact 学习模块会在算法上确定学习属性（标准或动态）何时不可预测。如果学习属性不可预测，那么学习模块在确定商品的权重时将不会考虑该属性。此情况持续到学习模块聚集学习数据为止。

如果设置为 No，那么学习模块会始终使用所有学习属性。如果不清除不可预测的属性，那么学习模块的准确度可能会低于它应该具有的准确度。

缺省值

是

有效值

Yes | No

Campaign | partitions | partition[n] | Interact | learning | learningAttributes | [learningAttribute]

这些配置属性定义学习属性。

attributeName

描述

每个 attributeName 都是要学习模块监视的访问者属性的名称。这必须与会话数据中的名称 - 值对的名称相匹配。

仅当将 Interact 运行时的 Interact > offerserving > optimizationType 属性设置为 BuiltInLearning 时，此属性才适用。

缺省值

未定义缺省值。

Campaign | partitions | partition[n] | Interact | deployment

这些配置属性定义部署设置。

chunkSize

描述

对于每个 Interact 部署包，分段的最大大小 (KB)。

缺省值

500

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup]

这些配置属性定义服务器组设置。

serverGroupName

描述

Interact 运行时服务器组的名称。这是出现在"交互式渠道摘要"选项卡上的名称。

缺省值

未定义缺省值。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

这些配置属性定义 Interact 运行时服务器。

instanceURL

描述

Interact 运行时服务器的 URL。服务器组可以包含多个 Interact 运行时服务器；但是，每个服务器都必须在新的类别下创建。

缺省值

未定义缺省值。

示例

`http://server:port/interact`

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | flowchart

这些配置属性定义用于交互式流程图测试运行的 Interact 运行时环境。

serverGroup

描述

Campaign 用来执行测试运行的 Interact 服务器组的名称。此名称必须与在 serverGroups 下面创建的类别名称相匹配。

缺省值

未定义缺省值。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

dataSource

描述

使用 dataSource 属性来标识在执行交互式流程图的测试运行时，Campaign 要使用的物理数据源。此属性应该与由 Campaign > partitions > partitionN > dataSources 属性为定义给 Interact 设计时的测试运行数据源定义的数据源相匹配。

缺省值

未定义缺省值。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

eventPatternPrefix

描述

eventPatternPrefix 属性是一个添加到事件模式名称前面的字符串值，添加该属性后事件模式可用于交互式流程图内的"选择"或"决策"流程中的表达式。

请注意，如果您更改此值，那么必须在交互式渠道中部署全局更改，以便此更新后的配置生效。

缺省值

EventPattern

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

这些配置属性定义缺省商品表的缺省单元代码。仅当您要定义全局商品分配时，才需要配置这些属性。

DefaultCellCode

描述

未在缺省商品表中定义单元代码时，Interact 将使用的缺省单元代码。

缺省值

未定义缺省值。

有效值

符合在 Campaign 中定义的单元代码格式的字符串

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

这些配置属性定义 offersBySQL 表的缺省单元代码。仅当您使用 SQL 查询来获取一组所需候选商品时，才需要配置这些属性。

DefaultCellCode

描述

缺省单元代码，Interact 用于 OffersBySQL 表中在单元代码列中具有空值的任何商品（或者在单元代码列完全缺失的情况下所使用的缺省单元代码）。此值必须是一个有效的单元代码。

缺省值

未定义缺省值。

有效值

符合在 Campaign 中定义的单元代码格式的字符串

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

这些配置属性定义分数覆盖表的缺省单元代码。仅当您要定义个别商品分配时，才需要配置这些属性。

DefaultCellCode

描述

未在分数覆盖表中定义单元代码时，Interact 将使用的缺省单元代码。

缺省值

未定义缺省值。

有效值

符合在 Campaign 中定义的单元代码格式的字符串

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | server | internal

此类别中的属性指定所选 Campaign 分区的集成设置和内部标识限制。如果您的 Campaign 安装具有多个分区，请针对您要影响的每个分区设置这些属性。

internalIdLowerLimit

配置类别

`Campaign|partitions|partition[n]|server|internal`

描述

internalIdUpperLimit 和 internalIdLowerLimit 属性将 Campaign 内部标识约束为指定范围内。请注意，值是包含性的：即，Campaign 可以使用下限和上限。

缺省值

0（零）

internalIdUpperLimit

配置类别

Campaign|partitions|partition[n]|server|internal

描述

internalIdUpperLimit 和 internalIdLowerLimit 属性将 Campaign 内部标识约束为指定范围内。值是包含性的：即，Campaign 可以使用下限和上限。如果安装了 Distributed Marketing，那么将值设置为 2147483647。

缺省值

4294967295

eMessageInstalled

配置类别

Campaign|partitions|partition[n]|server|internal

描述

指示已安装了 eMessage。选择 Yes 时, eMessage 功能在 Campaign 界面中可用。

对于您的 eMessage 安装中的缺省分区， IBM 安装程序会将此属性设置为 Yes。对于您安装了 eMessage 的其他分区，您必须手动配置此属性。

缺省值

No

有效值

Yes | No

interactInstalled

配置类别

Campaign|partitions|partition[n]|server|internal

描述

安装 Interact 设计环境后，此配置属性应该设置为 Yes 以在 Campaign 中启用 Interact 设计环境。

如果未安装 Interact，请设置为 No。将此属性设置为 No 不会从用户界面除去 Interact 菜单和选项。要除去这些菜单和选项，那么必须使用 configTool 实用程序手动注销 Interact。

缺省值

否

有效值

Yes | No

可用性

只有在安装了 Interact 的情况下，此属性才适用。

MO_UC_integration

配置类别

Campaign|partitions|partition[n]|server|internal

描述

如果**平台**配置设置中对 Marketing Operations 启用了集成，那么将针对此分区启用该集成。有关更多信息，请参阅《*IBM Marketing Operations* 与 *Campaign* 集成指南》。

缺省值

否

有效值

Yes | No

MO_UC_BottomUpTargetCells

配置类别

Campaign|partitions|partition[n]|server|internal

描述

针对此分区，如果启用了 **MO_UC_integration**，那么将允许对目标单元电子表格使用自下而上单元。当设置为 Yes 时，自上而下和自下而上的目标单位都将可见，但是自下而上目标单位为只读。有关更多信息，请参阅《*IBM Marketing Operations* 与 *Campaign* 集成指南》。

缺省值

否

有效值

Yes | No

Legacy_campaigns

配置类别

Campaign|partitions|partition[n]|server|internal

描述

针对此分区，将在集成 Marketing Operations 与 Campaign 之前启用对所创建营销活动的访问权。仅当 **MO_UC_integration** 设置为 Yes 时才适用。已有的营销活动还包括在 Campaign 7.x 中创建并且链接至 Plan 7.x 项目的营销活动。有关更多信息，请参阅《*IBM Marketing Operations* 与 *Campaign* 集成指南》。

缺省值

否

有效值

Yes | No

IBM Marketing Operations - 商品集成

配置类别

Campaign|partitions|partition[n]|server|internal

描述

如果对此分区启用了 **MO_UC_integration**，那么将在此分区上启用使用 Marketing Operations 执行商品生命周期管理任务的功能。必须在**平台**配置设置中启用商品集成。有关更多信息，请参阅《*IBM Marketing Operations 与 Campaign* 集成指南》。

缺省值

否

有效值

Yes | No

UC_CM_integration

配置类别

Campaign|partitions|partition[n]|server|internal

描述

为 Campaign 分区启用 Digital Analytics 联机细分市场集成。如果您将此值设置为 Yes，那么流程图中的"选择"流程框会提供用于选择 **Digital Analytics 细分市场**作为输入的选项。要为每个分区配置 Digital Analytics 集成，请选择**设置 > 配置 > Campaign | partitions | partition[n] | Coremetrics**。

缺省值

否

有效值

Yes | No

numRowsReadToParseDelimitedFile

配置类别

Campaign|partitions|partition[n]|server|internal

描述

将定界的文件映射为用户表时，将使用此属性。从 IBM SPSS® Modeler Advantage Enterprise Marketing Management Edition 中导入分数输出文件时，"分数"流程框也会使用此属性。要导入或映射定界的文件，Campaign 需要解析该文件以确定列、数据类型（字段类型）和列宽（字段长度）。

缺省值 100 意味着 Campaign 会检查定界的文件中最前面 50 行条目和最后面 50 行条目。然后 Campaign 会根据其在这些条目中找到的最大值来分配字段长度。在大多数情况下，缺省值足以确定字段长度。但是，在非常大的定界的文件中，较后面的字段可能会超过 Campaign 计算出的估算长度，这会导致流程图运行时发生错误。因此，如果您要映射非常大的文件，那么可以增大此

值，以使 Campaign 检查更多行条目。例如，值 200 会使 Campaign 检查该文件的最前面 100 行条目和最后面 100 行条目。

值 0 将检查整个文件。通常，仅当在以下情况下才有必要指定值 0：您要导入或映射其中的字段具有可变数据宽度的文件，但无法通过读取最前面和最后面的一些行来确定宽度。对于极大的文件，读取整个文件会增加表映射和"分数"流程框运行所需的处理时间。

缺省值

100

有效值

0（所有行）或任何正整数

Campaign | monitoring

此类别中的属性指定是否启用"操作监视"功能、"操作监视"服务器的 URL 和高速缓存行为。"操作监视"显示并允许您控制活动的流程图。

cacheCleanupInterval

描述

cacheCleanupInterval 属性指定自动清除流程图状态高速缓存之间的时间间隔（以秒计）。

此属性在低于 7.0 的 Campaign 版本中不可用。

缺省值

600（10 分钟）

cacheRunCompleteTime

描述

cacheRunCompleteTime 属性指定高速缓存已完成的运行并在"监视"页面上显示的时间长度（以分钟计）。

此属性在低于 7.0 的 Campaign 版本中不可用。

缺省值

4320

monitorEnabled

描述

monitorEnabled 属性指定是否打开监视器。

此属性在低于 7.0 的 Campaign 版本中不可用。

缺省值

FALSE

有效值

TRUE | FALSE

serverURL

描述

Campaign > monitoring > serverURL 属性指定"操作监视"服务器的 URL。这是必要的设置；如果"操作监视"服务器 URL 不是缺省值，请修改该值。

如果 Campaign 配置为使用安全套接字层 (SSL) 通信，请将此属性的值设置为使用 HTTPS。例如：serverURL=https://*host:SSL_port*/Campaign/OperationMonitor，其中：

- *host* 是安装 Web 应用程序的机器的名称或 IP 地址
- *SSL_Port* 是 Web 应用程序的 SSL 端口。

请注意 URL 中使用 https。

缺省值

http://localhost:7001/Campaign/OperationMonitor

monitorEnabledForInteract

描述

如果设置为 TRUE，那么会启用用于 Interact 的 Campaign JMX 连接器服务器。Campaign 没有 JMX 安全性。

如果设置为 FALSE，那么您无法连接至 Campaign JMX 连接器服务器。

此 JMX 监视仅用于 Interact 联系和响应历史记录模块。

缺省值

FALSE

有效值

TRUE | FALSE

可用性

只有在安装了 Interact 的情况下，此属性才适用。

协议

描述

如果将 monitorEnabledForInteract 设置为 yes，那么侦听 Campaign JMX 连接器服务器的协议。

此 JMX 监视仅用于 Interact 联系和响应历史记录模块。

缺省值

JMXMP

有效值

JMXMP | RMI

可用性

只有在安装了 Interact 的情况下，此属性才适用。

port

描述

如果将 monitorEnabledForInteract 设置为 yes，那么侦听 Campaign JMX 连接器服务器的端口。

此 JMX 监视仅用于 Interact 联系和响应历史记录模块。

缺省值

2004

有效值

在 1025 和 65535 之间的整数。

可用性

只有在安装了 Interact 的情况下，此属性才适用。

Campaign | partitions | partition[n] | Interact | outboundChannels

这些配置属性使您能够调整触发式消息的出站渠道。

category name

描述

此属性定义此出站渠道的名称。所有出站渠道的名称都必须唯一。

name

描述

出站渠道的名称。

注：必须重新启动应用程序服务器以使更改生效。

Campaign | partitions | partition[n] | Interact | outboundChannels | Parameter Data

这些配置属性使您能够调整触发式消息的出站渠道。

category name

描述

此属性定义此参数的名称。该出站渠道的所有参数的名称都必须唯一。

值

描述

此属性以"名称/值"对格式定义此出站网关需要的参数。

Campaign | partitions | partition[n] | Interact | Simulator

这些配置属性定义要用于运行 API 模拟的服务器组。

serverGroup

描述

指定用于运行 API 模拟的运行时服务器组。

缺省值

defaultServerGroup

第 15 章 客户机端的实时商品个性化

可能在某些情况下，您希望提供实时商品个性化，而不实现对 Interact 服务器的低级别 Java 代码或 SOAP 调用。例如，当访问者最初装入某个 JavaScript 内容是唯一可用扩展编程的 Web 页面时，或者当访问打开一封仅 HTML 内容能够呈现的电子邮件时。IBM Interact 提供了几个连接器，通过这几个连接器，您可以在只能控制位于客户机端的 Web 内容的情况下，或者在希望简化 Interact 的接口的情况下，管理实时商品。

您的 Interact 安装包含了两个连接器，以用于客户机端上启动的商品个性化：

- 『关于 Interact 消息连接器』. 通过使用消息连接器，电子邮件中的 Web 内容（举例而言）或者其他电子媒体可以包含图像和链接标记，以对用于页面-负载商品呈现和点击链接登录页面的 Interact 服务器进行调用。
- 第 290 页的『关于 Interact Web 连接器』. 使用 Web 连接器（也称为 JS 连接器），Web 页面可以使用客户机端 JavaScript，通过页面-负载商品呈现和点击链接登录页面来管理商品仲裁、呈现以及联系/响应历史记录。

关于 Interact 消息连接器

Interact 消息连接器允许电子邮件和其他电子媒体调用 IBM Interact，从而能够在打开时以及在客户单击指向指定站点的消息时呈现个性化商品。这通过使用以下两个关键标记来完成：图像标记 (IMG)，可在打开时装入个性化商品；链接标记 (A)，可捕获有关点击链接的信息并将客户重定向到某一特定登录页面。

示例

以下示例显示了您可能会包含在市场营销热点（例如，在电子邮件中）中的部分 HTML 代码，此 HTML 代码既包含 IMG 标记 URL（可在文档打开时将信息传递到 Interact 服务器并在响应中检索恰当的商品图像），又包含一个 A 标记 URL（确定在点击链接时要传递到 Interact 服务器的信息）：

```
<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>
```

在以下示例中，IMG 标记括在了 A 标记中，从而导致以下行为：

1. 在打开电子邮件时，消息连接器将收到一个请求，其中包含在 IMG 标记中进行编码的信息：此消息的 msgID 和 linkID 以及包含用户标识、收入水平和收入类型的客户参数。
2. 此信息通过某个 API 调用而传递到 Interact 运行时服务器。
3. 运行时服务器将商品返回到消息连接器，消息连接器检索商品图像的 URL，然后提供该 URL（包含任何附加参数）并将图像重定向到该商品 URL。
4. 客户看到图像形式的商品。

此时，客户可能单击该图像从而以某种方式来响应商品。使用 A 标记及其指定 HREF 属性（可指定目标 URL）的这一点击链接会将其他请求发送到与该商品的 URL 链接的登录页面的消息连接器。然后会将客户浏览器重定向到商品中所配置的登录页面。

请注意，点击链接 A 标记并非严格必需；商品可能仅包含一个图像，如，可供客户打印的优惠券。

安装消息连接器

IBM Interact 运行时服务器安装已经自动包含了安装、部署和运行消息连接器所需要的文件。本节概述了准备好让消息连接器可供使用而需要的步骤。

安装和部署消息连接器涉及到以下任务：

- （可选）按照『配置消息连接器』中所述，配置消息连接器的缺省设置。
- 按照第 286 页的『创建消息连接器表』中所述，创建存储消息连接器事务数据而需要的数据库表。
- 按照第 287 页的『部署和运行消息连接器』中所述，安装消息连接器 Web 应用程序。
- 按照第 287 页的『创建消息连接器链接』中所述，在您的市场营销热点（例如，电子邮件或 Web 页面）中创建在打开和点击链接时调用消息连接器商品而需要的 IMG 和 A 标记。

配置消息连接器

在部署消息连接器之前，必须定制您的安装所随附的配置文件以匹配您的特定环境。您可以修改名为 MessageConnectorConfig.xml 的 XML 文件，此文件位于 Interact 运行时服务器上的消息连接器目录中，类似于 <Interact_home>/msgconnector/config/MessageConnectorConfig.xml。

关于此任务

MessageConnectorConfig.xml 文件包含一些必需的配置设置和一些可选的配置设置。必须针对您的特定安装来定制您使用的设置。请遵循此处的步骤来修改配置。

过程

1. 如果在 Web 应用程序服务器上部署并运行消息连接器，在继续之前，请取消部署消息连接器。
2. 在 Interact 运行时服务器上，在任何文本或 XML 编辑器中打开 MessageConnectorConfig.xml 文件。
3. 根据需要来修改配置设置，确保以下必需设置对于您的安装是正确的。
   - <interactUrl>，消息连接器页面标记应连接到的以及运行消息连接器的 Interact 运行时服务器的 URL。
   - <imageErrorLink>，在处理商品图像的请求发生错误的情况下消息连接器应重定向到的 URL。
   -

<landingPageErrorLink>，在处理商品登录页面的请求发生错误的情况下消息连接器应重定向到的 URL。

- <audienceLevels>，配置文件的一部分，其中包含一个或多个受众级别设置集合，以及在消息连接器链接未指定任何内容的情况下哪个设置指定缺省受众级别。必须至少配置了一个受众级别。

『消息连接器配置设置』中更加详细地描述了所有配置设置。

4. 当您完成了配置更改时，保存并关闭 MessageConnectorConfig.xml 文件。
5. 继续设置和部署消息连接器。

消息连接器配置设置：

要配置消息连接器，您可以修改一个名为 MessageConnectorConfig.xml 的 XML 文件，此文件位于 Interact 运行时服务器上的消息连接器目录中，通常是 <Interact_home>/msgconnector/config/MessageConnectorConfig.xml。以下描述了此 XML 文件中的每个配置。请注意，如果在消息连接器已部署且在运行之后修改此文件，请务必在完成修改此文件之后，取消部署消息连接器，然后重新部署，或者重新启动应用程序服务器以重新装入这些设置。

常规设置

下表包含 MessageConnectorConfig.xml 文件的 generalSettings 节中可选设置和必需设置的列表。

表 *24.* 消息连接器常规设置

元素	描述	缺省值
<interactURL>	用于处理来自消息连接器页面标记的 Interact 运行时服务器（如运行消息连接器的运行时服务器）的 URL。该元素是必需的。	http://localhost:7001/interact
<defaultDateTimeFormat>	缺省日期格式。	MM/dd/yyyy
<log4jConfigFileLocation>	Log4j 属性文件的位置。此位置相对于 *$MESSAGE_CONNECTOR_HOME* 环境变量（如果已设置）；如果未设置此环境变量，那么此值是相对于消息连接器 Web 应用程序的根路径。	config/MessageConnectorLog4j.properties

缺省参数值

下表包含 MessageConnectorConfig.xml 文件的 defaultParameterValues 节中可选设置和必需设置的列表。

表 *25.* 消息连接器缺省参数设置

元素	描述	缺省值
<interactiveChannel>	缺省交互式渠道的名称。	
<interactionPoint>	缺省交互点的名称。	
<debugFlag>	确定是否启用调试。允许的值为 true 和 false。	false

表 25. 消息连接器缺省参数设置 (续)

元素	描述	缺省值
<contactEventName>	所发布的联系事件的缺省名称。	
<acceptEventName>	所发布的接受事件的缺省名称。	
<imageUrlAttribute>	包含商品图像的 URL 的缺省商品属性名称（如果消息连接器链接中未指定属性名称）。	
<landingPageUrlAttribute>	点击链接登录页面的缺省 URL（如果消息连接器链接中未指定 URL）。	

行为设置

下表包含 MessageConnectorConfig.xml 文件的 behaviorSettings 节中可选设置和必需设置的列表。

表 26. 消息连接器行为设置

元素	描述	缺省值
<imageErrorLink>	要将连接器重定向到的 URL（如果为商品图像处理请求时发生错误）。此设置是必需设置。	/images/default.jpg
<landingPageErrorLink>	要将连接器重定向到的 URL（如果为点击链接登录页面处理请求时发生错误）。此设置是必需设置。	/jsp/default.jsp
<alwaysUseExistingOffer>	确定是否应返回已高速缓存的商品，即使其已经到期。允许的值为 true 和 false。	false
<offerExpireAction>	在找到了原始商品但原始商品已到期的情况下要采取的操作。允许的值如下： • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

存储设置

下表包含 MessageConnectorConfig.xml 文件的 storageSettings 节中可选设置和必需设置的列表。

表 27. 消息连接器存储设置

元素	描述	缺省值
<persistenceMode>	当高速缓存将新条目持久存储到数据库时。允许的值为 WRITE-BEHIND（数据最初写入到高速缓存，随后更新到数据库）和 WRITE-THROUGH（数据同时写入到高速缓存和数据库）。	WRITE-THROUGH
<maxCacheSize>	内存高速缓存中条目的最大数量。	5000
<maxPersistenceBatchSize>	将条目持久存储到数据库时的最大批处理大小。	200

表 27. 消息连接器存储设置（续）

元素	描述	缺省值
<macCachePersistInterval>	在将某一条目持久存储到数据库之前，其在高速缓存中停留的最大时间（以秒为单位）。	3
<maxElementOnDisk>	磁盘高速缓存中条目的最大数量。	5000
<cacheEntryTimeToExpireInSeconds>	对于磁盘高速缓存中条目，在到期之前可保持的最大时间长度。	60000
<jdbcSettings>	包含特定信息的 XML 文件的节（如果使用 JDBC 连接）。此节与 <dataSourceSettings> 节互斥。	缺省情况下配置为连接到在本地服务器上配置的 SQL Server 数据库，但是如果启用此节，那么必须提供实际的 JDBC 设置和凭证才能登录。
<dataSourceSettings>	包含特定信息的 XML 文件的节（如果使用数据源连接）。此节与 <jdbcSettings> 节互斥。	缺省情况下配置为连接到本地 Web 应用程序服务器上定义的 InteractDS 数据源。

受众级别

下表包含 MessageConnectorConfig.xml 文件的 audienceLevels 节中可选设置和必需设置的列表。

请注意，可以选择使用 audienceLevels 元素来指定在消息连接器链接中未指定任何受众级别的情况下要使用的缺省受众级别，如以下示例中所示：

```
<audienceLevels default="Customer">
```

在此示例中，缺省属性的值与此节中定义的 audienceLevel 的名称相匹配。此配置文件中必须至少定义一个受众级别。

表 28. 消息连接器受众级别设置

元素	元素	描述	缺省值
<audienceLevel>		包含受众级别配置的元素。提供一个名称属性，如 <audienceLevel name="Customer"> 中一样	
	<messageLogTable>	日志表的名称。此值是必需的。	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<field>	此 audienceLevel 的一个或多个受众标识字段的定义。	
	<name>	受众标识字段的名称，如 Interact 运行时中所指定。	
	<httpParameterName>	此受众标识字段的对应参数名称。	
	<dbColumnName>	数据库中此受众标识字段的对应列名称。	
	<type>	受众标识字段的类型，如 Interact 运行时中所指定。值可以是 string 或 numeric。	

创建消息连接器表

要能够部署 IBM Interact 消息连接器，您必须首先在存储 Interact 运行时数据的数据库中创建表。您将为您已定义的每个受众级别创建一个表。对于每个受众级别，Interact 将使用您创建的表来记录有关消息连接器事务的信息。

关于此任务

使用数据库客户机来对相应数据库或模式运行消息连接器 SQL 脚本，从而创建必要的表。在您安装 Interact 运行时服务器时，将自动安装用于受支持数据库的 SQL 脚本。请参阅您在《IBM Interact Installation Guide》中完成的工作表，以了解有关连接到数据库（其中包含 Interact 运行时表）的详细信息。

过程

1. 启动您的数据库客户机并连接到当前存储 Interact 运行时表的数据库。
2. 运行 *<Interact_home>*/msgconnector/scripts/ddl 目录中的相应脚本。下表列出了您可以用于手动创建消息连接器表的样本 SQL 脚本：

表 *29.* 用于创建消息连接器表的脚本

数据源类型	脚本名称
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

请注意，这些脚本作为示例提供。您可以对受众标识值使用不同命名约定或结构，因此您可能需要在运行脚本之前修改脚本。通常，最佳实践是有一个专用于每个受众级别的表。

创建表以便包含以下信息：

表 *30.* 样本 *SQL* 脚本创建的信息

列名称	描述
LogId	此条目的主键。
MessageId	每个消息传递实例的唯一标识。
LinkId	电子媒体（如电子邮件）中每个链接的唯一标识。
OfferImageUrl	指向已返回商品的相关图像的 URL。
OfferLandingPageUrl	指向已返回商品的相关登录页面的 URL。
TreatmentCode	已返回商品的处理代码。
OfferExpirationDate	已返回商品的截止日期和时间。
OfferContactDate	商品返回给客户的日期和时间。
AudienceId	电子媒体的受众标识。

关于此表，请注意以下几点：

- 根据受众级别，受众键的每个组件将具有一个 AudienceId 列。
- MessageId、LinkId 和 AudienceId 的组合构成此表的唯一键。

在脚本完成运行后，您便已经为消息连接器创建了必要的表。

结果

您现在已经准备好部署消息连接器 Web 应用程序。

部署和运行消息连接器

IBM Interact 消息连接器作为独立 Web 应用程序部署在受支持的 Web 应用程序服务器上。

开始之前

要能够部署消息连接器，请确保已完成了以下任务：

- 您必须已安装了 IBM Interact 运行时服务器。可部署的消息连接器应用程序将与运行时服务器一起自动安装，并且可从 Interact 主目录中进行部署。
- 您必须还已经运行了您的安装随附的 SQL 脚本以在 Interact 运行时数据库中创建必要的表以供消息连接器使用（如第 286 页的『创建消息连接器表』中所述）

关于此任务

正如您在 Web 应用程序服务器上部署其他 IBM 应用程序一样，要能够运行这些应用程序，您必须部署消息连接器应用程序以使其可用于提供商品。

过程

1. 通过部署应用程序所需的特权来连接到 Web 应用程序服务器管理界面。
2. 针对您的 Web 应用程序服务器遵循以下指示信息以部署和运行名为 *<Interact_home>*/msgconnector/MessageConnector.war 的文件 将 *<Interact_home>* 替换为 Interact 运行时服务器的实际安装目录。

结果

消息连接器现在可供使用。在您配置了您的 Interact 安装以创建将由消息连接器用于提供商品（如交互式渠道和策略、流程图、商品等等）的底层数据之后，您可以在消息连接器将接受的电子媒体中创建链接。

创建消息连接器链接

要使用消息连接器以在最终用户与您的电子媒体进行交互时（例如，通过打开电子邮件）提供定制商品图像，以及在最终用户点击消息时提供定制登录页面，那么您需要创建要嵌入在您的消息中的链接。此部分提供了这些链接的 HTML 标记的摘要。

关于此任务

无论您使用何种系统来生成外发给最终用户的消息，您都需要生成 HTML 标记以包含相应字段（在 HTML 标记中作为属性提供），这些字段中包含了您要传递给 Interact 运行时服务器的信息。请遵循以下步骤来配置消息连接器消息至少所需要的信息。

请注意，尽管此处的指示信息专门提及了包含消息连接器链接的消息，但是您可以遵循相同步骤和配置来向 Web 页面或任何其他电子媒体中添加链接。

过程

1. 创建一个将显示在消息中的 IMG 链接，至少使用以下参数：
   - msgID，指示此消息的唯一标识。

- linkID，指示消息中链接的唯一标识。
- audienceID，消息的接收方所属于的受众的标识。

  请注意，如果受众标识是一个组合标识，那么所有构成项都必须包含在链接中。

您可能还要包含某些可选参数，这些可选参数包括受众级别、交互式渠道名称、交互点名称、图像位置 URL 以及您自己的定制参数（未专门由消息连接器使用）。

2. （可选）创建一个包含 IMG 链接的 A 链接，这样，当用户单击图像时，浏览器将装入包含用户的商品的页面。 A 链接必须还包含以上列出的三个参数（msgID、linkID 和 audienceID），外加任何可选参数（受众级别、交互式渠道名称和交互点名称）以及定制参数（未专门由消息连接器使用）。 请注意，A 链接将很可能包含消息连接器 IMG 链接，但是根据需要在页面上独立。如果链接确实包含 IMG 链接，那么 IMG 链接应该与括起的 A 链接包含一组相同参数（包括任何可选参数或定制参数）。
3. 在正确定义了链接后，生成并发送电子邮件。

结果

有关可用参数和样本链接的详细信息，请参阅『"IMG"和"A"标记 HTTP 请求参数』

"IMG"和"A"标记 HTTP 请求参数

在消息连接器收到请求时（无论是因为最终用户打开了包含以消息连接器编码的 IMG 标记的电子邮件，还是因为最终用户单击了 A 标记），消息连接器都会解析请求中包含的参数以返回相应的商品数据。此节提供了可以包含在请求 URL 中包含的参数列表（IMG 标记（当电子邮件打开时显示带有标记的图像时自动装入）或 A 标记（在查看电子邮件的用户单击指向指定站点的消息时装入））。

参数

消息连接器收到请求时，将解析请求中包含的参数。这些参数包括以下部分或所有内容：

参数名称	描述	是否必需?	缺省值
msgId	电子邮件实例或 Web 页面的唯一标识。	是	无。这由创建电子邮件唯一实例的系统或者包含标记的 Web 页面来提供。
linkId	此电子邮件或 Web 页面中链接的唯一标识。	是	无。这由创建电子邮件唯一实例的系统或者包含标记的 Web 页面来提供。
audienceLevel	此通信的接收方所属于的受众级别。	否	audienceLevel 在位于 MessageConnectorConfig.xml 文件中的 audienceLevels 元素中被指定为缺省值。
ic	目标交互式渠道 (IC) 的名称	否	interactiveChannel 元素（位于 MessageConnectorConfig.xml 文件的 defaultParameterValues 部分中）的值，缺省情况下为 "interactiveChannel"。
ip	应用交互点 (IP) 的名称	否	interactionPoint 元素（位于 MessageConnectorConfig.xml 文件的 defaultParameterValues 部分中）的值，缺省情况下为 "headBanner"。
offerImageUrl	消息中 IMG URL 的目标商品图像的 URL。	否	无。

参数名称	描述	是否必需?	缺省值
offerImageUrlAttr	具有目标商品图像 URL 的商品属性的名称	否	imageUrlAttribute 元素（位于 MessageConnectorConfig.xml 文件的 defaultParameterValues 部分中）的值。
offerLandingPageUrl	对应于目标商品的登录页面的 URL。	否	无。
offerLandingPageUrlAttr	商品属性的名称，此商品属性具有登录页面（对应于目标商品）的 URL。	否	landingPageUrlAttribute 元素（位于 MessageConnectorConfig.xml 文件的 defaultParameterValues 部分中）的值。
contactEvent	联系事件的名称。	否	contactEventName 元素（位于 MessageConnectorConfig.xml 文件的 defaultParameterValues 部分中）的值，缺省情况下为 "contact"。
responseEvent	接受事件的名称。	否	acceptEventName 元素（位于 MessageConnectorConfig.xml 文件的 defaultParameterValues 部分中）的值，缺省情况下为 "accept"。
debug	调试标志。仅当进行故障诊断以及在由 IBM 技术支持进行指导的情况下，才能将此参数设置为 "true"。	否	debugFlag 元素（位于 MessageConnectorConfig.xml 文件的 defaultParameterValues 部分中）的值，缺省情况下为"false"。
<audience id>	此用户的受众标识。此参数的名称是在配置文件中定义。	是	无。

当消息连接器收到无法识别的参数时（即，并非以上列表中显示的参数），将通过以下两种可能方式中的一种来处理此参数：

- 如果提供了无法识别的参数（例如，attribute="attrValue" 中的"attribute"）并且具有相同名称外加单词"Type"的匹配参数（例如，attributeType="string" 中的 "attributeType"），那么这将导致消息连接器创建一个匹配的 Interact 参数并将其传递给 Interact 运行时。

  Type 参数的值可以是下列任何一个值：

  - string
  - 数字
  - datetime

    对于"datetime"类型的参数，消息连接器还将查找其值为有效日期/时间格式并且名称相同外加单词"Pattern"的参数（例如，"attributePattern"）。例如，您可能提供参数 attributePattern="MM/dd/yyyy"。

    请注意，如果您指定了"datetime"类型的参数，但是未提供匹配的日期模式，那么将使用 Interact 服务器上消息连接器配置文件（位于 <installation_directory>/msgconnector/config/MessageConnectorConfig.xml 中）中指定的值。
- 如果提供了无法识别的参数并且没有匹配的 Type 值，那么消息连接器会将该参数传递到目标重定向 URL。

对于所有无法识别的参数，消息连接器会将其传递到 Interact 运行时服务器，而不会进行处理或保存。

消息连接器代码示例

以下 A 标记包含可能显示在电子邮件中的一组消息连接器链接的示例：

```
<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>
```

在此示例中，当打开电子邮件时，IMG 标记将自动装入。通过从指定页面中检索图像，消息将传递唯一消息标识 (msgID)、唯一链接标识 (linkID) 和唯一用户标识 (userid) 的参数，还将传递要传递到 Interact 运行时的另外两个参数（incomeCode 和 incomeType）。

A 标记提供了 HREF（超文本引用）属性，此属性可将商品图像转换为电子邮件中的可单击链接。如果消息的查看者在看到图像后单击以转向登录页面，那么唯一消息标识 (msgId)、链接标识 (linkId) 和用户标识 (userid) 将传递到服务器，另外一个传递到目标重定向 URL 的参数 (referral) 也将传递到服务器。

关于 Interact Web 连接器

Interact Web 连接器（也称为 JavaScript 连接器或 JS 连接器）在 Interact 运行时服务器上提供了一种服务，此服务允许 JavaScript 代码调用 Interact Java API。这使 Web 页面能够调用 Interact，从而仅使用嵌入的 JavaScript 代码便可进行实时商品个性化，而不必依赖于 Web 开发语言（如 Java、PHP、JSP 等等）。例如，您可能在您的 Web 站点的每个页面上嵌入一小段 JavaScript 代码以用于提供 Interact 所推荐的商品，以便在页面每次装入时，将调用 Interact API 来确保在站点访问者的装入页面上显示最佳商品。

请在以下情况下使用 Interact Web 连接器：您希望在页面上向访问者显示商品，而您无法对页面显示进行服务器端编程控制（例如，您本来可以通过 PHP 或其他基于服务器的脚本编制来进行），但是仍可在页面内容中嵌入将由访问者的 Web 浏览器执行的 JavaScript 代码。

提示：Interact Web 连接器文件自动安装到您的 Interact 运行时服务器中，并且安装在 *<Interact_home>*/jsconnector 目录中。在 *<Interact_home>*/jsconnector 目录中，您将发现 ReadMe.txt，其中包含有关 Web 连接器功能的重要注释和详细信息，还包含样本文件和 Web 连接器源代码以供用作开发您自己的解决方案的基础。如果您没有在此处找到解答您的问题的信息，请参阅 jsconnector 目录以获取更多信息。

在运行时服务器上安装 Web 连接器

Web 连接器的实例将随您的 IBM Interact 运行时服务器一起自动安装，并且缺省情况下将会启用。但是，您必须首先修改某些设置才能配置和使用 Web 连接器。

关于此任务

您必须进行修改才能使用 Web 连接器（安装在运行时服务器上）的某些设置已添加到 Web 应用程序服务器的配置。因此，您在完成这些步骤后需要重新启动 Web 应用程序服务器。

过程

1. 对于安装了 Interact 运行时服务器的 Web 应用程序服务器，请设置以下 Java 属性：

   -DUI_JSCONNECTOR_ENABLE_INPROCESS=true

   -DUI_JSCONNECTOR_HOME=<*jsconnectorHome*>

   将 <*jsconnectorHome*> 替换为运行时服务器上 jsconnector 目录的路径，即 <Interact_Home>/jsconnector。

   您设置 Java 属性的方式取决于您的 Web 应用程序服务器。例如，在 WebLogic 中，您将编辑 startWebLogic.sh 或 startWebLogic.cmd 文件以更新 JAVA_OPTIONS 设置，如以下示例中所示：

   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/jsconnector"

   在 WebSphere Application Server 中，您将在管理控制台的 Java 虚拟机面板中设置此属性。

   请参阅您的 Web 应用程序服务器文档以了解有关设置 Java 属性的特定指示信息。
2. 请重新启动 Web 应用程序服务器（如果其已在运行），或者立即启动 Web 应用程序服务器以确保使用新的 Java 属性。

结果

在 Web 应用程序服务器已完成其启动过程之后，您便已完成了在运行时服务器上安装 Web 连接器。下一步是连接到其"配置"Web 页面：http://<*host*>:<*port*>/interact/jsp/WebConnector.jsp，其中 <*host*> 是 Interact 运行时服务器名称，<*port*> 是 Web 连接器侦听的端口，如 Web 应用程序服务器所指定。

作为单独 Web 应用程序来安装 Web 连接器

Web 连接器的实例将随您的 IBM Interact 运行时服务器一起自动安装，并且缺省情况下将会启用。但是，您也可以将 Web 连接器部署为其自身的 Web 应用程序（例如，在单独系统上的 Web 应用程序服务器中）并将其配置为与远程 Interact 运行时服务器进行通信。

关于此任务

以下指示信息描述了将 Web 连接器部署为能够访问远程 Interact 运行时服务器的单独 Web 应用程序这一过程。

要能够部署 Web 连接器，您必须已安装了 IBM Interact 运行时服务器，并且必须在另一个系统上具有能够对 Interact 运行时服务器进行网络访问（未由任何防火墙阻止）的 Web 应用程序服务器。

过程

1. 将包含 Web 连接器文件的 jsconnector 目录从 Interact 运行时服务器复制到 Web 应用程序服务器（如 WebSphere Application Server）已配置且在运行的系统。 您可以在您的 Interact 安装目录中找到 jsconnector 目录。

2. 在要部署 Web 连接器实例的系统上，使用任何文本或 XML 编辑器来修改 interactURL 属性，从而配置 jsconnector/jsconnector.xml 文件。

   缺省情况下，这设置为 http://localhost:7001/interact，但是您必须进行修改以匹配远程 Interact 运行时服务器的 URL，如 http://runtime.example.com:7011/interact。

   在部署 Web 连接器之后，可使用 Web 界面来定制 jsconnector.xml 文件中的其余设置。有关更多信息，请参阅『配置 Web 连接器』。

3. 对于您将要部署 Web 连接器的 Web 应用程序服务器，请参阅以下 Java 属性：

   -DUI_JSCONNECTOR_HOME=<*jsconnectorHome*>

   将 <*jsconnectorHome*> 替换为已将 jsconnector 目录复制到 Web 应用程序服务器的实际路径。

   您设置 Java 属性的方式取决于您的 Web 应用程序服务器。例如，在 WebLogic 中，您将编辑 startWebLogic.sh 或 startWebLogic.cmd 文件以更新 JAVA_OPTIONS 设置，如以下示例中所示：

   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/InteractFiles/jsconnector"

   在 WebSphere Application Server 中，您将在管理控制台的 Java 虚拟机面板中设置此属性。

   请参阅您的 Web 应用程序服务器文档以了解有关设置 Java 属性的特定指示信息。

4. 请重新启动 Web 应用程序服务器（如果其已在运行），或者在此步骤中启动 Web 应用程序服务器以确保使用新的 Java 属性。

   请首先等待 Web 应用程序服务器完成其启动过程，然后再继续。

5. 通过部署应用程序所需的特权来连接到 Web 应用程序服务器管理界面。
6. 遵循 Web 应用程序服务器的指示信息来部署和运行以下文件： jsConnector/jsConnector.war

结果

Web 连接器现在已部署到 Web 应用程序中。在完全配置了 Interact 服务器并且已启动且正常运行之后，下一步是连接到"Web 连接器配置"Web 页面，地址为 http://<*host*>:<*port*>/interact/jsp/WebConnector.jsp，其中 <*host*> 是运行您刚刚在其中部署了 Web 连接器的 Web 应用程序服务器的系统，<*port*> 是 Web 连接器侦听的端口，由 Web 应用程序服务器指定。

配置 Web 连接器

Interact Web 连接器的配置设置存储在名为 jsconnector.xml 的文件中，而此文件存储在部署了 Web 连接器的系统上（如 Interact 运行时服务器自身，或者运行 Web 应用程序服务器的单独系统）。您可以使用任何文本编辑器或 XML 编辑器来直接编辑 jsconnector.xml 文件；但是，用于配置几乎所有可用配置设置的一种更简单的方法是从 Web 浏览器中使用"Web 连接器配置"页面。

开始之前

要能够使用 Web 界面来配置 Web 连接器，您必须首先安装和部署用于提供 Web 连接器的 Web 应用程序。在 Interact 运行时服务器上，将在您安装和部署 Interact 时自动安装 Web 连接器的实例。在任何其他 Web 应用程序服务器上，您必须按照第 291 页的『作为单独 Web 应用程序来安装 Web 连接器』中的描述来安装和部署 Web 连接器 Web 应用程序。

过程

1. 打开您的受支持的 Web 浏览器并打开类似于以下内容的 URL：

   `http://<host>:<port>/interact/jsp/WebConnector.jsp`

   - 将 *<host>* 替换为运行 Web 连接器的服务器，如运行时服务器的主机名，或者您在其中部署了单独 Web 连接器实例的服务器的名称。
   - 将 *<port>* 替换为 Web 连接器 Web 应用程序正在侦听连接的端口号，这通常与 Web 应用程序服务器的缺省端口相匹配。
2. 在显示的"配置"页面上，完成以下部分：

表 *31. Web* 连接器配置设置摘要.

部分	设置
基本设置	使用"基本设置"页面可配置在您即将转出标记页面的站点上，Web 连接器的总体行为。这些设置包括站点的基本 URL、Interact 需要使用的关于站点访问者的信息以及类似设置（适用于您计划要通过 Web 连接器代码进行标记的所有页面）。 有关详细信息，请参阅第 294 页的『Web 连接器配置基本选项』。
HTML 显示类型	使用"HTML 显示类型"页面可确定将为页面上的每个交互点提供的 HTML 代码。您可以从缺省模板（.flt 文件）的列表中进行选择，缺省模板中包含要用于每个交互点的级联样式表 (CSS) 代码、HTML 代码以及 JavaScript 代码的某些组合。您可以按原样使用提供的模板，根据需要来定制模板，或者创建您自己的模板。 此页面上的配置设置对应于 `jsconnector.xml` 配置文件的 `interactionPoints` 节。 有关详细信息，请参阅第 296 页的『Web 连接器配置 HTML 显示类型』。
增强页面	使用"增强页面"页面可将特定于页面的设置映射到 URL 模式。例如，您可能设置一个可显示常规欢迎页面的页面映射（如包含文本"index.htm"的任何 URL），其中包含为此映射定义的特定页面装入事件和交互点。 此页面上的配置设置对应于 `jsconnector.xml` 配置文件的 `pageMapping` 节。 有关详细信息，请参阅第 298 页的『Web 连接器配置的"增强页面"』。

3. 在"基本设置"页面上，验证站点级别设置是否对您的安装有效，并且可选择指定调试方式（除非您对问题进行故障诊断，否则不建议这样操作）、Digital Analytics for On Premises 页面标记集成、大部分交互点的缺省设置，然后单击"配置"下面的"HTML 显示类型"链接。
4. 在"HTML 显示类型"页面上，遵循以下步骤以来添加或修改用于定义客户 Web 页面上的交互点的显示模板。

缺省情况下，显示模板（.flt 文件）存储在 *<jsconnector_home>*/conf/html 中。

a. 从列表中选择要检查或用作起始点的 .flt 文件，或单击"添加类型"以创建一种要使用的全新空白交互点模板。

   有关模板内容的信息（如果有）将显示在模板列表的旁边。

b. （可选）在**此显示类型的文件名**字段中修改模板的名称。对于新模板，请将 CHANGE_ME.flt 更新为更有效的内容。

   如果您在此处重命名模板，那么在下次保存模板时，Web 连接器将使用该名称来创建一个新文件。在您修改主体文本然后导航到任何其他字段之后，将会保存模板。

c. 根据需要来修改或完成 HTML 片段信息，包括您要包含的任何样式表 (CSS)、JavaScript 和 HTML 代码。请注意，您还可以包含将在运行时由 Interact 参数替换的变量。例如，会在交互点的指定位置中将 ${offer.HighlightTitle} 自动替换为商品标题。

   使用"HTML 片段"字段下方显示的示例来指示如何设置您的 CSS、JavaScript 或 HTML 代码块的格式。

5. 根据需要使用"增强页面"页面来设置可确定如何在页面上处理特定 URL 模式的页面映射。
6. 完成对配置属性的设置后，单击**转出更改**。 单击**转出更改**将会执行以下操作：
   - 显示 IBM Interact Web 连接器页面标记，此标记包含您复制自 Web 连接器页面并插入到您的 Web 页面的 JavaScript 代码。
   - 备份 Interact 服务器上的现有 Web 连接器配置文件（安装了 Web 连接器的服务器上的 jsconnector.xml 文件），并创建一个包含您已定义的设置的新配置文件。

     备份配置文件存储在 *<jsconnector_home>*/conf/archive/jsconnector.xml.*<date>*.*<time>*中，格式为 jsconnector.xml.20111113.214933.750-0500（其中日期字符串为 20111113，包含时区指示符的时间字符串为 214933.750-0500）

结果

您现在已完成了配置 Web 连接器。

要修改配置，您可以返回到这些步骤的开头，并使用新值再次执行这些步骤，也可以在任何文本或 XML 编辑器中打开配置文件 (<Interact_home>/jsconnector/conf/jsconnector.xml)，并根据需要进行修改。

Web 连接器配置基本选项

使用"Web 连接器配置"页面的"基本设置"页面可配置在您即将转出标记页面的站点上，Web 连接器的总体行为。这些设置包括站点的基本 URL、Interact 需要使用的关于站点访问者的信息以及类似设置（适用于您计划要通过 Web 连接器代码进行标记的所有页面）。

站点范围设置

"站点范围设置"配置选项是全局设置，将影响您在配置的 Web 连接器安装的总体行为。您可以指定以下值：

表 *32. Web* 连接器安装的站点范围设置

设置	描述	jsconnector.xml 中的等效设置
Interact API URL	Interact 运行时服务器的基本 URL。**注：**仅当 Web 连接器未在 Interact 运行时服务器中运行时（即，您已将其单独部署），才使用此设置。	*<interactURL>*
Web 连接器 URL	基本 URL，用于生成点击链接 URL。	*<jsConnectorURL>*
目标 Web 站点的交互式渠道	您已在 Interact 服务器上定义的交互式渠道（表示此页面映射）的名称。	*<interactiveChannel>*
访问者的受众级别	入站访问者的 Campaign 受众级别；用于对 Interact 运行时的 API 调用中。	*<audienceLevel>*
概要文件表中的受众标识字段名称	将在对 Interact 的 API 调用中使用的 audienceId 字段的名称。请注意，当前不支持多字段受众标识。	*<audienceIdField>*
受众标识字段的数据类型	要在对 Interact 的 API 调用中使用的受众标识字段的数据类型（"numeric"或"string"）。	*<audienceIdFieldType>*
表示会话标识的 cookie 名称	将包含会话标识的 cookie 的名称。	*<sessionIdCookie>*
表示访问者标识的 cookie 名称	将包含访问者标识的 cookie 的名称。	*<visitorIdCookie>*

可选功能

"可选功能"配置选项是针对您在配置的 Web 连接器安装的一些可选全局设置。您可以指定以下值：

表 *33. Web* 连接器安装的可选站点范围设置

设置	描述	jsconnector.xml 中的等效设置
启用调试方式	指定（通过 yes 或 no 答案）是否使用特殊调试方式。如果您启用此功能，那么从 Web 连接器返回的内容将包含对 "alert"的 JavaScript 调用，以告知客户机刚刚发生的特定页面映射。客户机必须在 *<authorizedDebugClients>* 设置所指定的文件中具有一个条目，这样才能获取警报。	*<enableDebugMode>*
授权调试客户机主机文件	文件的路径，此文件包含符合调试方式资格的主机或 IP（因特网协议）地址的列表。客户机的主机名或 IP 地址必须显示在指定文件中，这样才能收集调试信息。	*<authorizedDebugClients>*
启用 Digital Analytics for On Premises 页面标记集成	指定（通过回答 yes 或 no）Web 连接器是否应在页面内容的结尾处附加指定的 IBM Digital Analytics for On Premises 标记。	`<enableNetInsightTagging>`

表 33. *Web* 连接器安装的可选站点范围设置 *(续)*

设置	描述	jsconnector.xml 中的等效设置
Digital Analytics for On Premises 标记 HTML 模板文件	HTML/Javascript 模板，用于集成对 Digital Analytics for On Premises 标记的调用。通常，您应接受缺省设置，除非指示您提供其他模板。	<netInsightTag>

Web 连接器配置 HTML 显示类型

使用"HTML 显示类型"页面可确定将为页面上的每个交互点提供的 HTML 代码。您可以从缺省模板（.flt 文件）的列表中进行选择，缺省模板中包含要用于每个交互点的级联样式表 (CSS) 代码、HTML 代码以及 JavaScript 代码的某些组合。您可以按原样使用提供的模板，根据需要来定制模板，或者创建您自己的模板。

注：此页面上的配置设置对应于 jsconnector.xml 配置文件的 interactionPoints 节。

此交互点还包含可以在其中自动放入商品属性的占位符（区域）。例如，您可包含 ${offer.TREATMENT_CODE}，在交互期间，会将其替换成为此商品分配的处理代码。

此页面上显示的模板将从 Web 连接器服务器的 <Interact_home>/jsconnector/conf/html 目录内存储的文件中自动装入。您在此处创建的任何新模板也都将存储在此目录中。

要使用"HTML 显示类型"页面来查看或修改任何现有模板，请从列表中选择 .flt 文件。

要在"HTML 显示类型"页面上创建新模板，请单击**添加类型**。

不论您选择哪一种方法来创建或修改某个模板，模板列表旁边都将显示以下信息：

设置	描述	jsconnector.xml 中的等效设置
此显示类型的文件名	为您正在编辑的模板分配的名称。此名称必须对正在运行 Web 连接器的操作系统有效；例如，如果您的操作系统是 Microsoft Windows，那么不能在名称中使用斜杠 (/)。 如果要创建新的模板，那么此字段将预设为 CHANGE_ME.flt。在继续之前，必须将此名称更改为一个有效的值。	*<htmlSnippet>*

设置	描述	jsconnector.xml 中的等效设置
HTML 片段	Web 连接器应插入到 Web 页面上的交互点的特定内容。此片段可以包含 HTML 代码、CSS 格式设置信息或者要在页面上执行的 JavaScript。 所有这三种类型的内容必须由 BEGIN 和 END 代码括起来，如以下示例中所示： • ${BEGIN_HTML} <*HTML* 内容> ${END_HTML} • ${BEGIN_CSS} <特定交互点的样式表信息> ${END_CSS} • ${BEGIN_JAVASCRIPT} <特定于交互点的 *JavaScript* 代码> ${END_JAVASCRIPT} 您还可以输入一些将在装入页面时自动替换的预定义特殊代码，其中包括： • `${logAsAccept}`：这是一个采用两个参数（目标 URL 以及用于识别商品的接受的处理代码）的宏，通过点击链接 URL 便可将其替换。 • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}` • `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` 此处列出的每个商品代码都表示在 IBM Campaign 中的商品模板（由市场营销人员用于创建 Interact 所返回的商品）中定义的商品属性。 请注意，Web 连接器使用一种提供了众多附加选项的名为 FreeMarker 的模板引擎，当您在页面模板上设置代码时，您会发现此模板引擎很有用。有关更多信息，请参阅http://freemarker.org/docs/index.html。	无等效对象，因为 HTML 片段驻留在其自身的文件中（独立于 jsconnector.xml）。
特殊代码示例	包含一些类型的特殊代码的样本，包括某些将块识别为可插入的 HTML、CSS 或 JAVASCRIPT 以及可放置区域的代码，以引用特定商品元数据。	无等效对象。

当您浏览到其他 Web 连接器配置页面时，您在此页面上进行的更改将自动保存。

Web 连接器配置的"增强页面"

使用"增强页面"页面可将特定于页面的设置映射到 URL 模式。例如，您可能设置一个可显示常规欢迎页面的页面映射（如包含文本"index.htm"的任何传入 URL），其中包含为此映射定义的特定页面装入事件和交互点。

注：此页面上的配置设置对应于 jsconnector.xml 配置文件的 pageMapping 节。

要使用"增强页面"页面来创建新页面映射，请单击**添加页面**链接，然后填写有关映射的必需信息。

页面信息

页面映射的"页面信息"配置选项定义充当此映射的触发器的 URL 模式，还有一些关于 Interact 处理此页面映射的方式的其他设置。

设置	描述	jsconnector.xml 中的等效设置
URL 包含	这是 Web 连接器应在传入页面请求中监视的 URL 模式。例如，如果请求 URL 包含"mortgage.htm"，那么您可能将其与您的抵押贷款信息页面进行匹配。	*<urlPattern>*
此页面或页面集的友好名称	一个可供您参考的有意义名称，用于描述此页面映射表示的内容，如"抵押贷款信息页面"。	*<friendlyName>*
还作为 JSON 数据返回商品以供 JavaScript 使用	一个下拉列表，用于指示您是否希望 Web 连接器在页面内容的结尾处包含 JavaScript 对象表示法 (http://www.json.org/) 格式的原始商品数据。	*<enableRawDataReturn>*

在访问此页面或页面集时要触发 (onload) 的事件。

页面映射的这些配置选项集合定义充当此映射的触发器的 URL 模式，还有一些关于 Interact 处理此页面映射的方式的其他设置。

注：此部分中的配置设置对应于 jsconnector.xml 的 <pageLoadEvents> 部分。

设置	描述	jsconnector.xml 中的等效设置
个别事件	可用于此页面或页面集合的事件的列表。此列表中的事件是您已在 Interact 中定义的事件，选择一个或多个您希望在装入页面时发生的事件。 Interact API 调用的顺序如下： 1. `startSession` 2. 各个页面装入事件的 `postEvent`（前提是您已在 Interact 中定义了个别事件） 3. 对于每个交互点： • `getOffers` • `postEvent(ContactEvent)`	*<event>*

此页面或页面集合上的交互点（商品显示位置）

页面映射的这些配置选项集合使您可以选择要在 Interact 的页面上显示的交互点。

注：此部分中的配置设置对应于 `jsconnector.xml` 的 `<pageMapping>` | `<page>` | `<interactionPoints>` 部分。

设置	描述	jsconnector.xml 中的等效设置
交互点名称复选框	配置文件中已经定义的每个交互点都显示在页面的此部分中。如果选中某个交互点名称旁边的复选框，那么将显示可用于此交互点的选项数量。	*<interactionPoint>*
HTML 元素标识（Interact 将设置 innerHTML）	HTML 元素的名称，此 HTML 元素应接收此交互点的内容。例如，如果您在页面上指定 `<div id="welcomebanner">`，那么您将在此字段中输入 `welcomebanner`（标识值）。	*<htmlElementId>*
HTML 显示类型	一个下拉列表，使您可以选择要用于此交互点的 HTML 显示类型（HTML 片段或者 .flt 文件，在先前的 Web 连接器配置页面上定义）。	*<htmlSnippet>*
要呈现的最大商品数量（如果这是旋转木马式或翻书式）	Web 连接器应从 Interact 服务器中为此交互点检索的最大商品数量。此字段为可选，且仅适用于定期更新呈现的商品而不重新装入页面的交互点，比如在旋转木马式场景中，此时将检索多个商品，以便能够一次将其全部提供。	*<maxNumberOfOffers>*
呈现商品时要触发的事件	要为此交互点发布的联系事件的名称。	*<contactEvent>*
接受商品时要触发的事件	在单击商品链接时要为此交互点发布的接受事件的名称。	*<acceptEvent>*
拒绝商品时要触发的事件	要为此交互点发布的拒绝事件的名称。 **注：**目前，此功能尚未使用	*<rejectEvent>*

Web 连接器配置选项

通常，您可以使用图形 Web 连接器接口来配置您的 Web 连接器设置。您指定的所有设置也存储在一个文件名为 jsconnector.xml 的文件中，该文件位于 jsconnector/conf 目录。此处描述了 jsconnector.xml 配置文件中保存的每个参数。

参数及其描述

以下参数存储在 jsconnector.xml 文件中，用于 Web 连接器交互。有两种方法可以修改这些设置：

- 使用部署并启动 Web 连接器应用程序后自动提供的"Web 连接器配置"web 页面。要使用"配置"Web 页面，请使用 Web 浏览器打开类似于以下内容的 URL：http://*<host>*:*<port>*/interact/jsp/WebConnector.jsp。

  您在"管理"Web 页面上所作的更改将存储在 Web 连接器部署所在的服务器上的 jsconnector.xml 文件中。
- 使用任何文本编辑器或 XML 编辑器直接编辑 jsconnector.xml 文件。在使用此方法之前，请确保您会编辑 XML 标记和值。

  注：无论何时手动编辑 jsconnector.xml 文件，都可以通过打开"Web 连接器管理"页面（在 http://*<host>*:*<port>*/interact/jsp/jsconnector.jsp 中查找）并单击**重新装入配置**来重新装入这些设置。

下表描述了 jsconnector.xml 文件中出现配置选项时，您可以设置的选项。

表 *34. Web* 连接器配置选项

参数组	参数	描述
defaultPageBehavior		
	friendlyName	用于显示在 Web 连接器的 Web 配置页面上的 URL 模式人工可读标识符。
	interactURL	Interact 运行时服务器的基本 URL。注意：仅当 Web 连接器（jsconnector）服务作为已部署的 Web 应用程序运行时，才需要设置此参数。如果 WebConnector 作为 Interact 运行时服务器的一部分自动运行，那么不必设置此参数。
	jsConnectorURL	用于生成点击链接 URL 的基本 URL，如 http://host:port/jsconnector/clickThru
	interactiveChannel	表示此页面映射的交互式渠道的名称。
	sessionIdCookie	包含会话标识的 cookie 的名称，用于对 Interact 的 API 调用。
	visitorIdCookie	包含受众标识的 cookie 名称。
	audienceLevel	入站访问者的营销活动受众级别，用于对 Interact 运行时的 API 调用。
	audienceIdField	audienceId 字段的名称，用于对 Interact 运行时的 API 调用。 **注：**注意：当前不支持多字段受众标识。
	audienceIdFieldType	受众标识字段 [numeric | string] 的数据类型，用于对 Interact 运行时的 API调用。

表 *34. Web* 连接器配置选项 *(续)*

参数组	参数	描述
	audienceLevelCookie	用于包含受众级别的 cookie 的名称。此为可选参数。如果未设置此参数，系统用户将使用为 audienceLevel 定义的参数。
	relyOnExistingSession	用于对 Interact 运行时的 API 调用。通常，此参数设置为"true"。
	enableInteractAPIDebug	用于对 Interact 运行时的 API 调用，以启用"调试输出到日志文件"。
	pageLoadEvents	此特殊页面装入的将一次性发布的事件。指定此标记中的一个或多个事件，格式类似于 <event>event1</event>。
	interactionPointValues	此类别中的所有项将充当 IP 特定类别中缺失值的缺省值。
	interactionPointValuescontactEvent	要针对此特定交互点发布的联系事件的缺省名称。
	interactionPointValuesacceptEvent	要针对此特定交互点发布的接受事件的缺省名称。
	interactionPointValuesrejectEvent	要针对此特定交互点发布的拒绝事件的缺省名称。（注：此时，不使用该功能。）
	interactionPointValueshtmlSnippet	针对此交互点要提供的 HTML 模板的缺省名称。
	interactionPointValuesmaxNumberOfOffers	针对此交互点，要从 Interact 检索的缺省最大商品数。
	interactionPointValueshtmlElementId	要为此交互点接收内容的 HTML 模板的缺省名称。
	interactionPoints	该类别包含各个交互点的配置。对于任何缺失属性，系统将依赖于 interactionPointValues 类别下配置的内容。
	interactionPointname	交互点 (IP) 的名称
	interactionPointcontactEvent	要为此特定 IP 发布的联系事件的缺省名称。
	interactionPointacceptEvent	要为此特定 IP 发布的接受事件的缺省名称。
	interactionPointrejectEvent	要为此特定 IP 发布的拒绝事件的缺省名称。（请注意，此功能尚未使用。）
	interactionPointhtmlSnippet	要为此 IP 提供的 HTML 模板的名称。
	interactionPointmaxNumberOfOffers	要为此 IP 从 Interact 中检索的最大商品数。
	interactionPointhtmlElementId	要为此交互点接收内容的 HTML 元素的名称。
	enableDebugMode	用于打开特殊调试方式的布尔标志（可接受的值：true 或 false）。如果您将其设置为 true，那么从 Web 连接器返回的内容将包含对"alert"的 JavaScript 调用，以告知客户机刚刚发生的特定页面映射。客户机必须有 authorizedDebugClients 文件中的条目才能生成警报。

表 *34. Web* 连接器配置选项 *(续)*

参数组	参数	描述
	authorizedDebugClients	特殊调试方式所使用的文件，其中包含符合调试方式的主机名或因特网协议 (IP) 地址的列表。
	enableRawDataReturn	用于确定 Web 连接器是否已连接到内容末端的 JSON 格式的原料报价数据的布尔标志（可接受的值：true 或 false）。
	enableNetInsightTagging	用于确定 Web 连接器是否已连接到内容末端的 Digital Analytics for On Premises 标记的布尔标志（可接受的值：true 或 false）。
	apiSequence	表示 APISequence 接口的实现，指示调用 pageTag 时，Web 连接器制定的 API 调用的顺序。缺省情况下，该实现采用的顺序为 会话、pageLoadEvents、getOffers 和 logContact，其中最后两个特定于每个交互点。
	clickThruApiSequence	表示 APISequence 接口的实现，指示调用 clickThru 时，Web 连接器制定的 API 调用的顺序。缺省情况下，该实现采用的顺序为 StartSession 和 logAccept。
	netInsightTag	表示用于将调用集成到 Digital Analytics for On Premises 标记的 HTML 和 JavaScript 模板。通常，您应该不需要更改此选项。

使用 Web 连接器的"管理"页面

Web 连接器包含一个"管理"页面，此页面提供了一些工具来帮助管理和测试配置，因为它可能与特定 URL 模式配合使用。您还可以使用"管理"页面来重新装入您已修改的配置。

关于"管理"页面

通过使用任何受支持的 Web 浏览器，您可以打开 http://*host*:*port*/interact/jsp/jsconnector.jsp，其中 *host:port* 是运行 Web 连接器的主机名以及此 Web 连接器正在侦听连接的端口，如 runtime.example.com:7001

您可以通过以下任何一种方式来使用"管理"页面：

表 *35. Web* 连接器的"管理"页面选项

选项	用途
重新装入配置	单击**重新装入配置**链接可重新装入任何已保存在磁盘或内存中的任何配置更改。如果您直接对 Web 连接器 jsconnector.xml 配置文件进行更改，而非使用"配置"Web 页面，那么此操作是必需的。
查看配置	根据您在**查看配置**字段中输入的 URL 模式来查看 Web 连接器配置。如果您输入页面的 URL 并单击**查看配置**，Web 连接器将根据该模式映射来返回系统将要使用的配置。如果未找到匹配项，那么会返回缺省配置。这可用于测试是否对特定页面使用了正确配置。

表 35. *Web* 连接器的"管理"页面选项 *(续)*

选项	用途
执行页面标记	通过填写此页面上的字段并单击**执行页面标记**，将会导致 Web 连接器根据 URL 模式来返回 pageTag 结果。这将模拟对页面标记的调用。 通过该工具调用 pageTag 与使用真实 Web 站点之间的差异在于，使用此"管理"页面将导致显示任何错误或异常。对于真实 Web 站点，不会返回异常，并且异常仅在 Web 连接器日志文件中可见。

Web 连接器页面样本

Interact Web 连接器（目录 <Interact_Home/jsconnector/webapp/html 中）包含了一个名为 WebConnectorTestPageSA.html 的文件来作为示例，用于演示将在页面中标记 Web 连接器的多少个功能。为方便起见，此处也显示了该样本页面。

Web 连接器 HTML 页面样本

```
<?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: IBM Interact Web Connector Page Tag -->
<!--
# ****************************************************************************
#  Licensed Materials - Property of IBM
#  IBM Interact
#  (c) Copyright IBM Corporation 2001, 2012.
#  US Government Users Restricted Rights - Use, duplication or disclosure
#  restricted by GSA ADP Schedule Contract with IBM Corp.
# ****************************************************************************
-->
<script language="javascript" type="text/javascript">
//<![CDATA[
   var unicaWebConnectorBaseURL=
        "[CHANGE ME - http://host:port/<jsconnector>/pageTag]";
   var unicaURLData = "ok=Y";
   try {
    unicaURLData += "&url=" + escape(location.href)
   } catch (err) {}
   try {
    unicaURLData += "&title=" + escape(document.title)
   } catch (err) {}
   try {
    unicaURLData += "&referrer=" + escape(document.referrer)
   } catch (err) {}
   try {
    unicaURLData += "&cookie=" + escape(document.cookie)
   } catch (err) {}
   try {
    unicaURLData += "&browser=" + escape(navigator.userAgent)
   } catch (err) {}
   try {
    unicaURLData += "&screensize=" +
        escape(screen.width + "x" + screen.height)
   } catch (err) {}
   try {
    if (affiliateSitesForUnicaTag) {
        var unica_asv = "";
        document.write("<style id=\"unica_asht1\" type=\"text/css\"> "
          + "p#unica_ashtp a {border:1px #000000 solid; height:100px "
          + "!important;width:100px "
          + "!important; display:block !important; overflow:hidden "
          + "!important;} p#unica_ashtp a:visited {height:999px !important;"
          + "width:999px !important;} <\/style>");
        var unica_ase = document.getElementById("unica_asht1");
        for (var unica_as in affiliateSitesForUnicaTag) {
         var unica_asArr = affiliateSitesForUnicaTag[unica_as];
         var unica_ashbv = false;
         for (var unica_asIndex = 0; unica_asIndex <
          unica_asArr.length && unica_ashbv == false;
     unica_asIndex++)
   {
          var unica_asURL = unica_asArr[unica_asIndex];
          document.write("<p id=\"unica_ashtp\" style=\"position:absolute; "
          + "top:0;left:-10000px;height:20px;width:20px;overflow:hidden; \
      margin:0;padding:0;visibility:visible;\"> \
      <a href=\"" + unica_asURL + "\">" + unica_as + " <\/a><\/p>");
          var unica_ae = document.getElementById("unica_ashtp").childNodes[0];
          if (unica_ae.currentStyle) {
           if (parseFloat(unica_ae.currentStyle["width"]) > 900)
              unica_ashbv = true
```

```
          } else if (window.getComputedStyle) {
           if (parseFloat(document.defaultView.getComputedStyle
              (unica_ae, null).getPropertyValue("width")) > 900)
            unica_ashbv = true
           }
           unica_ae.parentNode.parentNode.removeChild(unica_ae.parentNode)
          }
          if (unica_ashbv == true) {
           unica_asv += (unica_asv == "" ? "" : ";") + unica_as
          }
         }
         unica_ase.parentNode.removeChild(unica_ase);
         unicaURLData += "&affiliates=" + escape(unica_asv)
    }
   } catch (err) {}
   document.write("<script language='javascript' "
     + " type='text/javascript' src='" + unicaWebConnectorBaseURL + "?"
    + unicaURLData + "'><\/script>");
  //]]&gt;
</script>
<style type="text/css">
/*<![CDATA[*/
   .unicainteractoffer {display:none !important;}
   /*]]&gt;*/
</style>
   <title>Sample Interact Web Connector Page</title>
   </head>
   <body>
   <!-- END: IBM Interact Web Connector Page Tag -->
   <!--
################################################################################
end of pageTag paste
################################################################################
-->
   </body>
</html>
```

第 16 章 Interact 与 Digital Recommendations 集成

IBM Interact 可与 IBM Digital Recommendations 集成以提供 Interact 驱动的产品推荐。这两个产品都可以提供针对商品的产品推荐，但是使用的方法不同。Digital Recommendations 会通过访问者的 Web 行为（协作过滤）来构建访问者与推荐商品之间的相关性。Interact 将基于客户过去的行为、特性、历史记录（而较少基于视图级别的商品）来了解哪些商品与客户的行为概要信息（基于人口统计学和其他有关该客户的信息）最匹配。商品验收率可帮助构建经过自我学习后的预测性模型。利用这两个产品的优点，Interact 可使用个人概要信息来定义会将类别标识传递到 Digital Recommendations 中的商品并检索基于受欢迎程度（群体的智慧）的推荐产品以作为选定商品的一部分显示给访问者。这样可以为会导致更多点击链接的客户提供更好的产品推荐并产生比其中任一产品独自操作更好的效果。

以下各节描述此集成如何工作以及如何使用所提供的样本应用程序来创建您自己的定制商品集成。

Interact 与 Digital Recommendations 集成的概述

本节描述 IBM Interact 可以如何与 IBM Digital Recommendations 集成以提供 Interact 驱动的产品推荐，其中包括过程描述、进行集成所使用的机制。

IBM Interact 通过代表性状态传输 (REST) 应用程序编程接口 (API) 与 IBM Digital Recommendations 集成，Digital Recommendations 安装中已提供该 API。通过使用适当的类别标识进行 REST API 调用，Interact 可检索推荐的产品并将其包括在访问者正在查看的定制页面上显示的商品信息中。

当访问者查看 Web 页面（例如，Interact 安装随附的样本 JSP 页面）的 URL 时，该页面会调用 Interact 以访存商品。假设在 Interact 内使用正确的参数配置了商品，在最简单的情况下会发生以下步骤：

1. 页面逻辑识别访问者的客户标识。
2. 对 Interact 进行了 API 调用，并传入生成针对该客户的商品时必需的信息。
3. 返回的商品至少为 Web 页面提供三个属性：商品图像的 URL、客户点击链接时登录页面的 URL、用于确定要推荐的产品的类别标识。
4. 然后，类别标识用来调用 Digital Recommendations 以检索推荐的产品。此套产品采用 JSON（JavaScript 对象表示法）格式，按该类别中的最畅销产品顺序排列。
5. 然后，商品和产品显示在访问者的浏览器中。

对于将商品推荐和产品推荐结合在一起，此集成很有用。例如，您可能在一个 Web 页面上具有两个交互点：一个代表商品，一个代表匹配该商品的产品推荐。要实现这一点，Web 页面将对 Interact 进行调用以进行实时细分来确定最佳商品（比如，所有小家电 10% 的折扣）。当该页面接收到来自 Interact 的商品时，该商品会包含类别标识（在此示例中，为小家电的类别标识）。然后，该页面会通过 API 调用来将小家电的类别标识传递到 Digital Recommendations 中，并作为响应接收该类别中基于受欢迎程度的最佳产品推荐。

更简单的一个示例可以是，Web 页面对 Interact 进行调用以从中找出与客户概要信息匹配的类别（比如，高端餐具）。然后，该页面会将接收到的类别标识传递到 Digital Recommendations 中，然后获得餐具产品推荐。

集成先决条件

您必须先确保满足本节中描述的先决条件，然后才能使用 Digital Recommendations - Interact 集成。

确保满足以下先决条件：

- 熟悉 Interact API 的用法，如《管理员指南》中其他位置和联机帮助所记录。
- 熟悉 Digital Recommendations REST API，如 Digital Recommendations 开发人员文档中所描述。
- 对 JavaScript、CSS 和 JSON（JavaScript 对象表示法）有基本的了解。

  JSON 很重要，因为 Digital Recommendations REST API 会以 JSON 格式的数据形式返回您请求的产品信息。
- 熟悉 Web 页面的服务器端编码，因为 Interact 随附的演示应用程序会使用 JSP（尽管 JSP 不是必需的）。
- 具有有效 Digital Recommendations 帐户以及您计划让 Interact 检索产品推荐（您所指定类别中最畅销或者最受欢迎的产品）的类别标识列表。
- 具有 Digital Recommendations REST API 链接（您的 Digital Recommendations 环境的 URL）。

  请参阅您的 Interact 安装随附的样本应用程序作为示例，或者参阅第 309 页的『使用集成样本项目』中的样本代码以获取更多信息。

配置商品以进行 Digital Recommendations 集成

您必须先使用必需信息配置 IBM Interact 商品以传递到 Digital Recommendations，然后您的 Web 页面才能调用 Digital Analytics Digital Recommendations 以检索推荐产品。

关于此任务

要设置商品以链接至 Digital Recommendations，请首先确保满足以下条件：

- 确保 Interact 运行时服务器已安装并且正在正常运行。
- 确保运行时服务器可与 Digital Recommendations 服务器建立连接，其中包括确保您的防火墙不会阻止标准 Web 连接（端口 80）的传出建立。

要设置商品以与 Digital Recommendations 集成，请执行下列步骤。

过程

1. 为 Interact 创建或编辑商品。

   有关创建和修改商品的信息，请参阅《IBM Interact 用户指南》和 IBM Campaign 文档。
2. 除了商品中的其他设置外，还要确保该商品包含以下商品属性：
   - 链接至商品图像的 URL（统一资源定位符）。

- 链接至商品登录页面的 URL。
- 与此商品相关联的 Digital Recommendations 类别标识。

  您可以手动从 Digital Recommendations 配置中检索类别标识。Interact 无法直接检索类别标识值。

在 Interact 安装随附的演示 Web 应用程序中，这些商品属性名为 ImageURL、ClickThruURL 和 CategoryID。只要您的 Web 应用程序与该商品期望的值匹配，这些名称可以是对您有效的任意值。

例如，您可能会定义名为"10PercentOff"的包含这些属性的商品，其中类别标识（如从 Digital Recommendations 配置所检索）是 PROD1161127，商品点击链接的 URL 是 http://www.example.com/success，要显示的商品图像的 URL 是 http://localhost:7001/sampleIO/img/10PercentOffer.jpg（在这种情况下，Interact 运行时服务器本地的 URL）。

3. 定义交互式渠道的处理规则以包括此商品，并照常部署交互式渠道。

结果

现在，已使用 Digital Recommendations 集成所需的信息定义商品。剩余工作是使 Digital Recommendations 能够将产品推荐提供给 Interact，可通过配置 Web 页面以进行适当的 API 调用来实现。

在配置 Web 应用程序以向访问者提供集成页面时，请确保 WEB-INF/lib 目录中具有下列文件：

- *Interact_Home*/lib/interact_client.jar，是处理 Web 页面对 Interact API 的调用所必需的文件。
- *Interact_Home*/lib/JSON4J_Apache.jar，是处理从对 Digital Recommendations REST API（会返回 JSON 格式的数据）的调用返回的数据所必需的文件。

请参阅『使用集成样本项目』以获取有关如何向客户提供商品的更多信息。

使用集成样本项目

每个 Interact 运行时安装都包括一个用于描述 Digital Recommendations - Interact 集成过程的样本项目。该样本项目提供了有关创建 Web 页面的完整的端到端演示，该页面会调用包含类别标识的商品，然后该商品会传递到 Digital Recommendations 以检索页面交互点中呈示的建议产品列表。

概述

如果您想要测试集成过程，那么可以如所提供的一样使用所包含的样本项目，或者可以将其用作起始点来开发您自己的定制页面。样本项目位于以下文件中：

Interact_home/samples/IntelligentOfferIntegration/MySampleStore.jsp

此文件除了包含集成过程的完整的工作示例外，还包含用于描述在 Interact 中要设置的内容、在 .jsp 文件中要定制的内容以及如何正确部署页面以与您的安装一起运行的大量注释。

MySampleStore.jsp

为方便起见，此处显示了 MySampleStore.jsp 文件。由于此样本可以使用 Interact 的后续发行版来更新，因此可使用您的安装随附的文件作为起始点来进行所需的任何示例。

```
<!--
# ****************************************************************************
#  Licensed Materials - Property of IBM
#  IBM 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 Digital Recommendations.
 *
  * When the URL for this jsp is accessed via a browser. the logic will call Interact
  * to fetch an Offer.  Based on the categoryID associated to the offer, the logic
  * will call Digital Recommendations to fetch recommended products.  The offer and products
  * will be displayed.
  * To toggle the customerId in order to demonstrate different offers, one can simply
  * append cid=<id> to the URL of this JSP.
 *
  * Prerequisites to understand this demo:
  * 1) familiarity of Interact and its java API
  * 2) familiarity of IntelligentOffer and its RestAPI
  * 3) some basic web background ( html, css, javascript) to mark up a web page
  * 4) Technology used to generate a web page (for this demo, we use JSP executed on the server side)
 *
 *
  * Steps to get this demo to work:
  * 1) set up an Interact runtime environment that can serve up offers with the following
  * offer attributes:
  *  ImageURL : url that links to the image of the offer
  *  ClickThruURL : url that links to the landing page of the offer
  *  CategoryID : Digital Recommendations category id associated to the offer
  * NOTE: alternate names for the attributes may be used as long as the references to those
  * attributes in this jsp are modified to match.
  * 2) Obtain a valid REST API URL to the Intelligent Offer environment
  * 3) Embed this JSP within a Java web application
  * 4) Make sure interact_client.jar is in the WEB-INF/lib directory (communication with Interact)
  * 5) Make sure JSON4J_Apache.jar (from interact install) is in the
  *    WEB-INF/lib directory (communication with IO)
  * 6) set the environment specific properties in the next two sections
 *****************************************************************************/

 /*****************************************************************************
  *   ****************CHANGE THESE SETTINGS TO REFLECT YOUR ENV*******************
  * Set your Interact environment specific properties here...
 *****************************************************************************/

  final String sessionId="123";
  final String interactiveChannel = "SampleIO";
  final String audienceLevel = "Customer";
  final String audienceColumnName="CustomerID";
  final String ip="ip1";
  int customerId=1;
  final String interactURL="http://localhost:7011/interact/servlet/InteractJSService";
  final boolean debug=true;
  final boolean relyOnExistingSession=true;
```

```
/*****************************************************************************
     ****************CHANGE THESE SETTINGS TO REFLECT YOUR ENV*******************
 * Set your Digital Recommendations environment specific properties here...
*****************************************************************************/

 final String ioURL="http://recs.coremetrics.com/iorequest/restapi";
 final String zoneID="ProdRZ1";
 final String cID="90007517";



/*****************************************************************************
  ****************************************************************************/

 StringBuilder interactErrorMsg = new StringBuilder();
 StringBuilder intelligentOfferErrorMsg = new StringBuilder();

 // get the customerID if passed in as a parameter
 String cid = request.getParameter("cid");
 if(cid != null)
{
   customerId = Integer.parseInt(cid);
}

 // call Interact to get offer
 Offer offer=getInteractOffer(interactURL,sessionId,interactiveChannel,audienceLevel,
       audienceColumnName,ip,customerId,debug,relyOnExistingSession,interactErrorMsg);

 // get specific attributes from the offer (img url, clickthru url, & category id)
 String offerImgURL=null;
 String offerClickThru=null;
 String categoryId="";

 if(offer != null)
{
 for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
    {
                if(offerAttribute.getName().equalsIgnoreCase("ImageURL"))
        {
                  offerImgURL=offerAttribute.getValueAsString();
        }
                else if(offerAttribute.getName().equalsIgnoreCase("ClickThruURL"))
        {
                  offerClickThru=offerAttribute.getValueAsString();
        }
                else if(offerAttribute.getName().equalsIgnoreCase("CategoryID"))
        {
                  categoryId=offerAttribute.getValueAsString();
        }
    }
}

  // call Digital Recommendations to get products
  JSONObject products=getProductsFromIntelligentOffer(ioURL, cID, zoneID, categoryId,
       intelligentOfferErrorMsg);

%>

<html>
 <head>
    <title>My Favorite Store</title>

  <script language="javascript" type="text/javascript">
      var unicacarousel=(function(){var g=false;var h;var j=0;var k=0;var l=0;var m=40;
          var n=new Array(0,2,6,20,40,60,80,88,94,97,99,100);var o=function(a){var b=a.parentNode;
          h=b.getElementsByTagName("UL")[0];var c=h.getElementsByTagName("LI");j=c[0].offsetWidth;
          k=c.length;l=Math.round((b.offsetWidth/j));unicacarousel.recenter()};var p=function(a)
          {var b=parseFloat(h.style.left);if(isNaN(b))b=0;for(var i=0;i<n.length;i++)
          {setTimeout("unicacarousel.updateposition("+(b+(a*(n[i]/100)))+");",((i*m)+50))}
          setTimeout("unicacarousel.recenter();",((i*m)+50))};return{gotonext:function(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
 *   Digital Recommendations
 *****************************************************************************/

 /*****************************************************************************
  * Call Digital Recommendations to retrieve recommended products
 *****************************************************************************/
  private JSONObject getProductsFromIntelligentOffer(String ioURL, String cID,
     String zoneID, String categoryID, StringBuilder intelligentOfferErrorMsg)
 {

  try {

      ioURL += "?cm_cid="+cID+"&cm_zoneid="+zoneID+"&cm_targetid="+categoryID;
      System.out.println("CoreMetrics URL:"+ioURL);
      URL url = new java.net.URL(ioURL);

        URLConnection conn = url.openConnection();

        InputStreamReader inReader = new InputStreamReader(conn.getInputStream());
        BufferedReader in = new BufferedReader(inReader);

        StringBuilder response = new StringBuilder();

        while(in.ready())
    {
          response.append(in.readLine());
    }

        in.close();

        intelligentOfferErrorMsg.append(response.toString());

        System.out.println("CoreMetrics:"+response.toString());

        if(response.length()==0)
          return null;

        return new JSONObject(response.toString());
  }
    catch(Exception e)
  {
      intelligentOfferErrorMsg.append(e.getMessage());
      e.printStackTrace();
  }

       return null;

 }

 /*****************************************************************************
  * Call Interact to retrieve offer
 *****************************************************************************/
  private Offer getInteractOffer(String interactURL,String sessionId,String interactiveChannel,
     String audienceLevel,
```

```
              String audienceColumnName,String ip, int customerId,boolean debug,
                     boolean relyOnExistingSession, StringBuilder interactErrorMsg)
 {
  try {
      InteractAPI api = InteractAPI.getInstance(interactURL);
      NameValuePairImpl custId = new NameValuePairImpl();
            custId.setName(audienceColumnName);
              custId.setValueAsNumeric(Double.valueOf(customerId));
     custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
              NameValuePairImpl[] audienceId = { custId };

              // call startSession
            Response response = api.startSession(sessionId, relyOnExistingSession,
          debug, interactiveChannel, audienceId, audienceLevel, null);

            if(response.getStatusCode() == Response.STATUS_ERROR)
      {
              printDetailMessageOfWarningOrError("startSession",response, interactErrorMsg);
    }

            // call getOffers
            response = api.getOffers(sessionId, ip, 1);
            if(response == null  || response.getStatusCode() == Response.STATUS_ERROR)
      {
              printDetailMessageOfWarningOrError("getOffers",response, interactErrorMsg);
    }

            OfferList offerList=response.getOfferList();

             if(offerList != null && offerList.getRecommendedOffers() != null)
            {
              return offerList.getRecommendedOffers()[0];
            }
  }
    catch(Exception e)
  {
      interactErrorMsg.append(e.getMessage());
      e.printStackTrace();
  }
       return null;
 }

    private void printDetailMessageOfWarningOrError(String command, Response response,
        StringBuilder interactErrorMsg)
    {
      StringBuilder sb = new StringBuilder();
        sb.append("Calling "+command).append("<br>");
        AdvisoryMessage[] messages = response.getAdvisoryMessages();

        for(AdvisoryMessage msg : messages)
        {
           System.out.println(msg.getMessage());
          sb.append(msg.getDetailMessage());
          sb.append("<br>");
        }
        interactErrorMsg.append(sb.toString());
    }
%>
```

第 17 章 Interact 与 Digital Data Exchange 集成

使用 Digital Data Exchange 时，您的 Web 站点可链接到 Interact 以提供强大的 omni 渠道执行引擎，此引擎为最佳渠道提供最适合的商品并从商品反馈演进（学习）以持续提高市场营销效率。

如果市场营销团队将 Interact 用于 omni 渠道商品管理且希望将这些个性化智能商品扩展到 Web 站点，那么可使用此工具。

IBM Digital Data Exchange 通过实时数据联合 API 和企业级标记管理解决方案，将 IBM 和第三方市场营销解决方案与数字客户精辟见解集成。

不使用 IBM Digital Data Exchange 时，市场营销人员依赖于 IT 将 Interact 链接到其 Web 站点，并从各个 Web 页面调用 Interact API。使用 IBM Digital Data Exchange，市场营销人员可绕过 IT，直接通过 IBM Digital Data Exchange 在各个 Web 页面上包含 IBM Digital Data Exchange 标记。

先决条件

您必须先确保满足本部分中描述的先决条件，然后才能使用 Interact 与 Digital Data Exchange 的集成。

确保满足以下先决条件。

- 熟悉 Interact JavaScript API，如《管理员指南》中其他位置和联机帮助中所记录。
- 熟悉 Digital Data Exchange 标记和页面组。
- 具有有效 Digital Data Exchange 帐户。
- 您的 `interactapi.js` 文件已公共托管，以便可在**供应商**设置中访问此文件。

通过 IBM Digital Data Exchange 将 IBM Interact 与 Web 站点集成

使用这些步骤通过 Digital Data Exchange 将 Interact 与 Web 站点集成。

过程

1. 指定 `Interactapi.js` 文件的位置。
   a. 浏览到 Digital Data Exchange 中的**供应商** > **供应商设置**。
   b. 从**供应商**下拉列表选择 IBM Interact。
   c. 在**库路径**中，输入托管 `Interactapi.js` 的 URL。请勿在此 URL 中包含协议（http 或 https）。
   d. 在**公共 Rest servlet 的路径**中，添加 Rest servlet 的路径。
2. 浏览到 Digital Data Exchange 中的**管理** > **全局设置**，以指定要用作**唯一页面标识**中页面标识的对象名称。例如，可将对象名称设置为 `digitalData.pageInstanceID`。

3. 在希望 Digital Data Exchange 向其插入标记的 Web 页面上包含 eluminate.js 文件和标识。应该为每个 Web 页面指定一个唯一标识，以便 Digital Data Exchange 可区分各个页面。

   例如，可以向主页添加以下脚本。

```
<!-- Setting Page Identifier -->
          <script>
                  digitalData={pageInstanceID:"INTERACT_HomePage"};
          </script>

<!-- Including eluminate script -->
         <script type="text/javascript" src="http://libs.
                  coremetrics.com/eluminate.js">
         </script>
         <script type="text/javascript">
                  cmSetClientID("51310000|INTERACTTEST",false,"data.
                  coremetrics.com",document.domain);
         </script>
```

4. 在 Digital Data Exchange 中，创建标记、代码段、函数和要添加到 Web 页面的其他项。
5. 创建页面组以定义希望在每个页面上填充的内容。

   请参阅《 IBM Digital Data Exchange 用户指南》，以获取更多信息。

Digital Data Exchange 中的 Interact 标记

使用缺省 Digital Data Exchange 标记定义适用于 Web 页面的标记的变体，在这些页面中，从不同位置表示数据。定义这些标记后，即会将其添加到 Interact 标记列表。标记可能不具有要定义的字段，或可能不具有必填标记字段，可直接使用。

在 Digital Data Exchange 的**标记**下提供了以下 Interact 标记。

- 结束会话
- 获取商品
- 装入库
- 发布事件
- 设置受众
- 启动会话

要使用 Interact 标记，请编辑标记来定义每个 Interact 标记的"标记字段"、"方法"、"对象名称"、"数据类型"和"修饰符"。

"发布事件"、"设置受众"和"启动会话"标记接受定制标记字段。使用"标记字段添加"图标，然后单击"编辑"图标以定义定制参数。此过程与任何参数定义相同，不同之处在于在此过程中，参数的名称可编辑，并且必须包含参数名称、冒号和参数数据类型。可以使用向上箭头和向下箭头修改标记中的定制参数顺序。

还可以将标记绑定到 JavaScript 函数或 HTML 对象，从而使这些标记在函数触发或发生 HTML 对象事件后触发。

有关如何定义、绑定和使用标记的更多信息，请参阅《 IBM Digital Data Exchange用户指南》。

有关 Interact 与 Digital Data Exchange 集成的详细用例，请参阅 https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration。

结束会话

"结束会话"标记用于标记 Web 会话的结束。

以下标记字段可用于"结束会话"标记。

表 *36.* "结束会话"标记

标记字段	描述
*会话标识	标识会话标识。
成功时回调函数名称	定义结束会话方法成功时要调用的函数的名称。
失败时回调函数名称	定义结束会话方法失败时要调用的函数的名称。

需要标记了 * 的任何**标记字段**。

获取商品

使用"获取商品"标记从运行时服务器请求商品。

以下标记字段可用于"获取商品"标记。

表 *37.* "获取商品"标记

标记字段	描述
*会话标识	标识会话标识。
*交互点名称	标识此方法引用的交互点的名称。此名称必须与交互式渠道中定义的交互点的名称完全匹配。
*请求的数目	标识请求的商品的数目。
成功时回调函数名称	定义获取商品方法成功时要调用的函数的名称。
失败时回调函数名称	定义获取商品方法失败时要调用的函数的名称。

需要标记了 * 的任何**标记字段**。

应该将"获取商品"标记分配给其容器设置为 `Default` 的页面组。

装入库

"装入库"标记用于在页面的标头部分中装入 Interact JavaScript 库。

"装入库"标记不具有参数。此标记采用供应商设置中的**库路径**中的库位置。它应该包含在使用设置为 `Head` 的容器的页面组中，且应该在具有 Interact 标记的每个页面上运行。

要点：如果不包含"装入库"标记，那么其他标记都无效。如果不包含此标记，那么不会装入 interact.js。

发布事件

使用"发布事件"标记执行交互式渠道中定义的任何事件。

以下标记字段可用于"发布事件"标记。

表 *38.* ″发布事件″标记

标记字段	描述
*会话标识	标识会话标识。
*事件名称	标识事件的名称。事件的名称必须与交互式渠道中定义事件名称相匹配。此名称不区分大小写。
成功时回调函数名称	定义发布事件方法成功时要调用的函数的名称。
失败时回调函数名称	定义发布事件方法失败时要调用的函数的名称。

需要标记了 * 的任何**标记字段**。

可以使用定制标记字段功能添加可选参数。定制参数名称必须包含参数名称、冒号和数据类型。

设置受众

使用"设置受众"标记为访问者设置受众标识和级别。

以下标记字段可用于"设置受众"标记。

表 *39.* ″设置受众″标记

标记字段	描述
*会话标识	标识会话标识。
*受众标识	标识受众标识。名称必须与包含受众标识的任何表的物理列名称匹配。受众标识不能包含 17 个有效数字。如果受众标识长于 17 个有效数字，那么必须分区或必须将受众标识更改为字符串。
*受众级别	定义受众级别。
成功时回调函数名称	定义设置受众方法成功时要调用的函数的名称。
失败时回调函数名称	定义设置受众方法失败时要调用的函数的名称。

需要标记了 * 的任何**标记字段**。

可以使用定制标记字段功能添加可选参数。定制参数名称必须包含参数名称、冒号和数据类型。

启动会话

"启动会话"标记用于创建和定义 Web 会话。

以下标记字段可用于"启动会话"标记。

表 *40.* ″启动会话″标记

标记字段	描述
*会话标识	标识会话标识。

表 40. "启动会话"标记 (续)

标记字段	描述
*交互式渠道	定义此会话引用的交互式渠道的名称。此名称必须与 Campaign 中定义的交互式渠道的名称精确匹配。
*受众标识	标识受众标识。名称必须与包含受众标识的任何表的物理列名称匹配。
*受众级别	定义受众级别。
*依赖于现有会话	定义此会话是使用新会话还是现有会话
*调试	启用或禁用调试信息。
成功时回调函数名称	定义启动会话方法成功时要调用的函数的名称。
失败时回调函数名称	定义启动会话方法失败时要调用的函数的名称。

需要标记了 * 的任何**标记字段**。

可以使用定制标记字段功能添加可选参数。定制参数名称必须包含参数名称、冒号和数据类型。

应该将"启动会话"标记分配给其容器设置为 Default 的页面组。

示例标记设置

此示例显示"启动会话"、"发布事件"、"获取商品"和"结束会话"标记设置的简单配置。

针对任何标记，可通过 cookie 方法从 cookie 获取标记字段值，或通过 javascriptobject 方法从 JavaScript 对象获取标记字段值。

这些标记支持此简单示例未显示的其他参数。您可以在《 IBM Digital Data Exchange 用户指南》中找到有关其他参数的更多信息。

有关 Interact 与 Digital Data Exchange 集成的详细用例，请参阅 https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration。

示例"启动会话"标记设置

单击**标记** > **IBM 标记** > **IBM Interact** > **类型：启动会话**，以创建"启动会话"标记。使用以下设置编辑此标记。

"会话标识"设置

- **方法**：Constant
- **常量**：5555
- **数据类型**：String
- **修饰符**：<null>

"交互式渠道"设置

- **方法**：Constant
- **常量**：WSCDemo

- **数据类型**：String
- **修饰符**：<null>

"受众标识"设置

- **方法**：Constant
- **常量**：USERS_ID,2002,numeric
- **数据类型**：String
- **修饰符**：<null>

"受众级别"设置

- **方法**：Constant
- **常量**：WSCUserId
- **数据类型**：String
- **修饰符**：<null>

"依赖现有会话"设置

- **方法**：Constant
- **常量**：False
- **数据类型**：Boolean
- **修饰符**：<null>

调试

- **方法**：Constant
- **常量**：True
- **数据类型**：Boolean
- **修饰符**：<null>

"成功时回调函数名称"设置

- **方法**：Unassigned
- **值**：<null>

"失败时回调函数名称"设置

- **方法**：Unassigned
- **值**：<null>

示例"获取商品"标记设置

单击**标记** > **IBM 标记** > **IBM Interact** > **类型：获取商品**，以创建"获取商品"标记。使用以下设置编辑此标记。

"会话标识"设置

- **方法**：Constant
- **常量**：5555
- **数据类型**：String
- **修饰符**：<null>

"交互点名称"设置

- **方法**：Constant
- **常量**：AuroraHomepageHeaderBannerLeft
- **数据类型**：String
- **修饰符**：<null>

"请求的数目"设置

- **方法**：Constant
- **常量**:1
- **数据类型**：integer
- **修饰符**：<null>

"成功时回调函数名称"设置

- **方法**：Constant
- **常量**：onOfferReturnSuccess
- **数据类型**：string
- **修饰符**：<null>

"失败时回调函数名称"设置

- **方法**：Constant
- **常量**：onOfferReturnError
- **数据类型**：string
- **修饰符**：<null>

示例"发布事件"标记设置

单击**标记** > **IBM 标记** > **IBM Interact** > **类型：发布事件**，以创建"发布事件"标记。使用以下设置编辑此标记。

"会话标识"设置

- **方法**：Constant
- **常量**：5555
- **数据类型**：String
- **修饰符**：<null>

"事件名称"设置

- **方法**：Constant
- **常量**：ACCEPTOFFER
- **数据类型**：String
- **修饰符**：<null>

"成功时回调函数名称"设置

- **方法**：Constant
- **常量**：onSuccessTestFunction
- **数据类型**：String

- **修饰符**：<null>

"失败时回调函数名称"设置

- **方法**：Constant
- **常量**：onErrorTestFunction
- **数据类型**：String
- **修饰符**：<null>

"其他参数字段"设置

- **标记字段**：UACIOfferTrackingCode:string
- **方法**：JavaScriptObject
- **对象名称**：oa.treatmentCode
- **数据类型**：String
- **修饰符**：<null>

示例"结束会话"标记设置

单击**标记** > **IBM 标记** > **IBM Interact** > **类型：结束会话**，以创建"结束会话"标记。使用以下设置编辑此标记。

"会话标识"设置

- **方法**：Constant
- **常量**：5555
- **数据类型**：String
- **修饰符**：<null>

"成功时回调函数名称"设置

- **方法**：Unassigned
- **值**：<null>

"失败时回调函数名称"设置

- **方法**：Unassigned
- **值**：<null>

示例函数

针对用于"成功时回调函数名称"和"失败时回调函数名称"设置的函数，仅当 Web 页面上已存在此函数且创建新标记时，必须指定函数名称。

还可使用 Digital Data Exchange 实用程序来创建函数，并将其添加到 Web 页面。

以下示例说明如何在 Web 页面上显示从 Interact 返回的商品。必须在此页面上包含此脚本，或使用 Digital Data Exchange 代码片段来插入此脚本。

```
<script>
oa = {treatmentCode: ""};
function acceptOffer(treatmentCode) {
oa.treatmentCode = treatmentCode;
}
function onOfferReturnSuccess(response) {
```

```
var offer = response.offerList[0].offers[0];
var attributes = offer.attributes;
var offerText = "";
var offerLinkURL = "#";
for(var i = 0; i<attributes.length; i++)
{
if(attributes[i].n == "OfferTerms")
{
offerText = attributes[i].v;
}
else if(attributes[i].n == "OfferLinkURL")
{
offerLinkURL = attributes[i].v;
}
}

var link = "<a href=\"'+offerLinkURL+"\" onclick=\"acceptOffer
('"+offer.treatmentCode+"')\">"+offerText+"</a>";
document.getElementById("offerContainer").innerHTML="
<div style=\"text-align:center;padding:
10px 0;background-color:#f5f5f5;\">"+link+"</div>";
}
function onOfferReturnError(response) {
(JSON.stringify(response));
}
</script>
```

验证集成配置

使用 Digital Data Exchange 测试工具和 Interact.log 文件，对任何配置问题进行故障诊断。

您可以使用 Digital Data Exchange 测试工具检查百科全书，以了解配置是否按预期工作。要打开测试工具，请单击 Digital Data Exchange 中的**部署** > **测试工具**。

请参阅《 IBM Digital Data Exchange 用户指南》，以获取有关测试工具的更多信息。

您可以查看 Interact.log 文件，以查看有关执行的各种 Interact API 调用的详细信息。向每个标记添加"成功时回调函数"和"失败时回调函数"，以调试各种调用。

第 18 章 配置触发式消息的网关

使用触发式消息网关，从入站渠道和出站渠道发送和接收商品信息。

可以针对触发式消息使用以下入站网关和出站网关。

- IBM Interact Inbound Gateway for IBM Universal Behavior Exchange
- IBM Interact Outbound Gateway for IBM Universal Behavior Exchange
- IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud
- IBM Interact Outbound Gateway for IBM Mobile Push Notification

有关更多信息，请参阅 https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20Triggered%20Messages。

使用 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange

要使用 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange，必须配置 Interact，配置 UBX 订户端点，并在 UBX 中创建端点和事件。

使用以下配置作为配置的示例。

可以从 http://www.ibm.com/support/fixcentral/swg/quickorder?parent=Enterprise%2BMarketing%2BManagement&product=ibm/Other+software/Unica+Interact&release=All&platform=All&function=fixId&fixids=IBM_Interact_OMO_Gateway_for_UBX_Subscriber_2.0&includeRequisites=1&includeSupersedes=0&downloadMethod=http&source=fc 下载订户网关。

为 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 配置 Interact

使用以下步骤来配置 Interact。

1. 在 **Interact | activityOrceshtrator | receivers** 配置属性中，添加新接收器。将**类型**设置为 IBMMQ 或定制。如果选择定制，请输入 **ClassName** 和 **ClassPath**。如果选择 IBMMQ，请保留 **ClassPath** 和 **ClassName** 为空白。
2. 添加接收器的 providerURL、queueManager、messageQueueName、authDS 和 asmUserFor...." 参数。
3. 在 **Interact | activityOrceshtrator | gateways** 配置属性中，添加新网关。将 **ClassPath** 设置为 OMO_InteractGateway_UBX.jar 文件位置的 URI，将 **ClassName** 设置为

   com.ibm.interact.offerorchestration.inboundgateway.ubx。UBXInboundGateway
4. 在入站网关的 UBX 文件夹下创建 Interactubx11 文件夹，并将属性文件复制到此新文件夹。文件夹名称应该与在 UBX 中创建的订户端点的名称匹配。

5. 在 interactEventNamemapping.properties 文件中，添加条目以将有效内容事件字段的值映射到 Interact 事件名称。例如，recommededOffers=recommendedOffers。
6. 在 interactEventPayloadMapping.properties 文件中，添加字段定义，同时这些参数的名称分别设置为 OMO-conf_inbound_UBX_interactEventNameMapping 和 OMO-conf_inbound_UBX_interactEventNameMapping。

   例如：

   ```
   [SessionID]=(String)interactprofileid
   [EventName]=(String)code
   [AudienceIDFieldNames]=(String)"CustomerID"
   [AudienceIDFieldValues]=(Numeric)interactprofileid
   [AudienceLevel]=(String)"Customer"
   [InteractChannel]=(String"UBX_MM"
   ```

7. 在 **Interact | activityOrceshtrator | gateways | [gatewayname] | Parameter Data** 下添加 Interactubx11/interactEven tNameMapping.properties 和 Interactubx11/interactEven tPayloadMapping.properties 的位置作为网关参数。
8. 创建交互式渠道并向交互式渠道添加事件。
9. 使用 recommendedOffers 事件添加触发式消息规则，并为此规则分配商品。
10. 部署交互式渠道。
11. 重新启动 Interact 服务器。
12. 使用 REST API 客户机将事件发布到 UBX。

    示例事件主体：

    ```
    {
      "channel" : "mobile",
      "identifiers" : [
        {
          "name" : "interactprofileid",
          "value" : "55"
        }
      ],
    "events" : [
      {
        "code" : "recommendedOffers",
        "timestamp" : "2015-12-28T20:16:12Z"
       }
      ]
    }
    ```

13. 检查 Interact 日志以查看是否触发了触发式消息事件。

配置 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点

这是可用作示例的样本端点。

还应该使用这些指示信息来完成以下配置。

- 带有 IBM MQ 的 UBX 端点
- 端点 ubxInboundEndpoint.properties 文件
- 端点 inboundProducerNameConfig.properties 文件
- 端点 inboundQueueNameConfig.properties 文件
- 端点 log4j.properties 文件

部署 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 和端点

1. 下载 IBM_Interact_OMO_Gateway_for_UBX_Subscriber_2.0.zip 并将其解压缩到 Interact 运行时服务器上 Interact 的安装目录。
2. 下载 IBM_Interact_OMO_Endpoint_for_UBX_Subscriber_2.0.zip 并将其解压缩到公众可访问并启用了 JavaEE 的应用程序服务器或 Web 服务器上的任何目录（例如 c:\ubxInboundEndpoint）。此服务器将数据发布到 Interact 入站 JMS 队列，以供 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 稍后使用。

配置 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange Interact Inbound Gateway 端点

IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点将配置为接受来自 Universal Behavior Exchange 的请求并将其发送至 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange。

要配置 Universal Behavior Exchange Subscriber Gateway 端点，必须完成下列任务

1. 新的 Java 系统属性 (-DubxInboundEndpointConfigPath) 需要通过编辑 Web 服务器中或应用程序服务器的管理控制台中的配置文件进行配置。-D 属性应该指向服务器中的端点安装目录。此目录包含目标 JMS 队列的配置文件以及该端点的各种记录级别。例如 -DubxInboundEndpointConfigPath=c:\ubxInboundEndpoint。
2. 按照 Web 服务器或应用程序服务器文档中的描述，从安装目录部署 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点 Web 归档文件 (ubxInboundEndpoint.war)。

要验证是否正确安装了端点，请在任何浏览器中输入以下地址并寻找消息 UBX End Point is UP。

```
http://[Server]:[Port]/[ContextRoot]/UBXEndPoint
```

注：您应该保护公众可访问的 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点，方法是添加必要的防火墙规则，以仅接受来自 IBM Universal Behavior Exchange Server 的 HTTP 请求。

例如，可以使用下列指示信息在 WebSphere Application Server 上配置和部署 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点。

1. 打开管理控制台。
2. 选择**服务器** > **（展开服务器类型）** > **server_name** > **（展开 Java™ 和进程管理）** > **进程定义** > **Java 虚拟机**。
3. 在通用 JVM 参数中，添加 property-DubxInboundEndpointConfigPath=<Universal Behavior Exchange Subscriber Gateway endpoint install dir on the application server>。例如，添加属性 -DubxInboundEndpointConfigPath=C:\ubxInboundEndpoint。
4. 单击**确定**以将更改保存到主配置。
5. 重新启动应用程序服务器。

在 WebSphere Application Server 中部署端点。

1. 登录管理控制台。

2. 浏览到**应用程序** > **应用程序类型** > **Websphere 企业应用程序**。单击**安装**。
3. 使用**准备应用程序安装**选项来找到要安装的端点 WAR 文件 (ubxInboundEndpoint.war)，然后单击**下一步**。
4. 在后续页面中单击**下一步**以到达**映射 Web 模块的上下文根**。
5. 使用**映射 Web 模块的上下文根**来找到"上下文根"并将值更改为 /UBXEndPoint，此位置成为上下文根。单击**下一步**。
6. 单击**完成**。
7. 一旦应用程序完成了安装，单击**保存**以将更改保存到主配置。
8. 返回到列示的已安装应用程序，选中 ubxInboundEndpoint_war，然后单击**启动**以进行装入。

配置带有 IBM MQ 的 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点（可选）

缺省情况下，IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点使用 ActiveMQ。使用下列指示信息来配置带有 IBM MQ 的端点。

准备 IBM MQ JAR 文件：

运行该端点的客户机必须具有某些可用的 IBM MQ JAR 文件，连接工厂才能工作。

如果 IBM MQ 已安装在端点机器上，那么您需要的 JAR 文件已与 IBM MQ 安装版本打包在一起。将以下两个 JAR 文件添加到系统级别 CLASSPATH 环境变量。在 Windows 中，安装 IBM MQ 时，会自动将两个 JAR 文件添加到类路径。

```
[MQ_HOME]\java\bin\com.ibm.mq.jar
[MQ_HOME]\java\bin\com.ibm.mqjms.jar
```

如果 IBM MQ 未安装在机器上，那么您应该改为将 com.ibm.mq.allclient.jar 和 jms.jar 从 MQ 服务器复制到端点服务器，并手动将它们添加到 CLASSPATH。

有关安装或重新定位 IBM MQ JAR 文件的更多信息，请参阅 http://www.ibm.com/support/docview.wss?uid=swg21376217。

应用程序服务器需要运行 Java 1.7 或更高版本，因为 IBM MQ V8 JAR 文件不支持 Java 1.6。

WebSphere Application Server 以预先打包的形式附带了 IBM MQ 支持，不需要任何其他 JAR 文件。

配置端点。

1. 转至 <endpoint install dir on the application server> 目录。
2. 备份或重命名 ubxInboundEndpoint-spring.xml 和 ubxInboundEndpoint.properties。
3. 浏览到 IBMMQ 子目录。它将包含以上文件的备用版本。
4. 将 MQ 服务器连接信息添加到 ubxInboundEndpoint.properties 的此版本。
5. 将 ubxInboundEndpoint-spring.xml 和 ubxInboundEndpoint.properties 从 /ubxInbdoundEndpoint/IBMMQ 复制到 main/ubxInboundEndpoint 目录。

配置 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点 ubxInboundEndpoint.properties 文件

使用 ubxInboundEndpoint.properties 文件来配置要将 Universal Behavior Exchange 事件有效内容发送至 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 的哪个位置。ubxInboundEndpoint.properties 文件位于 <gateway endpoint install dir on the application server> 目录中。

jmsBrokerUrl

必需 - 生产程序在其中写入数据的 JMS 队列信息。

jmsMaximumRetries

必需 - 将消息发送至 JMS 队列的最大重试次数。

jmsRetryDelay

必需 - 重新传递延迟（以毫秒计）。

maximumEndPointThreadPoolSize

必需 - 用于处理 IBM Universal Behavior Exchange 事件数据并将这些数据写入至 JMS 队列的线程池的最大线程数。此整数定义该线程池的大小。

clientIDFieldName

可选 - 客户机标识的有效内容中使用的字段名称（子类别）。此程序在同一产品的多个实例上运行时，将使用子类别。例如：clientIDFieldName=clientID

要使此文件中的任何更改生效，需要在 Web 服务器或应用程序服务器中重新启动网关端点 Web 应用程序 (ubxInboundEndpoint.war)。

配置 IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点 inboundProducerNameConfig.properties 文件（可选）

IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点通过将事件写入至 JMS 队列来将事件发送至 Interact。缺省事件消息使用生产程序名称值 UBX。使用 inboundProducerNameConfig.properties 文件来覆盖基于有效内容中 UBX 源字段值的生产程序名称。此名称通常是 UBX 端点名称。inboundProducerNameConfig.properties 文件位于 <gateway endpoint install dir on the application server> 目录中。

SOURCE.{UBX source name}={producer name}

示例：SOURCE.CustomerAEndpoint=UBX-CustomerAEndpoint。

要使此文件中的任何更改生效，需要在 Web 服务器或应用程序服务器中重新启动网关端点 Web 应用程序 (ubxInboundEndpoint.war)。

配置网关端点 inboundQueueNameConfig.properties 文件（可选）

IBM Interact Inbound Gateway for IBM Universal Behavior Exchange 端点通过将事件写入至 JMS 队列来将事件发送至 Interact。缺省队列名称与生产程序名称相同。使用 inboundQueueNameConfig.properties 文件将缺省 JMS 队列名称覆盖为生产程序名称。除非在 inboundQueueNameConfig.properties 文件中覆盖了缺省生产程序名称，否则该名称是 UBX。inboundProducerNameConfig.properties 文件位于 <gateway endpoint install dir on the application server> 目录中。

{producer name}={JMS queue name}

```
Example:
UBX=UBXInboundQueue.
UBX-CustomerAEndpoint=UBX-CustomerAEndpointQueue
```

要使此文件中的任何更改生效，需要在 Web 服务器或应用程序服务器中重新启动网关端点 Web 应用程序 (ubxInboundEndpoint.war)。

配置网关端点 log4j.properties 文件

使用 log4j.properties 文件来配置端点的不同记录级别。log4j.properties 文件位于 <gateway endpoint install dir on the application server> 目录中。

描述

相应地设置 log4j.logger.com.ibm.x1solution.jms.producer、log4j.logger.com.ibm.web.offerorchestration.inbound.common 和 log4j.logger.com.ibm.web.offerorchestration.inbound.ubx 的记录级别。

配置 interactEventNameMapping.properties 文件

使用此文件将 interactEventPayloadMapping.properties 文件中定义的有效内容事件字段的值作为 [EventName] 映射到 Interact 事件名称。回退是使用事件名称，因为它包含在 Universal Behavior Exchange 事件有效内容中。interactEventNameMapping.properties 文件位于 <Install dir>\conf\inbound\UBX 目录中。

{UBX event name}={Interact event name}

示例：matchedIdentity=recommendedOfferEven

如果需要对特定源中的有效内容数据进行支持，那么也可以将此文件放置在 <Install dir>\conf\inbound\UBX\{source} 目录中。source 的值应该与 Universal Behavior Exchange 事件有效内容中 source 字段的值（通常是 Universal Behavior Exchange 端点名称）匹配。如果需要对使用特定版本的数据进行支持，那么也可以将此文件放置在 <Install dir>\conf\inbound\UBX\{source}\version-{version} 目录中。version 的值应该与 Universal Behavior Exchange 事件有效内容中 version 字段的值匹配。要支持多个 Universal Behavior Exchange 实例数据，也可以将此文件放置在 <Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID} 目录中。clientID 的值应该与 Universal Behavior Exchange 事件有效内容中 clientID 的值匹配。

配置 interactEventPayloadMapping.properties 文件

使用 interactEventPayloadMapping.properties 文件将入站字段映射到 Interact API 参数。interactEventPayloadMapping.properties 文件位于 <Install dir>\conf\inbound\UBX 目录中。

Interact API 参数：值必须以字段类型定义开头，后跟静态值（当该值用双引号引起来时）或有效内容数据中的字段名称。(FIELD_TYPE)"STATIC_VALUE" 或 (FIELD_TYPE) PAYLOAD_FIELD_NAME。FIELD_TYPE 可以是 String、Numeric 或 DateTime。

```
Example:
[SessionID]=(String)interactprofileid
[EventName]=(String)code
```

```
[AudienceIDFieldNames]=(String)"change_me"
[AudienceIDFieldValues]=(String)interactprofileid
[AudienceLevel]=(String)"change_me"
[InteractChannel]=(String)"change_me"
```

事件数据：这些属性用来映射可以在出站渠道通信中使用的事件属性。左边包含您在出站渠道通信中使用的变量名称。

值必须以字段类型定义开头，后跟静态值（当该值用双引号引起来时）或有效内容数据中的字段名称。(FIELD_TYPE)"STATIC_VALUE" 或 (FIELD_TYPE) PAYLOAD_FIELD_NAME。FIELD_TYPE 可以是 String、Numeric 或 DateTime。

如果需要对特定源中的有效内容数据进行支持，那么也可以将此文件放置在 <Install dir>\conf\inbound\UBX\{source} 目录中。source 的值应该与 Universal Behavior Exchange 事件有效内容中 source 字段的值（通常是 Universal Behavior Exchange 端点名称）匹配。如果需要对使用特定版本的数据进行支持，那么也可以将此文件放置在 <Install dir>\conf\inbound\UBX\{source}\version-{version} 目录中。version 的值应该与 Universal Behavior Exchange 事件有效内容中 version 字段的值匹配。要支持多个 Universal Behavior Exchange 实例数据，也可以将此文件放置在 <Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID} 目录中。clientID 的值应该与 Universal Behavior Exchange 事件有效内容中 clientID 的值匹配。

在 UBX 中创建端点和事件

这是可用作示例的样本端点和事件。

使用以下步骤在 UBX 中创建端点和事件。

1. 使用 REST API 客户机将请求发布到 UBX。
2. 使用 JSON 在 UBX 中注册端点。请参阅以下示例。

```
Method Call: PUT
URL: https://ubx-qa1-api.adm01.com/v1/endpoint
Headers:
Content-Type: application/json
Accept-Charset: UTF-8
Authorization: Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
(Note: This is the Auth Key generated from the UBX UI.)
Body
 {
   "name":"Interactubxdk1",
   "description":"Interactubxdk1",
   "providerName":"IBM", "
   "url":"http://169.38.71.122:9081/ubxEndPoint/UBXEndPoint",
   "endpointTypes":{
     "event":{
       "source":{
         "enabled":true
       },
       "destination":{
         "enabled":true,
         "url":"http://169.38.71.122:9081/UBXEndPoint/UBXEndPoint",
         "destinationType":"push"
        }
      }
},
"marketingDatabasesDefinition":{
  "marketingDatabases":[
    {
```

```
      "name":"IDSync",
      "identifiers":[
        {
      "name":"interactprofileid",
      "type":"INTERACTID"
          }
        ]
     }
  ]
}
}
```

3. 使用 JSON 在 UBX 中注册事件类型。请参阅以下示例。

```
UBX 中针对 Interact 事件的事件注册
Method Call: POST
URL: https://ubx-qa1-api.adm01.com/v1/eventtype

Headers:
Content-Type: application/json
Accept-Charset: UTF-8
Authorization: Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
Note: This is the Auth Key generated from the UBX UI.)
Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
Body
{
   "name": "recommendedOffers",
   "description": "recommended offers by OMO",
   "code": "recommendedOffers"
}
```

4. 使用 JSON 将事件发布到 UBX。请参阅以下示例。

```
{
  "channel" : "mobile",
  "identifiers" : [
    {
      "name" : "interactprofileid",
      "value" : "55"
    }
  ],
"events" : [
  {
    "code" : "recommendedOffers",
    "timestamp" : "2015-12-28T20:16:12Z"
  }
 ]
}
```

使用 IBM Interact Outbound Gateway for IBM Universal Behavior Exchange

要使用 IBM Interact Outbound Gateway for IBM Universal Behavior Exchange，必须配置 Interact、UBX 和该网关。

使用以下配置作为配置的示例。

如果将 UBX 用作出站渠道，那么 Interact 充当端点的发布程序类型，此发布程序用于将事件发布到 UBX。从 UBX，这些事件可发送到订户。

开始配置之前，请求主机的出站访问权。需要启用对主机的网络访问权。

可以从 http://www.ibm.com/support/fixcentral/swg/quickorder?parent=Enterprise%2BMarketing%2BManagement&product=ibm/Other+software/Unica+Interact&release=All&platform=All&function=fixId&fixids=IBM_Interact_OMO_Gateway_for_UBX_Publisher_2.0&includeRequisites=1&includeSupersedes=0&downloadMethod=http&source=fc 下载该网关。

在 UBX 中注册端点和事件

1. 从 UBX，浏览到**端点**选项卡。单击**注册新端点**以获取认证密钥。从 UBX 生成的认证密钥应该用于发布程序端点和添加事件。针对订户端点，应该从 UBX 生成新认证密钥。记录此密钥。
2. 注册发布程序端点。
   a. 打开 REST API 客户机工具。
   b. 选择 PUT 作为方法。
   c. 将头作为以下内容传递

      ```
      Content-Type : application/json
      Accept-Charset : UTF-8
      Authorization : Bearer 520301d7-7855-4ea7-b19d-0b395c1e6ae4
      (authKey generated in UBX)
      ```

   d. 将 URL 作为以下内容传递

      ```
      URL: https://ubx-qa1-api.adm01.com/v1/endpoint
      ```

   e. 针对主体，传递发布程序端点的相应名称。

      例如：

      ```
      {
          "name":"Interact_Publisher",
          "description":"Endpoint for server created on 30thJan",
          "providerName":"IBM",    "url":"",
          "endpointTypes":{
              "event":{
              "source":{
                 "enabled":true
              }
          }
      },
      "marketingDatabasesDefinition":{
          "marketingDatabases":[
            {
                "name":"IDSync",
                "identifiers":[
                  {
                "name":"interactprofileid",
           "type":"INTERACTID"
              }
            ]
          }
         ]
      }
      }
      ```

3. 注册事件。记录主体中传递的 [Event] 代码。这需要在 ubxContentMapping.properties 文件中进行映射。该值区分大小写。
   a. 打开 REST API 客户机工具。
   b. 选择 POST 作为方法。
   c. 传递在先前步骤中用于端点的相同头。

d. 将 URL 作为以下内容传递

```
URL: https://ubx-qa1-api.adm01.com/v1/eventtype
```

e. 针对主体，传递事件的相应名称。

例如：

```
{
   "name": "recommendedOffer",
   "description": "recommended
      contact frm UBX",    "code":
      "recommendedOffer"}
```

注：必须在 ubxContentMapping.properties 文件中映射传递的事件代码。事件代码区分大小写。

4. 添加订户端点。
   a. 打开 REST API 客户机工具。
   b. 选择 PUT 作为方法。
   c. 传递在先前步骤中用于端点的相同头。
   d. 为注册订户端点，请在 UBX 中创建新认证密钥。
   e. 将 URL 作为以下内容传递

```
URL: https://ubx-qa1-api.adm01.com/v1/endpoint
```

   f. 针对主体，传递发布程序端点的相应名称。

例如：

```
{
   "name":"UBX_Subscriber",
   "description":"UBX_Subscriber for Subscribing Events ",
   "providerName":"IBM",
   "url":"http://ubxeventconsumer.mybluemix.net/ubxeventconsumer",
   "endpointTypes":{
      "event":{
        "source":{
          "enabled":true
        },
      "destination":{
        "enabled":true,
         "url":"http://ubxeventconsumer.mybluemix.net
          /ubxeventconsumer",
      }
   }
},
"marketingDatabasesDefinition":   {
  "marketingDatabases":[
    {
       "name":"IDSync",
       "identifiers":[
          {
       "name":"interactprofileid",
     "type":"INTERACTID"
         }
       ]
   }
 ]
}
```

5. 添加发布程序以及订户端点和事件后，必须在 UBX 预订从发布程序到订户的事件。
   a. 在 UBX 中，单击**事件**选项卡上的**预订事件**。

b. 选择事件和目标。

c. 单击**预订**。

配置 Interact 和该网关

1. 在 **Interact | triggeredMessage | gateways** 配置属性下添加 UBX 网关。将 **ClassPath** 设置为 file:///root/opt/OMO/lib/OMO_OutboundGateway_UBX.jar，将 **ClassName** 设置为

   com.ibm.interact.offerorchestration.outboundgateway.ubx。
   UBXOutboundGateway

2. 将 OMO_OutboundGateway_UBX.zip 文件解压缩到主机上，并从解压缩路径指向 UBX jar。

3. 在 **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data** 下将 OMO-conf_outbound_common_httpConnectionConfig 作为参数添加。将**值**设置为 file:///opt/Interact<version>/Interact/OMO/conf/outbound/common/httpConnectionConfig.properties。这是 Interact 安装目录。网关安装程序将网关目录下载到 Interact 安装目录。

   在 Interact 文件夹的 httpConnectionConfig.properties 文件中，指定超时。

   例如：

   connectTimeoutMs=180000

4. 在 **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data** 下将 OMO-conf_outbound_ubx_ubxConfig 作为参数添加。将**值**设置为 Interact 文件夹中 ubxConfig.properties 文件的路径。

   在 ubxConfig.properties 文件中，指定 ubxURL、authKey 和 interactProfileIdFieldName。

   例如：

   authKey=912586bf-190d-48f9-8488-26f1bf532ef3
   [Auth Key used to register publisher endpoint and event in UBX]
   interactProfileIdFieldName=interactprofileid
   [Field name from the ubxContentMapping.properties file]

5. 在 **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data** 下将 OMO-conf_outbound_ubx_ubxContentAdditionalAttributes 作为参数添加。将**值**设置为 Interact 文件夹中 ubxContentAdditionalAttributes.properties 文件的路径。

6. 在 **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data** 下将 OMO-conf_outbound_ubx_ubxContentMapping 作为参数添加。将**值**设置为 Interact 文件夹中 ubxContentMapping.properties 文件的路径。

   在 ubxContentMapping.properties 文件中，更新 interactprofileid 和 eventName 的值。

   您可以通过三种格式传递事件名称：值采用双引号时，为静态值；值采用 offer.offerAttributeName 格式时，将映射到商品属 offerAttributeName；值采用 profile.profileAttributeName 格式时，将映射到概要文件属性 profileAttributeName。"事件名称"值应该匹配用于在 UBX 中注册事件的代码该值区分大小写。

例如：

```
eventName="abandoned_shopping_carts"
eventName=offer.Card
eventName=profile.EMAIL
```

7. 在 **Interact | triggeredMessage | channel** 配置属性下添加渠道。
8. 在 **Campaign | partitions | partition [n] |Interact | outboundChannels** 下定义设计时中相同渠道
9. 重新启动应用程序服务器。
10. 创建具有事件名称且使用在先前步骤中添加的渠道的触发式消息规则。
11. 部署交互式渠道。
12. 从 API 测试客户机，启动配置了触发式消息规则的交互式渠道以及可触发将商品发布到 UBX 的发布事件的会话。

使用 IBM Interact Outbound Gateway for IBM Mobile Push Notification

要使用此移动推送出站网关或发布程序网关，必须配置 Interact、IBM Marketing Cloud 和网关。

使用以下配置作为配置的示例。

可以从 https://www-945.ibm.com/support/fixcentral/swg/downloadFixes 下载此网关

配置 IBM Marketing Cloud

1. 确保具有带推送访问权的 IBM Marketing Cloud 帐户。另外，记录客户机标识、客户机密钥和刷新令牌。
2. 在**数据**选项卡上，创建新数据库。向数据库以及缺省字段添加新移动用户标识。
3. 在**搜索**选项卡上，按移动用户标识字段搜索。将鼠标悬停在第一个无电子邮件字段上。将在浏览器窗口底部看到收件人标识。将此收件人标识添加到 Interact 概要文件表。

配置 IBM Interact Outbound Gateway for IBM Mobile Push Notification

1. 从 https://www-945.ibm.com/support/fixcentral/swg/downloadFixes 下载和安装移动推送出站网关
2. 配置 `silverpopEngagePushConfig.properties` 文件。

   例如：

   ```
   OauthServiceURL=https://apipilot.silverpop.com/oauth/token
   pushServiceURL=https://apipilot.silverpop.com/rest/channels/push/sends
   ```

3. 配置 `silverpopEngagePushContentMapping.properties` 文件。

   例如：

   ```
   Interact Profile table attributes:
   appKey=appKey
   engageRecipientId=recipientId
   mobileUserId=mobileUserId
   deviceType=deviceType
   ```

```
Interact Offer attributes:
simpleSubject=simpleSubjectAttr
simpleMessage=simpleMessageAttr
simpleActionData=simpleActionDataAttr
simpleActionType=simpleActionTypeAttr
simpleActionLabel=simpleActionLabelAttr
personalizeAttributeList=personalizeAttributeList
contentId=ContentID
campaignId=campaignId
```

配置 Interact

1. 创建以下商品属性。

   ```
   simpleActionDataAttr: string
   simpleActionLabelAttr: String
   simpleActionTypeAttr: string
   simpleMessageAttr: string
   simpleSubjectAttr: string
   contentID: string
   campaignId=string
   personalizeAttributeList=string
   ```

2. 使用商品属性和以下商品值创建商品模板。

   ```
   simpleActionDataAttr: www.ibm.com
   simpleActionLabelAttr: Open URL
   simpleActionTypeAttr: url
   simpleMessageAttr: <Enter your message text here>
   simpleSubjectAttr: <Enter subject here>
   contentID: ID of the push message template that is created in Engage.
   PersonalizeAttributeList: A comma separated list of attribute name
   value pairs that you want to put in the personalizationDefaults
   section of the payload to be sent to Engage.
   ```

   使用 contentID 属性时，会忽略其他 simple.. 属性，因为会从 Engage 模板获取完整详细信息。

   示例 personalizedAttributeList

   ```
   personalizeAttributeList=discount=10,Offercost=20
   campaignId=campaignname that you want to use for this campaign.
   ```

3. 您的概要文件表具有以下列和值。

   ```
   appKey: gcsTQo6v79
   recipientId: 13472242
   deviceType: android or ios
   ```

4. 在 **Interact | triggeredMessage | gateways** 配置属性下添加网关。将 **ClassName** 设置为

   ```
   com.ibm.interact.offerorchestration.outboundgateway.silverpop.engage.push。
   SilverpopEngagePushOutboundGateway
   ```

   将 **ClassPath** 设置为 file://<EngagePushGateway_home_dir>/lib/OMO_OutboundGateway_Silverpop_Engage_Push.jar。

5. 在 **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data** 下将 OMO-silverpopEngagePushConfig 作为参数添加。将**值**设置为 silverpopEngagePushConfig.properties 文件的文件路径。

6. 在 **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data** 下将 OMO-silverpopEngagePushContentMapping 作为参数添加。将**值**设置为 silverpopEngagePushContentMapping.properties 文件的文件路径。

7. 在 **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data** 下将 OMO-conf_outbound_common_httpConnectionConfig 作为参数添加。将**值**设置为 httpConnectionConfig.properties 文件的文件路径。

   在 Interact 文件夹的 httpConnectionConfig.properties 文件中，指定超时。

   例如：

   ```
   connectTimeoutMs=6000
   ```

8. 在 **Interact | triggeredMessage** 下创建渠道和处理程序，并使用在此渠道上创建的 [Mobile_Push] 网关。此渠道用于触发式消息以发送推送消息。
9. 创建交互式渠道并将使用先前创建的商品的触发式消息添加到触发器规则。
10. 部署交互式渠道。
11. 从 API 测试客户机，针对配置了触发式消息规则的交互式渠道执行 startSession 以及执行 postEvent（触发向移动推送发布商品）
12. 检查 Interact 日志以确保是否成功发送了推送。状态码 202 表示传递成功。

使用 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud

可以将此集成与 Silverpop、Interact 和 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 配合使用，以将触发式电子邮件商品发送给客户。

确保满足以下先决条件。

- 创建具有电子邮件列的客户受众概要文件表。将此概要文件表用于您的交互式渠道。
- 请求为您的出站渠道启用对主机的访问权。
- 复制主机上的 OMO_OutboundGateway_Silverpop.zip 文件并将其解压缩

可以从 http://www.ibm.com/support/fixcentral/swg/quickorder?parent=Enterprise%2BMarketing%2BManagement&product=ibm/Other+software/Unica+Interact&release=All&platform=All&function=fixId&fixids=IBM_Interact_OMO_OutboundGateway_Silverpop_2.0&includeRequisites=1&includeSupersedes=0&downloadMethod=http&source=fc 下载该网关。

为网关集成添加分派器

分派器将商品添加到 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 的队列中，以便可以发送商品电子邮件。

关于此任务

必须添加分派器，才能使用 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud。

过程

1. 在配置属性中浏览到 **Interact | triggeredMessage | dispatchers | <dispatcherName>**。
2. 为分派器添加**新类别名称**。

3. 选择**类型**。您可以从 InMemoryQueue、JMSQueue 和 Custom 中进行选择。
4. 输入 **className**。
5. 输入 **classPath**。

为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 添加网关

在该集成中，网关通过电子邮件将合格商品发送给客户。

关于此任务

您必须为该集成添加网关。

注：Interact 不支持同一网关的多个实例。

过程

1. 在配置属性中浏览到 **Interact | triggeredMessage | gateways | <gatewayName>**。
2. 为网关添加**新类别名称**。
3. 将 **className** 设置为以下路径。

   ```
   com.ibm.interact.offerorchestration.outboundgateway.
   silverpop.SilverpopEmailOutboundGateway
   ```

4. 将 **classPath** 设置为解压缩的文件夹中出站网关 JAR 路径的位置。

   例如：

   ```
   file:///opt/OMO_SilverPop/OMO_OutboundGateway_Silverpop/lib/
   OMO_OutboundGateway_Silverpop.jar
   ```

5. 将下列参数添加到网关。

   ```
   OMO-conf_outbound_common_httpConnectionConfig
   OMO-conf_outbound_silverpop_silverpopConfig
   OMO-conf_outbound_silverpop_silverpopContentMapping
   deliveryTimeoutMillis
   ```

配置 OMO-conf_outbound_common_httpConnectionConfig 参数

必须为网关配置 OMO-conf_outbound_common_httpConnectionConfig 参数。

过程

1. 在配置属性中浏览到 **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | OMO-conf_outbound_common_httpConnectionConfig**。
2. 将**值**设置为 file:///opt/Interact<version>/Interact/OMO/conf/outbound/common/httpConnectionConfig.properties。这是 Interact 安装目录。Interact 安装程序会将 httpConnectionConfig.properties 文件下载到 Interact 安装目录。
3. 在 Interact 文件夹的 httpConnectionConfig.properties 文件中，指定超时。

   例如：

   ```
   connectTimeoutMs=60000
   ```

配置 OMO-conf_outbound_silverpop_silverpopConfig 参数

必须为网关配置 OMO-conf_outbound_silverpop_silverpopConfig 参数。

过程

1. 在配置属性中浏览到 **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | OMO-conf_outbound_silverpop_silverpopConfig**。
2. 将**值**设置为 OMO_OutboundGateway_silverpop 文件夹中的 silverpopConfig.properties 文件的路径。

   例如：

   file:///opt/OMO_SilverPop/OMO_OutboundGateway_Silverpop/conf/outbound/silverpop/silverpopConfig.properties
3. 在解压缩的 OMO_OutboundGateway_Silverpop.zip 文件夹的 silverpopConfig.properties 文件中，设置 OauthServiceURL、xmlAPIServiceURL、clientID、clientSecret 和 refreshToken 的值。向 Marketing Cloud 管理员咨询如何从 transact.xml 文件中获取特定于客户的值。

配置 OMO-conf_outbound_silverpop_silverpop ContentMapping 参数

必须为网关配置 OMO-conf_outbound_silverpop_silverpopContentMapping 参数。

过程

1. 在配置属性中浏览到 **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | OMO-conf_outbound_silverpop_silverpopContentMapping**。
2. 将**值**设置为 OMO_OutboundGateway_silverpop 文件夹中的 silverpopContentMapping.properties 文件的路径。
3. 在 OMO_OutboundGateway_Silverpop.zip 文件夹的 silverpopContentMapping.properties 文件中，为您的内容映射设置值。
   a. 设置 campaignId 属性。此属性的值是在商品模板中指定的商品属性名称。
   b. 设置 email 属性。此属性的值是概要文件表中的列名。在概要文件表中添加 email 列并指定电子邮件标识。这些标识是收件人的邮件标识。
   c. 在 additionalOfferPfAttributesUsedInEmail 中定义商品属性。此属性将根据商品模板来设置邮件模板所需的属性。您可以使用 additionalProfilePfAttributesUsedInEmail 来定义概要文件表中的字段。可以使用 * 以考虑所有商品属性和列值。

配置 deliveryTimeoutMillis 参数

要增加 Interact 服务器超时以与 Marketing Cloud 服务器进行连接，请设置 deliveryTimeoutMills 参数。

关于此任务

过程

1. 在配置属性中浏览到 **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | deliveryTimeoutMillis**。

2. 设置**值**。例如，您可以将**值**设置为 60000。这会将服务器超时增加至 60000 毫秒。

为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 添加渠道处理程序

在 Interact 运行时环境中添加渠道处理程序。

过程

1. 在配置属性中浏览到 **Interact | triggeredMessage | channels | <SilverpopChannelName> | <handlerName>**。
2. 为渠道处理程序添加**新类别名称**。
3. 设置您先前添加的分派器的名称。
4. 设置您先前添加的网关的名称。
5. 设置**方式**。如果选择了 Failover，那么仅当此渠道中定义的所有具有较高优先级的处理程序未能发送商品时，才会使用此处理程序。如果选择了 **Addon**，那么无论其他处理程序是否已成功发送商品，都会使用此处理程序。
6. 设置此处理程序的**优先级**。

为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 添加出站渠道

在 Interact 设计环境中添加出站渠道。

过程

1. 在配置属性中浏览到 **Campaign | partitions | partition[n] | Interact | outboundChannels**。
2. 为出站渠道添加**新类别名称**。
3. 为出站渠道添加**名称**。确保渠道名称与您在 **Interact | triggeredMessage | channels | <SilverpopChannelName>** 配置属性中添加的渠道名称相同。

为 IBM Interact Email (Transact) Outbound Gateway for IBM Marketing Cloud 配置交易邮件

必须配置交易邮件，才能发送电子邮件商品。

过程

1. 在 Marketing Cloud (Transact) 中，单击**数据** > **创建数据库**。然后单击**创建**以创建概要文件表。您还可以导入您在其中添加了电子邮件列的概要文件表。
2. 单击**自动化** > **交易消息** > **创建组**。对**事件触发器**选择**交易**。您还需要选择您先前创建的数据源。单击**保存并激活**。

   通过 Marketing Cloud 发送的商品所具有的属性应该与您在 `silverpopContentMapping.properties` 文件中为 `campaignId` 设置的属性相同。此商品属性的值是为自动消息组生成的 `campaignId`。
3. 单击**内容** > **创建邮件**，然后选择上一步骤中的内容源。输入邮件正文。单击**自动化**。选择**将邮件分配到现有自动消息组**。单击**提交并激活**。

可以使用商品属性和概要文件属性使邮件主题行和正文个性化。使用 %%Attribute_Name%% 语法来定义属性。

4. Marketing Cloud 服务器仅接受来自预先设置的 IP 地址的出站网关提交。要添加 IP 地址，请浏览到**设置** > **组织管理** > **安全设置** > **访问限制**。
5. 如果使用 WebSphere Application Server，那么需要导入 Marketing Cloud SSL 证书。WebLogic 用户不需要执行此操作。
   a. 在 WebSphere Application Server 控制台中，浏览到 **SSL 证书和密钥管理** > **密钥存储库和证书** > **NodeDefaultTrustStore** > **签署者证书** > **从端口中检索**。
   b. 设置主机和端口。
   c. 重新启动 WebSphere Application Server。

在与 IBM 技术支持联系之前

如果您遇到无法通过查阅文档解决的问题，那么贵公司的指定支持联系人可致电 IBM 技术支持中心。使用这些准则来确保您的问题得以有效且成功地解决。

如果您不是贵公司的指定支持联系，请与 IBM 管理员联系以了解相关信息。

注：技术支持不会编写或创建 API 脚本。有关实现 API 产品的帮助，请与 IBM 专业服务联系。

要收集的信息

联系 IBM 技术支持前，请收集以下信息：

- 有关问题性质的简短描述。
- 发生问题时看到的详细错误消息。
- 重现该问题的详细步骤。
- 相关的日志文件、会话文件、配置文件和数据文件。
- 关于 产品和系统环境的信息，您可以按"系统信息"中所述来获取。

系统信息

致电 IBM 技术支持时，可能会要求您提供有关系统环境的信息。

如果问题不妨碍登录，那么可在"关于"页面上获得大部分此类信息，该页面提供有关所安装的 IBM 应用程序的信息。

可以通过选择**帮助 > 关于**来访问"关于"页面。如果"关于"页面不可访问，请检查位于应用程序安装目录下面的 `version.txt` 文件。

IBM 技术支持的联系信息

有关联系 IBM 技术支持中心的方法，请参见 IBM 产品技术支持中心网站：(http://www.ibm.com/support/entry/portal/open_service_request)。

注：要输入支持请求，您必须使用 IBM 帐户登录。此帐户必须已链接至 IBM 客户编号。要了解有关将您的帐户与 IBM 客户编号相关联的更多信息，请参阅"支持门户网站"上的**支持资源 > 授权的软件支持**。

声明

本信息是为在美国提供的产品和服务而编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您所在区域当前可获得的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并不意味着授予用户使用这些专利的任何许可。您可以用书面形式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节 (DBCS) 信息的许可证查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区：International Business Machines Corporation"按现状"提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对任何非 IBM Web 站点的引用都只是为了方便起见才提供的， 不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无需对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Corporation
B1WA LKG1
550 King Street
Littleton, MA 01460-1250
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本文档中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估算的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时变更或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

显示的所有 IBM 价格都是 IBM 建议的最新零售价，可随时更改而不另行通知。经销商的价格可能会有所不同。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名字都是虚构的，若现实生活中实际业务企业使用的名字和地址与此相似，纯属巧合。

版权许可：

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无需向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。样本程序都是"按现状"提供的，不附有任何种类的保证。对于因使用样本程序而引起的任何损害，IBM 不承担责任。

如果您正以软拷贝格式查看本信息，那么图片和彩色图例可能无法显示。

商标

IBM、IBM 徽标和 ibm.com 是 International Business Machines Corp. 在全球许多管辖区域中注册的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。Web 上 www.ibm.com/legal/copytrade.shtml 处的"版权和商标"提供了 IBM 商标的当前列表。

隐私策略和使用条款注意事项

IBM 软件产品（包括作为服务解决方案的软件，即"提供的软件"）可能会使用 cookie 或其他技术来收集产品使用情况信息，以便改善最终用户体验、定制与最终用户的交互或用于其他用途。cookie 是 Web 站点可发送至浏览器的数据，随后可将其存储在您的计算机上作为标识您的计算机的标记。在许多情况下，这些 cookie 不会收集个人信息。如果您要使用的软件产品允许您通过 cookie 或类似技术收集个人信息，我们将在下面告知您具体情况。

根据已部署的配置，此软件产品可能使用会话和持久性 cookie，它们收集各个用户的用户名和其他个人信息以用于会话管理、增强用户可用性或其他使用跟踪或功能性目的。可以禁用这些 cookie，但禁用 cookie 同时也会除去它们所启用的功能。

通过 cookie 和类似技术收集的个人信息由不同的管辖区域监管。如果为此软件产品部署的配置为您（作为客户）提供了通过 cookie 和其他技术从最终用户处收集个人信息的能力，那么您应自行寻求适用于此类数据收集的任何法律的相关法律意见，包括在适当时提供通知和同意文件的任何要求。

IBM 要求客户端 (1) 提供明确、显著的指向客户的 Web 站点使用条款（其中包括指向 IBM 和客户端的数据收集和使用实践的链接）的链接（如隐私政策），(2) 告知访问者 IBM 代表客户将 cookie 和透明 GIF/网络信标存放在访问者的计算机上并说明此类技术的目的，并且 (3) 在法律允许的范围内，在客户或 IBM 代表客户将 cookie 和透明 GIF/网络信标存放在 Web 站点访问者的设备上之前征得 Web 站点访问者的同意。

有关如何使用包括 cookie 在内的各种技术实现这些目的的更多信息，请参阅 IBM"网上隐私声明"(http://www.ibm.com/privacy/details/us/en) 中的"Cookie、Web Beacon 和其他技术"部分。

Printed in China