версия 10 выпуск 0
Февраль 2017

IBM Interact: Руководство администратора

Примечание

Перед тем как использовать данный документ и продукт, описанный в нем, прочтите сведения под заголовком “Замечания” на стр. 379.

Данное издание относится к версии 10, выпуску 0, модификации 0 продукта IBM Interact и ко всем его последующим выпускам и модификациям, пока в новых изданиях не будет указано иное.

Содержание

Глава 14. Свойства конфигурации среды разработки Interact 281

Глава 15. Персонализация предложений в реальном времени на стороне клиента 305

Глава 16. Интеграция Interact и Digital Recommendations 337

Глава 17. Интеграция Interact и Digital Data Exchange. 347

Глава 1. Администрирование IBM Interact

При администрировании Interact вы конфигурируете пользователей и роли, задаете источники данных и дополнительные возможности продукта. Кроме того, вы отслеживаете и конфигурируете среды разработки и выполнения. Вы можете использовать интерфейсы прикладного программирования (application programming interface, API) для конкретных продуктов.

Администрирование Interact состоит из ряда задач. Это, в частности, могут быть такие задачи:

- Обслуживание пользователей и ролей
- Обслуживание источников данных
- Конфигурирование дополнительных функций обслуживания предложений Interact
- Мониторинг эффективности среды выполнения и управление средой выполнения

Прежде чем приступить к администрированию Interact, познакомьтесь с основными понятиями работы Interact. Это упростит выполнение задач управления. В приведенных ниже разделах описываются задачи управления, связанные с Interact.

Вторая часть руководства администратора описывает API, доступные для работы с Interact:

- API Interact
- API ExternalCallout
- API обучения

Ключевые понятия Interact

IBM® Interact - это интерактивный механизм, который направляет персонализированные маркетинговые предложения различным аудиториям.

В этом разделе описаны некоторые ключевые понятия, которые вы должны понимать, прежде чем работать с Interact.

Уровни аудитории

Уровень аудитории - это набор идентификаторов, на которые может быть направлена кампания. Можно задать уровни аудитории, чтобы задать правильный набор аудиторий для кампании.

Например, в наборе кампаний могут использоваться уровни аудитории "Семья", "Перспективный клиент", "Клиент" и "Счет". Каждый из этих уровней соответствует определенному представлению маркетинговых данных, доступных для кампании.

Уровни аудитории, как правило, организованы иерархически. В вышеприведенных примерах:

- Семья находится на вершине иерархии, и в каждой семье может содержаться несколько покупателей и один или несколько потенциальных клиентов.
- Следующим в иерархии идет покупатель, и у каждого покупателя может быть несколько счетов.
- Счет находится в самом низу иерархии.

Однако в средах взаимодействия разных предприятий друг с другом существуют более сложные примеры иерархий аудитории, когда могут понадобиться уровни аудитории для корпораций, компаний, подразделений, групп, физических лиц, счетов и так далее.

У этих уровней аудитории могут быть разные взаимосвязи друг с другом, например, один-один, много-один или много-много. Задавая уровни аудитории, вы позволяете представить эти понятия в Campaign, чтобы пользователи могли управлять взаимосвязями между этими разными уровнями аудитории для фокусирования. Например, хотя может быть несколько перспективных клиентов для семьи, вы можете ограничить почтовые сообщения одним перспективным клиентом на одну семью.

Среда разработки

Используйте среду разработки для конфигурирования различных компонентов Interact и внедрения этих компонентов в среду выполнения.

Среда разработки - это среда, в которой вы выполняете значительную часть конфигурирования Interact. В среде разработки вы задаете события, точки взаимодействия, интеллектуальные сегменты и правила процедур. После конфигурирования этих компонентов вы внедряете их в среду выполнения.

Среда разработки устанавливается вместе с веб-программойCampaign.

События

Событие - это действие, предпринятое посетителем, которое инициирует действие в среде выполнения. Примеры событий: помещение посетителя в сегмент, представление предложения или запись данных в журнал.

События сначала создаются в интерактивном канале, а затем инициируются вызовом в Interact API с использованием метода `postEvent`. Событие может вызвать одно из следующих действий, заданных в среде разработки Interact:

- **Инициировать пересегментацию**: Среда выполнения запускает интерактивные потоковые диаграммы для текущего уровня аудитории, связываемого с интерактивным каналом повторно, при помощи текущих данных в сеансе посетителя.

  При разработке взаимодействия, если не указать конкретную потоковую диаграмму, действие пересегментации будет запускать все интерактивные потоковые диаграммы, связанные с этим интерактивным каналом, с текущим уровнем аудитории повторно, и любое требование для предложений будет ожидать, пока все потоковые диаграммы не закончатся. Чрезмерная пересегментация в ходе одного посещения может повлиять на эффективность точки контакта, и это станет видно покупателю.

  Поместите покупателя в новый сегмент после добавления новых данные в объект среды выполнения, например, новых данных из требований API Interact (таких как изменение аудитории) или действий покупателей (например, добавления новых элементов в список пожеланий или в корзину для виртуальных покупок).
- **Записать контакт для предложения**: Среда выполнения помечает флагом рекомендуемые предложения для службы базы данных с целью записи этих предложений в журнал хронологии контактов.

  В случае веб-интеграций записывайте контакт предложения в том же вызове, в котором вы запрашиваете предложения минимизировать число требований между точкой контакта и сервером среды выполнения.

Если точка контакта не возвращает коды процедур для предложений, представленных посетителю компонентом Interact, среда выполнения запишет в журнал последний список рекомендуемых предложений.

- **Записать принятие предложения**: Среда выполнения помечает флагом выбранное предложение для службы базы данных с целью записи в журнал хронологии ответов.
- **Записать отклонение предложения**: Среда выполнения помечает флагом выбранное предложение для службы базы данных с целью записи в журнал хронологии ответов.
- **Пользовательское выражение триггера**: *Действие выражения* - это действие, которое можно определить при помощи макрокоманд Interact, в состав которых входят функции, переменные и операторы, включая EXTERNALCALLOUT. Вы можете задать возвращаемое значение для любого атрибута профиля.

  Когда вы щелкнете по значку изменения рядом с выражением пользователя триггера, появится стандартное диалоговое окно для изменения выражения пользователя, и вы сможете использовать это диалоговое окно, чтобы задать уровень аудитории, необязательное имя поля, для которого следует назначить результаты, и определение самого выражения.
- **Инициировать события**: При помощи действия Инициировать события можно ввести имя события, которое будет инициироваться этим действием. Если вы введете уже заданное событие, это событие будет инициировано при запуске этого действия. Если введенное вами имя события не существует, это действие вызовет создание этого события с заданным действием.

Вы также можете использовать события для инициирования действий, заданных методом `postEvent`, включая запись данных в таблицу, включение данных для обучения или инициирование отдельных потоковых диаграмм.

В среде разработки события можно для вашего удобства организовать в виде категорий. В производственной среде категории не несут никакой функциональной нагрузки.

Интерактивные каналы

Используйте интерактивные каналы в Interact для координации всех объектов, данных и ресурсов серверов, занятых в интерактивном маркетинге.

Интерактивный канал - это представление точки контакта в Campaign, где методом интерфейса является интерактивный диалог. Это программное представление, используемое для координации всех объектов, данных и ресурсов серверов, занятых в интерактивном маркетинге.

Интерактивный канал - это инструмент, который вы используете, чтобы задать точки взаимодействия и события. Вы также можете получить доступ к отчетам для интерактивного канала с вкладки Анализ для этого интерактивного канала.

Интерактивные каналы также содержат назначения производственных серверов среды выполнения и промежуточных серверов. Вы можете создать несколько интерактивных каналов, чтобы организовать события и точки взаимодействия, если у вас есть только один набор производственных серверов среды выполнения и промежуточных серверов, или чтобы разделить события и точки взаимодействия по системам, ориентированным на покупателей.

Интерактивные потоковые диаграммы

Используйте интерактивные потоковые диаграммы, чтобы разделить покупателей на сегменты и назначить профиль сегменту.

Интерактивная потоковая диаграмма связана с пакетной потоковой диаграммой Campaign, но немного отличается от нее. Интерактивная потоковая диаграмма выполняет ту же основную функцию, что и пакетные потоковые диаграммы - делит покупателей на группы, называемые сегментами. Однако в случае интерактивных потоковых диаграмм группы будут интеллектуальными сегментами. Interact использует эти интерактивные потоковые диаграммы для назначения профиля сегменту, когда событие или системное событие указывает, что требуется пересегментация посетителей.

Интерактивные потоковые диаграммы содержат поднабор процессов пакетных потоковых диаграмм, а также несколько процессов, связанных только с интерактивными потоковыми диаграммами.

Примечание: Интерактивные потоковые диаграммы можно создавать только в сеансе Campaign.

Точки взаимодействия

Точка контакта - это место, в котором вы представляете предложение.

Точки взаимодействия содержат содержимое фильтра по умолчанию в тех случаях, когда в среде выполнения нет другого содержимого, которое можно было бы представить. Точки взаимодействия можно огранизовать в зоны.

Предложения

Предложение - это одно маркетинговое сообщение, которое можно доставить различными способами.

В Campaign вы создаете предложения, которые можно использовать в одной или нескольких кампаниях.

Предложения можно использовать многократно:

- В разных кампаниях
- В разные моменты времени
- Для разных групп людей (ячеек)
- В виде разных "версий", изменяя параметризованные поля предложения

Вы назначаете предложения точкам взаимодействия в точках контакта, представленных посетителям.

Профили

Профиль - это набор данных о покупателе, используемый средой выполнения. Это может быть подмножество данных о покупателе, имеющееся в вашей базе данных покупателей, и/или данных, собранных в реальном времени.

Данные покупателя используются для следующих целей:

- Чтобы назначить покупателя в один или более интеллектуальных сегментов в сценариях взаимодействия в реальном времени.

Вы должны задать данные профиля для каждого уровня аудитории, на основе которого вы хотите производить сегментирование. Например, если вы сегментируете данные на основе расположения, вы можете из всей имеющейся у вас информации об адресе взять только почтовый индекс покупателя.

- Чтобы персонализировать предложения
- Как атрибуты для отслеживания при обучении

  Например, можно сконфигурировать Interact для отслеживания семейного положения посетителя и того, сколько посетителей с тем или иным положением приняли данное предложение. После этого среда выполнения может использовать эту информацию для уточнения выбора предложений.

Для среды выполнения эти данные доступны только для чтения.

Среда выполнения

Среда выполнения соединяется с точкой контакта и выполняет взаимодействия. Среда выполнения может состоять из одного или нескольких серверов среды выполнения, соединенных с точкой контакта.

Среда выполнения использует информацию, внедренную из среды разработки, в сочетании с API Interact, чтобы представить предложения для вашей точки контакта.

Сеансы среды выполнения

Сеанс среды выполнения существует на сервере среды выполнения для каждого посетителя вашей точки контакта. В этом сеансе удерживаются все данные для посетителя, которые среда выполнения использует для назначения посетителей для сегментов и выдаче рекомендованных предложений.

Вы создаете сеанс среды выполнения, когда используете вызов `startSession`.

Точки взаимодействия

Точка контакта - это программа или место, где вы можете взаимодействовать с покупателем. Точка контакта может служить каналом, в котором покупатель инициирует контакт ("входное" взаимодействие) или в котором вы осуществляете контакт с покупателем ("исходящее взаимодействие").

Типичные примеры: веб-сайты и программы центров обработки звонков. Используя API Interact, вы можете интегрировать Interact с вашими точками контакта, чтобы представлять покупателям предложения на основе их действий в точке контакта. Точки взаимодействия также называются системами, обращенными к клиенту (client-facing system, CFS).

Правила процедур

Правила процедур назначают предложение для интеллектуального сегмента. Эти назначения дополнительно ограничиваются пользовательской зоной, которую вы связали с предложением в правиле процедуры.

Например, у вас может быть один набор предложений, назначенный для интеллектуального сегмента в зоне "входа", и другой набор предложений для того же сегмента - в зоне "после покупки". Правила процедур должны быть заданы на вкладке стратегии взаимодействия кампании.

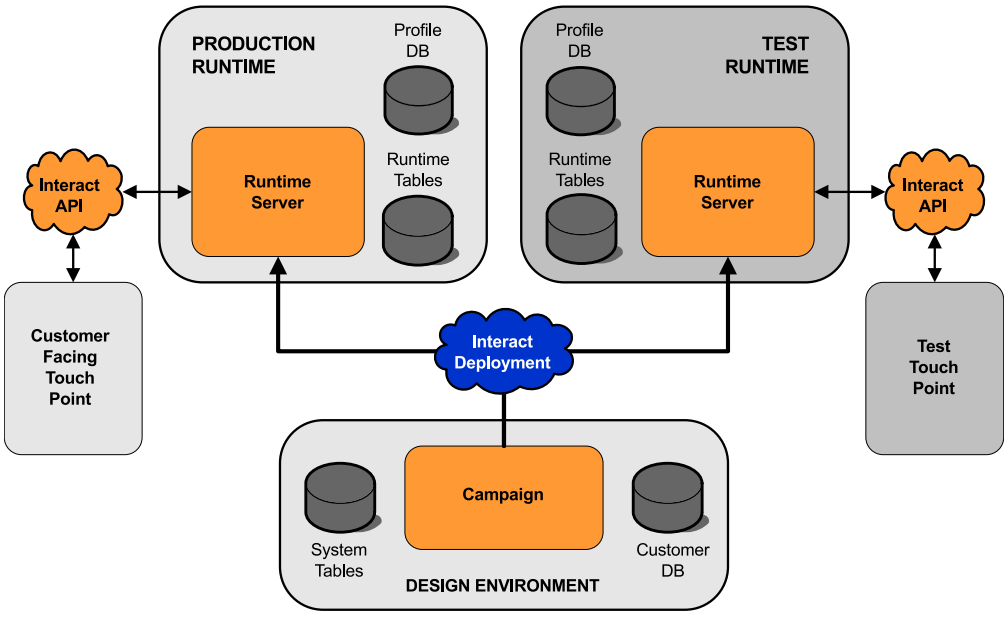
У каждого правила процедуры также есть маркетинговая оценка. Если покупатель назначен более чем в один сегмент, в связи с чем доступно более одного предложения,

маркетинговая оценка поможет определить, какое предложение рекомендует Interact. На то, какие предложения будет рекомендовать среда разработки, можно повлиять за счет модуля обучения, списка подавления предложений, а также глобальных и отдельных назначений предложений.

Архитектура Interact

Среда Interact состоит как минимум из двух основных компонентов - среды разработки и среды выполнения. Также могут быть необязательные серверы тестирования среды выполнения.

Ниже представлен рисунок с обзором высокоуровневой архитектуры.



Среда разработки - это среда, в которой вы выполняете большую часть конфигурации Interact. Среда разработки устанавливается вместе с веб-программой Campaign и ссылается на системные таблицы Campaign и ваши базы данных покупателей. Среда разработки используется для задания точек взаимодействия и событий, используемых с API.

После того, как вы разработали и сконфигурировали характер обработки средой выполнения взаимодействий покупателей, вы внедряете эти данные в группу промежуточных серверов для тестирования или же в группу серверов производственной среды выполнения для взаимодействий покупателей в реальном времени.

API Interact обеспечивает соединение между вашей точкой взаимодействия и сервером среды выполнения. Вы ссылаетесь на объекты (точки взаимодействия и события), созданные в среде разработки при помощи API Interact и используете их для затребования информации с сервера среды выполнения.

Замечания относительно сети Interact

Производственная установка Interact занимает как минимум два компьютера. В высокопроизводительной производственной среде используется несколько серверов времени выполнения Interact и распределенных баз данных, и установка может содержать не один десяток компьютеров.

Для повышения производительности учитывайте ряд требований к топологии сети.

- Если реализация API Interact запускает и завершает сеансы в одном и том же вызове, например:

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

  вам не нужно разрешать закрепленные сеансы связи между балансировщиком нагрузки и серверами времени выполнения Interact. Можно сконфигурировать управление сеансом связи с серверами времени выполнения Interact для локального типа кэшей.
- Если реализация API Interact использует несколько вызовов для запуска и завершения сеансов, например:

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

  и для серверов времени выполнения Interact используется балансировщик нагрузки, вам нужно разрешить для него некоторый тип хранения сеансов (так называемые закрепленные сеансы). Если это невозможно, или вы не используете балансировщик нагрузки, сконфигурируйте управление сеансами серверов Interact с кэшами распределенного типа (`cacheType`). Если вы используете распределенный кэш, все серверы времени выполнения Interact должны иметь возможность обмениваться информацией в режиме широковещания. Возможно, потребуется настроить сеть, чтобы передача информации между серверами Interact, использующими один и тот же IP-адрес широковещания и порт, не снижала производительность системы. У балансировщика нагрузки с закрепленными сеансами производительность выше, чем при использовании распределенного кэша.
- Распределенное кэширование между несколькими группами серверов не поддерживается.
- Для повышения производительности используйте одно географическое положение для всей среды выполнения, включая серверы Interact, Marketing Platform, балансировщики нагрузки и точку контакта. Можно использовать другое географическое положение для среды разработки, хотя это замедлит внедрение.
- Используйте быстрое сетевое соединение (минимум 1 Гб) между группой производственных серверов Interact и соответствующей точкой контакта.
- У среды разработки должен быть доступ к среде выполнения по соединению http или https, чтобы выполнять задачи внедрения. Любые брандмауэры или другие сетевые программы нужно сконфигурировать так, чтобы они разрешали внедрение. Возможно, для больших внедрений потребуется продлить срок ожидания соединения HTTP между средами разработки и выполнения.
- Для модуля хронологии контактов и ответов требуется доступ к базе данных среды разработки (системные таблицы Campaign) и доступ к базе данных среды выполнения (таблицы среды выполнения Interact). Для такой передачи данных нужно сконфигурировать базы данных и сеть.

В тестовой или промежуточной установке можно установить на одном компьютере среду разработки Interact и среду выполнения. Для производственной среды такой сценарий не рекомендуется.

Порты сервера Interact и защита сети

Сконфигурируйте Interact для защиты портов сервера.

Порты среды выполнения Interact

В зависимости от конфигурации некоторые из этих портов могут быть закрыты или нужны не во всех установках Interact.

Порт сервера прикладных программ Interact для HTTP
Порт по умолчанию, где обрабатываются запросы Interact.

Порт сервера прикладных программ Interact для HTTPS
Порт SSL по умолчанию, где обрабатываются запросы Interact.

Порт Interact systemTablesDataSource
Смотрите конфигурацию источника данных JDBC в Marketing Platform.

Порт Interact learningTablesDataSource
Смотрите конфигурацию источника данных JDBC в Marketing Platform.

Порт Interact contactAndResponseHistoryDataSource
Смотрите конфигурацию источника данных JDBC в Marketing Platform.

Порт Interact prodUserDataSource
Смотрите конфигурацию источника данных JDBC в Marketing Platform.

Порт Interact testRunDataSource
Смотрите конфигурацию источника данных JDBC в Marketing Platform.

Коммуникационный порт ETL
Сконфигурируйте этот порт в свойствах конфигурации **Interact | ETL | patternStateETL | communicationPort**.

Порт широковещания EHCache
Сконфигурируйте этот порт в свойствах конфигурации **Interact | cacheManagement | Cache | Managers | EHCache | Parameter Data | multicastPort**, если режим кэша - распределенный.

Порт каталога ExtremeScale
Сконфигурируйте этот порт в свойствах конфигурации **Interact | Cache Managers | Extreme Scale | Parameter Data | catalogURLs** .

Порт мониторинга JMX Interact
Сконфигурируйте этот порт в свойствах конфигурации **Interact | monitoring | port** или выполните `-Dinteract.jmx.monitoring.port=номер порта`.

Порт WebConnector Interact

Обычно это тот же порт, что порт сервера Interact, но его можно изменить в файле `jsconnector.xml`.

О портах для любых интегрированных продуктах Interact смотрите в документации по эти продуктам.

Мониторинг JMX необязателен для обычной работы Interact. Но он используется для диагностики и отслеживания.

Доступ к порту JMX можно отключить в конфигурации Interact или ограничить конкретным IP-адресом в конфигурациях брандмауэра. Это рекомендуется в связи с уязвимостью JMX, которая была недавно обнаружена в библиотеке Apache Commons Library стороннего поставщика.

Функция JMX remoting в Apache Geronimo 3.x до 3.0.1, используемая сервером IBM WebSphere Application Server (WAS) Community Edition 3.0.0.3 и другими продуктами, неправильно реализует загрузчик классов RMI, что позволяет удаленным атакующим выполнять условный код, используя соединитель JMX для отправки созданного вручную сериализованного объекта. Смотрите http://www-01.ibm.com/support/docview.wss?uid=swg21643282.

Порты разработки Interact

В зависимости от конфигурации некоторые из этих портов могут быть закрыты или нужны не во всех установках Interact.

Порт сервера прикладных программ Campaign для HTTP
Порт по умолчанию, где обрабатываются запросы Interact.

Порт сервера прикладных программ Campaign для HTTPS
Порт SSL по умолчанию, где обрабатываются запросы Interact.

Порт приемника Campaign
Порт, используемый Campaign внутренним образом для принятия соединений от веб-клиента.

Другие порты разработки Campaign
Дополнительную информацию об этих портах смотрите в документации Campaign.

Порт соединителя JMX Campaign
Сконфигурируйте этот порт в свойствах конфигурации **Campaign | monitoring | port** только для мониторинга хронологии контактов и ответов.

Порт сервера мониторинга операций Campaign
Сконфигурируйте этот порт в свойствах конфигурации **Campaign | monitoring | serverURL**.

Вход в систему IBM Marketing Software

Используйте для входа в IBM Marketing Software следующую процедуру.

Прежде чем начать

Вам понадобится следующее:

- Соединение с внутренней сетью для доступа к серверу IBM Marketing Software.
- Поддерживаемый браузер, установленный на вашем компьютере.
- Имя пользователя и пароль для входа в систему IBM Marketing Software.
- URL для доступа к IBM Marketing Software в вашей сети.

Это следующий URL:

`http://`*`хост.домен.com:порт`*`/unica`

где

`хост` - это компьютер, на котором установлен Marketing Platform.

домен.com - это домен, в котором находится хост-компьютер.

порт - это номер порта ожидания сервера прикладных программ Marketing Platform.

Примечание: В следующей процедуре предполагается, что вы входите в систему с учетной записью с правами доступа Администратор для Marketing Platform.

Процедура

Откройте в браузере URL IBM Marketing Software.

- Если IBM Marketing Software интегрирован с Windows Active Directory или с платформой управления доступом в веб-систему, и вы вошли в эту систему, то откроется страница сводной панели по умолчанию. Ваш вход в систему будет завершен.
- Если вы видите экран входа в систему, то войдите в систему с идентификационными данными по умолчанию. В среде с одним разделом используйте `asm_admin` и пароль `password`. В среде с несколькими разделами используйте `platform_admin` и пароль `password`.

  Вас попросят изменить пароль. Можно ввести существующий пароль, но для улучшения защиты лучше выбрать новый.
- Если IBM Marketing Software сконфигурирован для использования SSL, то при первом входе в систему вас могут попросить принять цифровой сертификат безопасности. Нажмите **Да**, чтобы принять сертификат.

Если вход в систему выполнен успешно, то IBM Marketing Software открывает страницу сводной панели по умолчанию.

Результаты

С разрешениями по умолчанию, назначенными учетным записям администратора Marketing Platform, вы можете выполнять администрирование учетных записей и защиты, используя опции в меню **Параметры**. Чтобы выполнить задачи администрирования высшего уровня для IBM Marketing Software, нужно войти в систему как **platform_admin**.

Глава 2. Конфигурирование пользователей

Для Interact требуется сконфигурировать два набора пользователей - пользователей среды выполнения и пользователей среды разработки.

- **Пользователи среды разработки** создаются в среде Marketing Platform, сконфигурированной для работы с серверами среды выполнения.
- **Пользователи среды разработки** - это пользователи Campaign. Сконфигурируйте защиту для различных участников вашей команды разработки, как для Campaign.

Конфигурирование пользователей среды выполнения

Установив Interact, нужно сконфигурировать по меньшей мере одного пользователя Interact, пользователя среды выполнения. Пользователи среды выполнения создаются в Marketing Platform.

Об этой задаче

Пользователь среды выполнения служит для доступа к таблицам среды выполнения. Пользователь среды выполнения - это имя пользователя и пароль, используемые для внедрения интерактивных каналов. Серверы среды выполнения проверяют учетные записи базы данных при помощи аутентификации JDBC на сервере веб-программ. Не требуется добавлять никаких источников данных среды выполнения для пользователя среды выполнения.

Пользователь LDAP и любой пользователь платформы могут внедрить интерактивный канал. Роль InteractAdminRole для внедрения интерактивного канала не обязательна.

Когда вы создаете пользователей среды выполнения:

- Если у вас отдельные экземпляры Marketing Platform для каждого сервера среды выполнения, на каждом нужно создать одного и того же пользователя с одним и тем же паролем. Все серверы среды выполнения, принадлежащие к одной группе серверов, должны совместно использовать учетные данные пользователя.
- Если используется утилита загрузки базы данных, нужно определить таблицы среды выполнения как источники данных с параметрами регистрации для среды выполнения в свойствах конфигурации в разделе `Interact > general > systemTablesDataSource`.
- Если вы включили защиту для мониторинга JMX с протоколом JMXMP, может потребоваться отдельный пользователь для защиты мониторинга JMX.

О действиях для создания пользователей среды выполнения смотрите в документации по Marketing Platform.

Конфигурирование пользователей среды разработки

Пользователи среды разработки - это пользователи Campaign. Вы конфигурируете пользователей вашей среды разработки так же, как конфигурируете разрешения ролей Campaign.

Об этой задаче

Некоторым пользователям среды разработки требуются также определенные разрешения Campaign, такие как Пользовательские макрокоманды.

Когда вы создаете пользователей среды разработки:

- Если у вас есть какие-либо пользователи Campaign с разрешениями редактировать интерактивные потоковые диаграммы, предоставляют им доступ к источнику данных таблиц тест-запуска.
- Если вы установили и сконфигурировали Interact, следующие дополнительные опции доступны для глобальной политики по умолчанию и новых политик.
-

Категория	Разрешения
Кампании	• Просмотр стратегий взаимодействия кампаний - возможность просматривать, но не редактировать вкладки стратегии взаимодействия в кампании. • Редактирование стратегий взаимодействия кампаний - возможность вносить изменения во вкладки стратегии взаимодействия, в том числе в правила процедур. • Удаление стратегий взаимодействия кампании - возможность удалять вкладки стратегии взаимодействия из кампаний. Если стратегия взаимодействия была включена во внедрение интерактивного канала, возможность удаления вкладки стратегии взаимодействия ограничена. • Добавление стратегии взаимодействия кампании - возможность создавать новые вкладки стратегии взаимодействия в кампании. • Инициирование внедрения стратегии взаимодействия кампании - возможность помечать вкладку стратегии взаимодействия для внедрения или отмены внедрения.
Интерактивные каналы	• Внедрение интерактивных каналов - возможность внедрения интерактивного канала в среды выполнения Interact. • Редактирование интерактивных каналов - возможность вносить изменения на вкладку сводки интерактивных каналов. • Удаление интерактивных каналов - возможность удалять интерактивные каналы. Если интерактивный канал был внедрен, возможность удаления этого интерактивного канала ограничена. • Просмотр интерактивных каналы - возможность просматривать, но не редактировать интерактивные каналы. • Добавление интерактивных каналов - возможность добавлять интерактивные каналы. • Просмотр отчетов интерактивных каналов - возможность просматривать вкладку анализа интерактивного канала. • Добавление дочерних объектов интерактивного канала - возможность добавлять точки взаимодействия, зоны, события и категории.

Категория	Разрешения
Сеансы	• Просмотр интерактивных потоковых диаграмм - возможность просматривать интерактивную потоковую диаграмму в сеансе. • Добавление интерактивных потоковых диаграмм - возможность создавать новые интерактивные потоковые диаграммы в сеансе. • Редактирование интерактивных потоковых диаграмм - возможность вносить изменения в интерактивные потоковые диаграммы. • Удаление интерактивных потоковых диаграмм - возможность удалять интерактивные потоковые диаграммы. Если интерактивный поток, которому назначена интерактивная потоковая диаграмма, был внедрен, возможность удаления интерактивных потоковых диаграмм ограничена. • Копирование интерактивных потоковых диаграмм - возможность копировать интерактивные потоковые диаграммы. • Тест-запуск интерактивных потоковых диаграмм - возможность инициировать тест-запуск интерактивной потоковой диаграммы. • Пересмотр интерактивных потоковых диаграмм - возможность просматривать интерактивную потоковую диаграмму, но не вносить в нее изменения. • Внедрение интерактивных потоковых диаграмм - возможность помечать интерактивные потоковые диаграммы для внедрения или отмены внедрения.

Пример разрешений для среды разработки

В этом примере перечисляются разрешения, предоставляемые двум различным ролям, одной для пользователей, которые создают интерактивные потоковые диаграммы, другой для пользователей, которые определяют стратегии взаимодействия.

Роль интерактивной потоковой диаграммы

Эта таблица содержит разрешения, которые даны роли интерактивной потоковой диаграммы:

Категория	Разрешение
Пользовательская макрокоманда	У роли пользователя есть следующие разрешения: • Добавить пользовательские макрокоманды • Изменить пользовательские макрокоманды • Использовать пользовательские макрокоманды
Производное поле	У роли пользователя есть следующие разрешения: • Добавить производные поля • Изменить производные поля • Использовать производные поля
Шаблон потоковой диаграммы	У роли пользователя есть следующие разрешения: • Вставить шаблоны

Категория	Разрешение
Шаблон сегмента	У роли пользователя есть следующие разрешения: • Добавить сегменты • Изменить сегменты
Сеанс	У роли пользователя есть следующие разрешения: • Показать сводную информацию о сеансах • Просмотреть интерактивные потоковые диаграммы • Добавить интерактивные потоковые диаграммы • Изменить интерактивные потоковые диаграммы • Копировать интерактивные потоковые диаграммы • Тест-запуск интерактивных потоковых диаграмм • Внедрить интерактивные потоковые диаграммы

Стратегия взаимодействия: Роль

Эта таблица содержит разрешения, которые даны роли стратегии взаимодействия:

Категория	Разрешение
Кампания	У роли пользователя есть следующие разрешения: • Показать сводную информацию о кампаниях • Управление ячейками назначения кампании • Просмотреть стратегии взаимодействия кампании • Изменить стратегии взаимодействия кампании • Добавить стратегии взаимодействия кампании • Инициировать внедрения стратегии взаимодействия кампании
Предложение	У роли пользователя есть следующие разрешения: • Показать сводную информацию о предложениях
Шаблон сегмента	У роли пользователя есть следующие разрешения: • Просмотр сводной информации о сегментах
Сеанс	У роли пользователя есть следующие разрешения: • Проверить интерактивные потоковые диаграммы

Глава 3. Управление источниками данных Interact

Для правильного функционирования Interact требуется несколько источников данных. Некоторые источники данных содержат информацию, нужную для работы Interact, а в других источниках данных содержатся ваши данные.

В следующих разделах описываются источники данных Interact, в том числе информация, необходимая для их правильного конфигурирования, и некоторые предложения по поддержке данных.

Источники данных Interact

Для работы Interact требуется несколько наборов данных. Эти наборы хранятся в источниках данных и получаются из них, а сами сконфигурированные вами источники данных зависят от включенных возможностей Interact.

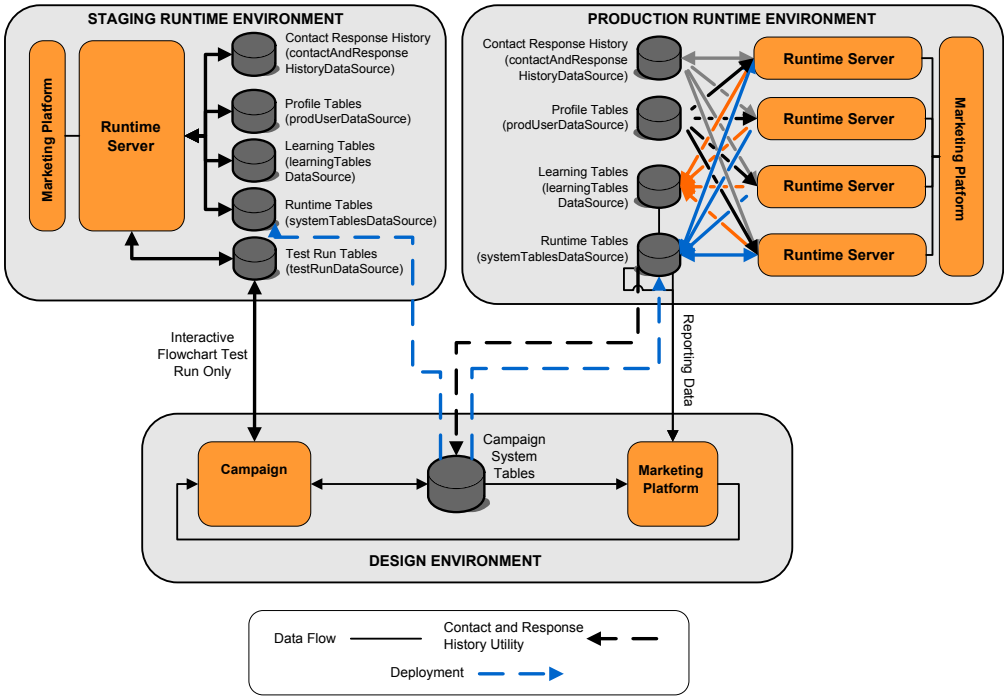
- **Системные таблицы Campaign**. Кроме всех данных для Campaign, системные таблицы Campaign содержат данные для компонентов Interact, создаваемых в среде разработки, такие как правила процедур и интерактивные каналы. Среда разработки и системные таблицы Campaign используют одну и ту же физическую базу данных и схему.
- **Таблицы среды выполнения**(`systemTablesDataSource`). Этот источник данных содержит данные внедрения из среды разработки, промежуточные таблицы для хронологии контактов и ответов и статистику среды выполнения.
- **Таблицы профилей** (`prodUserDataSource`). Этот источник данных содержит все пользовательские данные в дополнение к информации, собираемой в реальном времени, что требуется интерактивным потоковым диаграммам для правильного расположения посетителей в интеллектуальные сегменты. Если вы основываетесь исключительно на данных реального времени, таблицы профилей не требуются. Если таблицы профилей используются, у вас должна быть по крайней мере одна таблица профилей на уровень аудитории, используемый интерактивным каналом.

  Таблицы профилей могут содержать также таблицы для расширения представления предложений, в том числе таблицы для подавления предложений, переопределения оценки и назначения глобальных и индивидуальных предложений.
- **Таблицы тест-запусков** (`testRunDataSource`). Этот источник данных содержит образец всех данных, необходимых для интерактивных потоковых диаграмм, чтобы расположить посетителей по интеллектуальным сегментам, в том числе данные, отображающие, что собрано в реальном времени при взаимодействии. Эти таблицы требуются для группы серверов, выделенной как группа серверов тест-запусков только для среды разработки.
- **Таблицы обучения** (`learningTablesDataSource`). Этот источник содержит все данные, собранные встроенной утилитой обучения. Среди этих таблиц может быть и таблица, определяющая динамические атрибуты. Если вы не используете утилиту обучения или используете внешнюю создаваемую утилиту обучения, эти таблицы обучения не требуются.
- **Хронология контактов и ответов для межсеансовых ответов** (`contactAndResponseHistoryDataSource`). Этот источник данных содержит или таблицы хронологии контактов Campaign, или их копию. Если вы не используете возможность межсеансовых ответов, эти таблицы хронологии контактов конфигурировать не нужно.

Базы данных и прикладные программы

Создаваемые вами для использования Interact источники данных могут использоваться также для обмена данными или их совместного использования с другими прикладными программами IBM Marketing Software.

На следующей диаграмме показаны источники данных Interact и их связь с прикладными программами IBM Marketing Software.



- И Campaign, и группа серверов тест-запусков обращаются к таблицам тест-запусков.
- Таблицы тест-запусков используются только для тест-запусков интерактивных потоковых диаграмм.
- При использовании сервера среды выполнения для проверки внедрения, в том числе API Interact, сервер среды выполнения использует таблицы профиля для данных.
- При конфигурировании модуля хронологии контактов и ответов этот модуль использует фоновый процесс Extract, Transform, Load (ETL), чтобы переместить данные из промежуточных таблиц среды выполнения в таблицы хронологии контактов и ответов Campaign.
- Данные запросов с функцией отчетов из обучающих таблиц, таблиц среды выполнения и системных таблиц Campaign для вывода отчетов в Campaign.

Необходимо сконфигурировать среды выполнения для тестирования, чтобы использовать набор таблиц, отличающийся от используемого в производственных средах выполнения. При разделении таблиц по производственным и промежуточным можно сохранять результаты тестирования отдельно от действительных производственных результатов. Обратите внимание на то, что модуль хронологии контактов и ответов всегда вставляет данные в фактические таблицы хронологии контактов и ответов Campaign (у Campaign нет таблиц хронологии контактов и ответов для тестирования). Если у вас есть отдельные таблицы обучения для среды

выполнения тестирования и нужно увидеть результаты в отчетах, необходим отдельный экземпляр IBM Cognos BI, запускающий отчеты обучения для среды тестирования.

CampaignСистемные таблицы

При установке среды разработки Interact можно создать также новые, относящиеся к Interact таблицы среди системных таблиц Campaign. Создаваемые таблицы зависят от включаемых возможностей Interact.

При включении модуля хронологии контактов и ответов этот модуль копирует хронологию контактов и ответов из промежуточных таблиц среди таблиц среды выполнения в таблицы хронологии контактов и ответов из числа системных таблиц Campaign. Таблицы по умолчанию - это `UA_ContactHistory`, `UA_DtlContactHist` и `UA_ResponseHistory`, но модуль хронологии контактов и ответов использует любые таблицы, отображенные в Campaign на таблицы хронологии контактов и ответов.

Если для назначения предложений использовать таблицы глобальных предложений и таблицы переопределения оценки, может потребоваться заполнить таблицу `UACI_ICBatchOffers` из числа системных таблиц Campaign, если вы используете предложения, отсутствующие в правилах процедур для интерактивного канала.

Таблицы среды выполнения

Если у вас несколько уровней аудитории, необходимо создать промежуточные таблицы для данных хронологии контактов и ответов для каждого уровня аудитории.

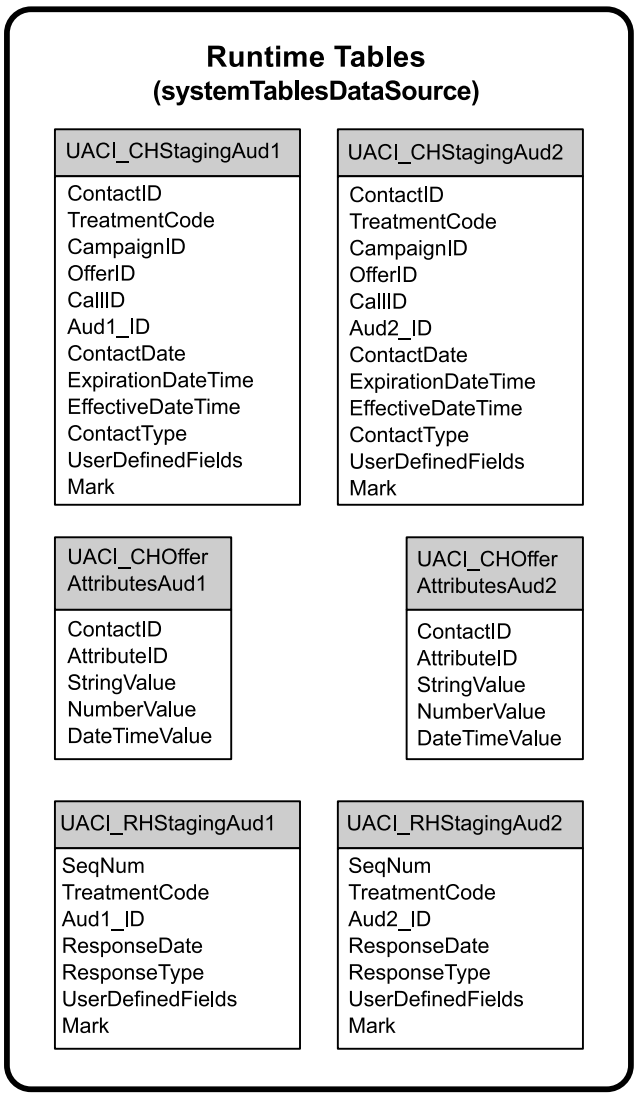
Сценарии SQL создают следующие таблицы для уровня аудитории по умолчанию:

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

Необходимо создать копии этих трех таблиц для каждого из ваших уровней аудитории в таблицах среды выполнения.

Если в ваших таблицах хронологии контактов и ответов Campaign есть пользовательские поля, необходимо создать такие же имена и типы полей в таблицах `UACI_CHStaging` и `UACI_RHStaging`. Эти поля можно заполнить в среде выполнения, создав пары имя-значение для того же имени в данных сеанса. Допустим, например, что в ваших таблицах хронологии контактов и ответов есть поле `catalogID`. Поле `catalogID` необходимо добавить в таблицы `UACI_CHStaging` и `UACI_RHStaging`. После этого API Interact заполняет это поле, определяя параметр события как пару имя-значение с именем `catalogID`. Данные сеанса могут предоставляться таблицей профилей, временными данными, возможностью обучения или API Interact.

На следующей диаграмме показаны образцы таблиц для аудиторий Aud1 и Aud2. Эта диаграмма не содержит всех таблиц базы данных среды выполнения.



Все поля в этих таблицах обязательны. Значения `CustomerID` и `UserDefinedFields` можно изменить, чтобы они соответствовали вашим таблицам хронологии контактов и ответов Campaign.

Таблицы тест-запусков

Таблицы тест-запусков используются только для тест-запусков интерактивных потоковых диаграмм. Тест-запуски интерактивных потоковых диаграмм должны проверить вашу логику сегментации. Требуется только сконфигурировать одну базу данных для тест-запусков в вашей установке Interact. Нет необходимости помещать таблицы тест-запусков в автономную базу данных. Например, можно использовать таблицы данных ваших покупателей для Campaign.

У пользователя базы данных, связанного с таблицами тест-запусков, должны быть привилегии CREATE, чтобы добавлять таблицы результатов тест-запусков.

База данных тест-запусков должна содержать все таблицы, отображенные в интерактивном канале.

Эти таблицы должны содержать данные для запуска сценариев, которые вы хотите проверить на ваших интерактивных потоковых диаграммах. Например, если логика интерактивных диаграмм заключается в сортировке людей по сегментам на основании выбора, сделанного в системе голосовой почты, для каждого возможного выбора должна существовать по крайней мере одна строка. Если вы создаете взаимодействие, работающее через форму на веб-сайте, необходимо включить строки, представляющие пропущенные или плохо сформулированные данные, например, использование `name@domaincom` для значения адреса электронной почты.

В каждой таблице тест-запусков должен быть по крайней мере список ID для соответствующего уровня аудитории и столбец, представляющий данные реального времени, которые вы собираетесь использовать. Так как у тест-запусков нет доступа к данным реального времени, необходимо предоставить образцы данных для каждого набора предполагаемых данных реального времени. Например, если вы хотите использовать данные, которые можно собрать в реальном времени, такие как имя последней посещенной веб-страницы, хранимое в атрибуте `lastPageVisited`, или количество товаров в покупательской корзине, хранимое в атрибуте `shoppingCartItemCount`, необходимо создать столбцы с такими же именами и заполнить их образцами данных. Это позволит произвести тест-запуски для ветвей логики вашей потоковой диаграммы, которые по своей природе поведенческие или зависят от контекста.

Тест-запуски интерактивных потоковых диаграмм не оптимизированы для работы с большими наборами данных. Вы можете ограничить число строк, используемых для тест-запуска, в процессе Взаимодействие. Однако это всегда приводит к выбору первого набора строк. Чтобы обеспечить выбор различных наборов строк, используйте разные представления таблиц тест-запусков.

Чтобы проверить производительность пропускной способности интерактивных потоковых диаграмм в среде выполнения, необходимо создать тестовую среду выполнения, включив таблицу профиля для проверочной среды.

На практике для тестирования вам может понадобиться три набора таблиц: таблица тест-запусков интерактивных потоковых диаграмм, тестовые таблицы профиля для проверки группы серверов и набор производственных таблиц профилей.

Переопределение типов данных по умолчанию, используемых для динамически создаваемых таблиц

Среда выполнения Interact динамически создает таблицы по двум сценариям: при тест-запуске потоковой диаграммы и при запуске процесса Snapshot, при котором производится запись в еще не существующую таблицу. Чтобы создать эти таблицы, Interact основывается на жестко запрограммированных типах данных для каждого поддерживаемого типа базы данных.

Типы данных по умолчанию можно переопределить, создав таблицу альтернативных типов данных с названием `uaci_column_types` в `testRunDataSource` или `prodUserDataSource.` Эта дополнительная таблица позволяет Interact обрабатывать те редкие случаи, которые не подходят под жестко запрограммированные типы данных.

Когда таблица `uaci_column_types` определена, Interact использует метаданные для столбцов как типы данных, которые будут использоваться для создания любой таблицы. Если таблица `uaci_column_types` не определена или при попытке чтения такой таблицы возникла непредвиденная ситуация, используются типы данных по умолчанию.

При запуске система среды выполнения сначала проверяет наличие в testRunDataSource таблицы uaci_column_types. Если таблицы uaci_column_types нет в testRunDataSource или prodUserDataSource относится к другому типу баз данных, Interact проверяет наличие этой таблицы в prodUserDataSource.

Переопределение типов данных по умолчанию

Используйте эту процедуру, чтобы переопределить типы данных по умолчанию для динамически создаваемых таблиц.

Об этой задаче

При всяком изменении таблицы uaci_column_types необходимо перезапустить сервер среды выполнения. Так запланируйте свои изменения, чтобы перезапуск сервера оказывал минимальное воздействие на операции.

Процедура

1. Создайте таблицу в TestRunDataSource или в ProdUserDataSource со следующими свойствами:

   Имя таблицы: uaci_column_types

   Имена столбцов:

   - uaci_float
   - uaci_number
   - uaci_datetime
   - uaci_string

   Для определения каждого столбца используйте соответствующий тип данных, поддерживаемый вашей базой данных.
2. Перезапустите сервер среды выполнения, чтобы Interact мог распознать новую таблицу.

Типы данных по умолчанию для динамически создаваемых таблиц

Для каждой поддерживаемой базы данных, используемой системой среды выполнения Interact, существуют жестко запрограммированные типы данных по умолчанию для столбцов с плавающей запятой, чисел, даты/времени и строк.

Таблица 1. Типы данных по умолчанию для динамически создаваемых таблиц

База данных	Типы данных по умолчанию
DB2	• float • bigint • timestamp • varchar
Informix	• float • int8 • DATETIME YEAR TO FRACTION • char2
Oracle	• float • number(19) • timestamp • varchar2

Таблица 1. Типы данных по умолчанию для динамически создаваемых таблиц (продолжение)

База данных	Типы данных по умолчанию
сервер SQL	• float • bigint • datetime • nvarchar

База данных профилей

Содержимое базы данных профилей полностью зависит от данных, которые нужны вам для конфигурирования интерактивных потоковых диаграмм и API Interact. Для Interact требуется или рекомендуется, чтобы каждая база данных содержала определенные таблицы или данные.

База данных профилей должна содержать:

- Все таблицы, отображенные в интерактивном канале.

  Эти таблицы должны содержать все данные для запуска ваших интерактивных потоковых диаграмм в производстве. Эти таблицы должны быть упрощены, оптимизированы и правильно индексированы. Так как для обращения к многомерным данным требуется высокая производительность, необходимо при всякой возможности использовать денормализованную схему. Как минимум, необходимо индексировать таблицу профилей для полей ID уровня аудитории. Если есть другие полученные из многомерных таблиц поля, они должны быть нужным образом индексированы для сокращения времени выборки. ID аудитории для таблиц профилей должны совпадать с ID аудитории, определенными в Campaign.
- Если для свойства конфигурации `enableScoreOverrideLookup` задать значение true, по крайней мере для одного уровня аудитории необходимо включить таблицу переопределения оценки. Имена таблиц переопределения оценки определяются свойством `scoreOverrideTable`.

  Таблица переопределения оценки может содержать отдельные пары покупатель-предложение. Образец таблицы переопределения оценки `UACI_ScoreOverride` можно создать, запустив сценарий SQL `aci_usrtab` для вашей базы данных профилей. Вы должны также проиндексировать эту таблицу по столбцу ID аудитории.

  Если для свойства `enableScoreOverrideLookup` задать значение false, включать таблицу переопределения оценки не нужно.
- Если для свойства конфигурации `enableDefaultOfferLookup` задать значение true, необходимо включить таблицу глобальных предложений (`UACI_DefaultOffers`). Таблицу глобальных предложений можно создать, запустив сценарий SQL `aci_usrtab` для вашей базы данных профилей.

  Таблица глобальных предложений может содержать пары аудитория-предложение.
- Если для свойства `enableOfferSuppressionLookup` задать значение true, по крайней мере для одного уровня аудитории необходимо включить таблицу подавления предложений. Имена таблиц подавления предложений определяются свойством `offerSuppressionTable`.

  Таблица подавления предложений может содержать строку для каждого подавленного предложения для участника аудитории, хотя запись для всех

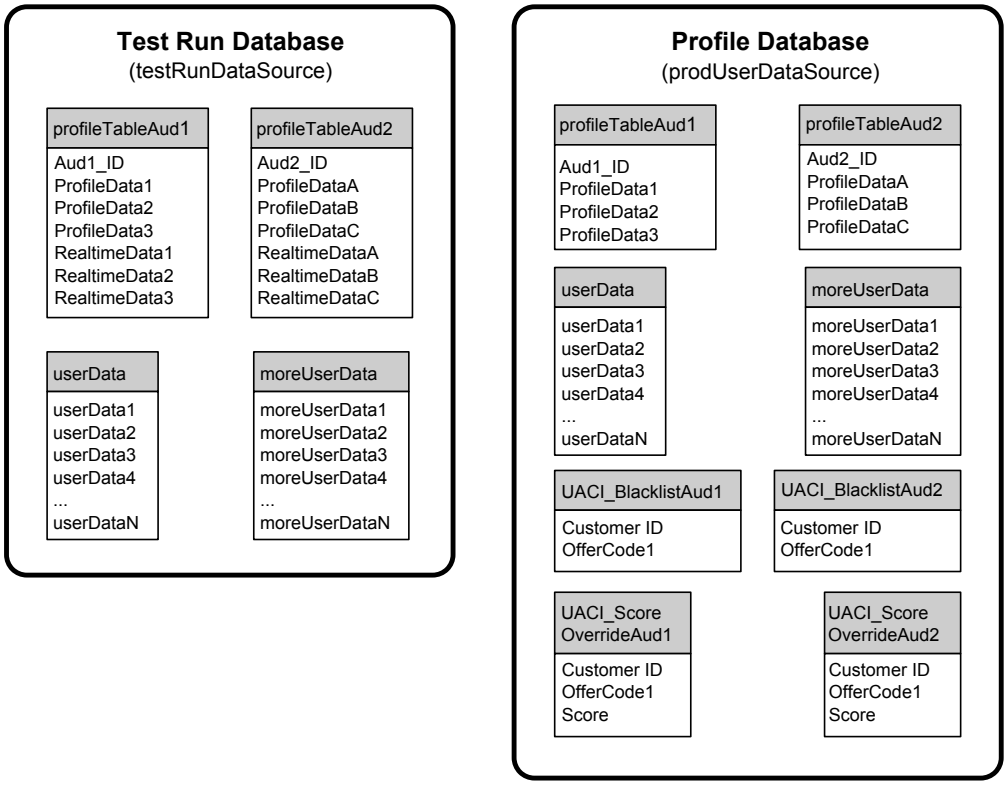
участников аудитории не требуется. Образец таблицы подавления предложения `UACI_BlackList` можно создать, запустив сценарий SQL `aci_usrtab` для вашей базы данных профилей.

Если для свойства `enableOfferSuppressionLookup` задать значение false, включать таблицу подавления предложений не нужно.

Большой объем данных в любой из этих таблиц может существенно повлиять на производительность. Для получения наилучших результатов используйте соответствующие индексы столбцов уровня аудитории для таблиц с большим объемом данных, используемых в среде выполнения.

Все указанные выше свойства конфигурации принадлежат категории **Interact > profile** или **Interact > profile > Audience Levels > *уровень_аудитории***. Сценарий SQL `aci_usrtab` расположен в каталоге `ddl` вашего каталога установки среды выполнения.

На следующей диаграмме показаны примеры таблиц для тест-запуска и базы данных профилей для уровней аудитории Aud1 и Aud2.



Таблицы обучения

При использовании встроенной возможности обучения Interact необходимо сконфигурировать таблицы обучения. Эти таблицы содержат все данные встроенной возможности для обучения.

Если используются атрибуты динамического обучения, необходимо заполнить таблицу `UACI_AttributeList`.

Обучение включает в себя запись в промежуточные таблицы и агрегирование информации из промежуточных таблиц в таблицы обучения. Свойства конфигурации `insertRawStatsIntervalInMinutes` и `aggregateStatsIntervalInMinutes` в категории `Interact > offerserving > Конфигурация встроенного обучения` определяют, как часто заполняются таблицы обучения.

Атрибут `insertRawStatsIntervalInMinutes` определяет, как часто информация о доступе и контакте для каждого покупателя и предложения переносится из памяти в промежуточные таблицы `UACI_OfferStatsTX` и `UACI_OfferTxAll`. Хранимая в промежуточных таблицах информация агрегируется и переносится в таблицы `UACI_OfferStats` и `UACI_OfferStatsAll` через регулярные интервалы времени, определяемые свойством конфигурации `aggregateStatsIntervalInMinutes`.

Встроенная возможность обучения Interact использует эти данные для вычисления окончательных оценок для предложений.

Хронология контактов и отслеживание межсеансовых ответов

Если включить возможность межсеансовых ответов, для среды выполнения требуется доступ с правом чтения к таблицам хронологии контактов Campaign. Можно сконфигурировать среду выполнения для просмотра системных таблиц Campaign или же создать копию таблиц хронологии контактов Campaign. При создании копии таблиц нужно управлять процессом поддержания этой копии на актуальном уровне. Модуль хронологии контактов и ответов не изменяет копию таблиц хронологии контактов.

Для этих таблиц хронологии контактов нужно запустить сценарий SQL `aci_crhtab`, чтобы добавить таблицы, необходимые для возможности отслеживания межсеансовых ответов.

Запускаемые сценарии базы данных для включения возможностей

Чтобы использовать дополнительные возможности, доступные в , запустите сценарии базы данных для этой базы данных для создания таблиц или изменения существующих таблиц.

Ваша установка , включая среду разработки и среду выполнения, содержит сценарии **`ddl`** для возможностей. Сценарии **`ddl`** добавляют требуемые столбцы в ваши таблицы.

Чтобы включить необязательную возможность, запустите соответствующий сценарий для базы данных или указанной таблицы.

`dbType` - это тип базы данных, например, `sqlsvr` для Microsoft SQL Server, `ora` для Oracle или `db2` для IBM DB2.

Воспользуйтесь следующей таблицей для запуска сценариев базы данных для базы данных, чтобы создать новые или изменить существующие таблицы:

Таблица 2. Сценарии базы данных

Имя возможности	Сценарий возможности	Запустить для	Изменить
Глобальные предложения, подавление предложений и переопределение оценок	**aci_usrtab_***тип_бд***.sql** в каталоге *домашний_каталог_Interact*\ddl\acifeatures\ (каталог установки среды выполнения)	Ваша база данных профилей (userProdDataSource)	Создает таблицы UACI_DefaultOffers, UACI_BlackList и UACI_ScoreOverride.
Оценка	**aci_scoringfeature_***тип_бд***.sql** в каталоге *домашний_каталог_Interact*\ddl\acifeatures\ (каталог установки среды выполнения)	Таблицы переопределения оценок в вашей базе данных профилей (userProdDataSource)	Добавляет столбцы LikelihoodScore and AdjExploreScore.
Обучение	**aci_lrnfeature_***тип_бд***.sql** в каталоге *домашний_каталог_Interact*\interactDT\ddl\acifeatures\ (каталог установки среды разработки)	База данных Campaign, содержащая таблицы хронологии ваших контактов	Добавляет столбцы RTSelectionMethod, RTLearningMode и RTLearningModelID в таблицу UA_DtlContactHist. Кроме того, добавляет столбцы RTLearningMode и RTLearningModelID в таблицу UA_ResponseHistory. Этот сценарий также требуется для функций составления отчетов из дополнительного пакета отчетов .

Об отслеживании хронологии контактов и ответов

Можно сконфигурировать среду выполнения для записи хронологии контактов и ответов в таблицы хронологии контактов и ответов Campaign. Серверы среды выполнения хранят хронологию контактов и ответов в промежуточных таблицах. Модуль хронологии контактов и ответов копирует эти данные из промежуточных таблиц в таблицы хронологии контактов и ответов Campaign.

Модуль хронологии контактов и ответов работает, только если вы задали для свойств `Campaign > partitions > partition1 > Interact > interactInstalled` и `contactAndResponseHistTracking > isEnabled` на странице Конфигурация для среды разработки значение `yes`.

Если вы используете модуль отслеживания межсеансовых ответов, модуль хронологии контактов и ответов - это отдельный объект.

Типы контактов и ответов

С Interact можно записать один тип контакта и два типа ответа. Можно также записать дополнительные пользовательские типы ответов при помощи метода `postEvent`.

Свойства таблицы contactAndResponseHistTracking

В этой таблице перечисляются свойства из категории `contactAndResponseHistTracking`:

Событие	Тип контакта/ответа	Свойство конфигурации
Записать контакт для предложения	Контакт	установлен контакт
Записать принятие предложения	Ответ	принять
Записать отклонение предложения	Ответ	отклонить

Свойства таблицы UA_UsrResponseType

Убедитесь, что столбец CountsAsResponse таблицы UA_UsrResponseType в системных таблицах Campaign сконфигурирован правильно. Все типы ответов должны существовать в таблице UA_UsrResponseType.

Чтобы запись в таблице UA_UsrResponseType была допустимой, надо определить значение для всех столбцов в этой таблице, в том числе для CountsAsResponse. Допустимые значения для CountsAsResponse:

- 0 - нет ответа
- 1 - ответ
- 2 - отклонение
-

Эти ответы используются для создания отчетов.

Дополнительные типы ответов

В Interact можно использовать метод postEvent в API Interact для инициирования события, которое записывает в журнал действие "accept" или "reject" для предложения. Можно также расширить систему, разрешив вызову postEvent записывать дополнительные типы ответов, такие как Explore, Consider, Commit или Fulfill.

Все эти типы ответов должны существовать в таблице UA_UsrResponseType в системных таблицах Campaign. Используя параметры определенного события в методе postEvent, можно записывать дополнительные типы ответов и определить, надо ли включить принятие в обучение.

Для записи в журнал дополнительных типов ответов надо добавить следующие параметры события:

- **UACIResponseTypeCode** - строка, представляющая код типа ответов. Значение должно быть действительной записью в таблице UA_UsrResponseType.

  Чтобы запись в таблице UA_UsrResponseType была допустимой, надо определить значения для всех столбцов в этой таблице, в том числе для CountsAsResponse. Допустимые значения для CountsAsResponse: 0, 1 и 2. 0 указывает отсутствие ответа, 1 указывает ответ, а 2 - отклонение. Эти ответы используются для создания отчетов.
- **UACILogToLearning** - Число со значением 1 или 0. 1 указывает, что Interact должен записать в журнал событие как принятие для системы обучения или включить подавление предложений в сеансе. 0 указывает, что Interact не должен записывать в журнал событие для системы обучения или включать подавление предложений в сеансе. Этот параметр позволяет создать несколько различных методов postEvent,

записывающих в журнал различные типы ответов, не влияя при этом на обучение. Если вы не определяете UACILogToLearning, Interact предполагает значение по умолчанию 0.

Если при отправке события принятия задается responseTypeCode, это предложение не подавляется при принятии. Независимо от значения ResponseTypeCode (например, 0, 1, 2), если для logToLearningAsAccept задано значение 0, это предложение никогда не будет подавляться. Чтобы подавить это предложение, не задавайте в вашем postEvent параметр UACIResponseTypeCode. Если же параметр UACIResponseTypeCode задан, для UACILogToLearning надо задать значение 1, когда вы хотите, чтобы это предложение было подавлено.

Вам может понадобиться создать несколько событий с действием Записать принятие предложения: по одному для каждого типа ответов для записи в журнал, или одно событие с действием Записать принятие предложения для каждого вызова postEvent, используемого для записи отдельных типов ответов.

Например, создайте событие с действием Записать принятие предложения для каждого типа ответа. Вы задаете следующие пользовательские ответы в таблице UA_UsrResponseType [в качестве имени (код)]: Исследовать (EXP), Рассмотреть (CON) и Принять (CMT). После этого вы создаете три события и присваиваете им имена LogAccept_Explore, LogAccept_Consider и LogAccept_Commit. Все три события абсолютно одинаковы (с действием Записать принятие предложения), но у них разные имена, чтобы тот, кто работает с API, мог их различать.

Или можно создать одно событие с действием Записать принятие предложения, используемое для всех пользовательских типов ответов. Например, присвойте ему имя LogCustomResponse.

При работе с API нет никаких функциональных различий между этими событиями, но правила именования могут сделать код более понятным. Также, если присвоить каждому пользовательскому ответу отдельное имя, в отчете Сводная информация об интенсивности событий канала будут показаны более точные данные.

Сначала зададим все пары значение - имя

```
//Определяем пары значение - имя для UACIResponseTypeCode
// Тип ответа Explore
NameValuePair responseTypeEXP = new NameValuePairImpl();
responseTypeEXP.setName("UACIResponseTypeCode");
responseTypeEXP.setValueAsString("EXP");
responseTypeEXP.setValueDataType(NameValuePair.DATA_TYPE_STRING);

// Тип ответа Consider
NameValuePair responseTypeCON = new NameValuePairImpl();
responseTypeCON.setName("UACIResponseTypeCode");
responseTypeCON.setValueAsString("CON");
responseTypeCON.setValueDataType(NameValuePair.DATA_TYPE_STRING);

// Тип ответа Commit
NameValuePair responseTypeCMT = new NameValuePairImpl();
responseTypeCMT.setName("UACIResponseTypeCode");
responseTypeCMT.setValueAsString("CMT");
responseTypeCMT.setValueDataType(NameValuePair.DATA_TYPE_STRING);

//Определяем пары значение - имя для UACILOGTOLEARNING
//Не записывать в журнал для обучения
NameValuePair noLogToLearning = new NameValuePairImpl();
noLogToLearning.setName("UACILOGTOLEARNING");
noLogToLearning.setValueAsString("0");
noLogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

```
//Записывать в журнал для обучения
NameValuePair LogToLearning = new NameValuePairImpl();
LogToLearning.setName("UACILogToLearning");
LogToLearning.setValueAsString("1");
LogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

В первом примере показано использование отдельных событий.

```
//ПРИМЕР 1: Этот набор вызовов postEvent использует названные по отдельности события
//PostEvent с ответом Explore
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogAccept_Explore, postEventParameters);

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

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

Второй пример показывает использование единственного события.

```
//ПРИМЕР 2: Этот набор вызовов postEvent использует единственное событие
//PostEvent с ответом Explore
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogCustomResponse, postEventParameters);

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

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

Оба примера выполняют одни и те же действия, однако одна версия, возможно, читается легче другой.

Отображение промежуточных таблиц среды выполнения на таблицы хронологии Campaign

Промежуточные таблицы хронологии контактов Interact отображаются на таблицы хронологии Campaign. У вас должны быть по одной промежуточной таблице среды выполнения на каждый уровень аудитории.

Отображение промежуточной таблицы хронологии контактов UACI_CHStaging

В этой таблице показано, как промежуточная таблица среды выполнения `UACI_CHStaging` отображается на таблицу хронологии контактов Campaign. Показанные имена таблиц - это таблицы примера, созданные для аудитории по умолчанию в таблицах среды выполнения и системных таблицах Campaign.

Таблица 3. Хронология контактов

UACI_CHStaging **Имена столбцов промежуточной таблицы хронологии контактов Interact**	**Таблица хронологии контактов Campaign**	**Имя столбца таблицы**
ContactID	Н/П	Н/П
TreatmentCode	UA_Treatment	TreatmentCode

Таблица 3. Хронология контактов (продолжение)

UACI_CHStaging **Имена столбцов промежуточной таблицы хронологии контактов Interact**	**Таблица хронологии контактов Campaign**	**Имя столбца таблицы**
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` - ключевой столбец для объединения таблицы `UACI_CHOfferAtrib` с таблицей `UACI_CHStaging`. Столбец `userDefinedFields` может содержать любые данные по вашему выбору.

Отображение промежуточной таблицы хронологии контактов UACI_CHOfferAttrib

В этой таблице показано, как промежуточная таблица среды выполнения `UACI_CHOfferAttrib` отображается на таблицу хронологии контактов Campaign. Показанные имена таблиц - это таблицы примера, созданные для аудитории по умолчанию в таблицах среды выполнения и системных таблицах Campaign.

Таблица 4. Атрибуты предложения

UACI_CHOfferAttrib **Имена столбцов промежуточной таблицы хронологии контактов Interact**	**Таблица хронологии контактов Campaign**	**Имя столбца таблицы**
ContactID	Н/П	Н/П
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

Отображение промежуточной таблицы хронологии контактов и ответов UACI_RHStaging

В этой таблице показано, как промежуточная таблица среды выполнения `UACI_RHStaging` отображается на таблицу хронологии ответов Campaign. Показанные имена таблиц - это таблицы примера, созданные для аудитории по умолчанию в таблицах среды выполнения и системных таблицах Campaign.

Таблица 5. Хронология ответов

UACI_RHStaging **Имена столбцов промежуточной таблицы хронологии ответов Interact**	**Таблица хронологии ответов Campaign**	**Имя столбца таблицы**
SeqNum	Н/П	Н/П
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime
ResponseType	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

SeqNum - это ключ, используемый модулем хронологии контактов и ответов для идентификации данных, но он не записывается в таблицы ответов Campaign. Столбец userDefinedFields может содержать любые данные по вашему выбору.

Дополнительные столбцы в промежуточных таблицах

Если вы добавляете столбцы в промежуточные таблицы подготовки, модуль хронологии контактов и ответов записывает их в таблицы UA_DtlContactHist или UA_ResponseHistory в столбцы с тем же именем.

Например, если вы добавляете столбец linkFrom в вашу таблицу UACI_CHStaging, модуль хронологии контактов и ответов копируют эти данные в столбец linkFrom в таблице UA_DtlContactHist.

Дополнительные столбцы в таблицах хронологии контактов и ответов Campaign

Если у вас есть дополнительные столбцы в вашей таблице хронологии контактов и ответов Campaign, добавьте соответствующие столбцы в промежуточные таблицы перед запуском модуля хронологии контактов и ответов.

Вы заполняете дополнительные столбцы в промежуточных таблицах, создавая столбцы с теми же именами как пары имя - значение в данных сеанса среды выполнения.

Например, вы создаете пары имя - значение NumberItemsInWishList и NumberItemsInShoppingCart и добавляете их в вашу таблицу UACI_RHStaging. Когда происходит событие Запись в журнал принятия предложения или Запись в журнал отклонения предложения, среда выполнения заполняет эти поля. Среда выполнения заполняет таблицу UACI_CHStaging, когда происходит событие Запись в журнал контакта предложения.

Использование таблиц для включения оценок предложений

Вы можете использовать пользовательские поля, чтобы включить оценку, применяемую для представления предложения. Добавьте столбец с именем FinalScore и в таблицу UACI_CHStaging в таблицах среды выполнения, и в таблицу UA_DtlContactHist в системных таблицах Campaign. Interact автоматически заполняет столбец FinalScore итоговой оценкой, используемой для предложения, если вы применяете встроенное обучение.

Если вы создаете настроенный модуль обучения, можно использовать метод `setActualValueUsed` интерфейса `ITreatment` и метод `logEvent` интерфейса `ILearning`.

Если вы не используете обучение, добавьте столбец с именем `Score` и в таблицу `UACI_CHStaging` в таблицах среды выполнения, и в таблицу `UA_DtlContactHist` в системных таблицах Campaign. Interact автоматически заполняет столбец `Score` оценкой, используемой для предложения.

Создание новых таблиц хронологии в Campaign и промежуточных таблиц в Interact

Если вы используете уровень аудитории, отличный от уровня `Покупатель`, надо будет создать новые таблицы хронологии в Campaign и новые промежуточные таблицы в Interact.

Например, приведенный ниже демонстрационный сценарий используется в базе данных среды разработки DB2 IBM для создания таблиц хронологии в Campaign для уровня аудитории типа `Счет`.

```
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;
```

Приведенный ниже демонстрационный сценарий используется в базе данных DB2 IBM среды выполнения для создания промежуточных таблиц хронологии в Interact для уровня аудитории типа Счет.

```
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);
```

Конфигурирование мониторинга JMX для модуля хронологии контактов и ответов.

Выполните эту процедуру, чтобы сконфигурировать мониторинг JMX для модуля хронологии контактов и ответов. Поддерживаются протоколы JMXMP и RMI. Конфигурирование мониторинга JMX не включает защиту для модуля хронологии контактов и ответов. Для конфигурирования мониторинга используется Marketing Platform для среды разработки.

Об этой задаче

При использовании инструмента мониторинга JMX для модуля хронологии контактов и ответов адрес по умолчанию:

- Протокол JMXMP - `service:jmx:jmxmp://`*`Сервер Campaign`*`:`*`порт`*`/campaign`.
- Протокол RMI - `service:jmx:rmi:///jndi/rmi://`*`Сервер Campaign`*`:`*`порт`*`/campaign`.

При просмотре данных в инструменте мониторинга JMX атрибуты результатов организованы по разделу, затем по уровню аудитории.

Процедура

В Marketing Platform для среды разработки отредактируйте следующие свойства конфигурации в категории `Campaign > monitoring`.

Свойство конфигурации	Значение
`monitorEnabledForInteract`	**True**
port	Номер порта для службы JMX
protocol	Протокол: • **JMXMP** • **RMI** Защита для модуля хронологии контактов и ответов не применяется, даже если выбрать протокол JMXMP.

О межсеансовом отслеживании ответов

Посетители не всегда могут выполнить транзакцию за одно посещение вашей точки контакта. Покупатель может добавить позицию в корзину на вашем веб-сайте и задержать завершение ее покупки на два дня. Бессрочное сохранение сеанса среды выполнения в активном состоянии нецелесообразно. Можно включить поддержку межсеансового отслеживания ответов, чтобы отследить представление предложения в одном сеансе и получить для него соответствие с ответом в другом сеансе.

Межсеансовое отслеживание ответов Interact может получать соответствия для кодов процедур или других кодов по умолчанию. Его можно также сконфигурировать для получения соответствий для любого пользовательского кода по вашему выбору. Межсеансовое отслеживание ответов получает соответствия для доступных данных.

Например, ваш веб-сайт включает в себя предложение с промокодом, генерируемым во время вывода, для скидки, действительной в течение одной недели. Пользователь может добавить позиции в свою корзину, но не завершать их покупку в течение трёх дней. При использовании вызова `postEvent` для записи в журнал события принятия можно включить только промокод. Поскольку среда выполнения не может найти код процедуры или предложения для получения соответствия в текущем сеансе, она вкладывает событие принятия с доступной информацией в промежуточную таблицу межсеансовых ответов (XSessResponse). Служба CrossSessionResponse периодически читает таблицу XSessResponse и пытается получить соответствие записей с доступными данными хронологии контактов. Служба CrossSessionResponse сопоставляет промокод с хронологией контактов и собирает все требуемые данные для записи в журнал правильного ответа. Затем служба CrossSessionResponse записывает ответ в промежуточные таблицы ответов и (если включена поддержка обучения) в таблицы обучения. Затем модуль хронологии контактов и ответов записывает ответ в таблицы хронологии контактов и ответов Campaign. Успешная обработка межсеансового ответа зависит от исходных записей хронологии контактов, которая была перенастроена ETL хронологии контактов в базу данных Campaign.

Конфигурация источника данных для межсеансового отслеживания ответов

Межсеансовое отслеживание ответов Interact сопоставляет данные сеансов из среды выполнения с хронологией контактов и ответов Campaign. По умолчанию межсеансовое отслеживание ответов получает соответствие для кода процедуры или кода предложения. Среду выполнения можно сконфигурировать для получения соответствия для пользовательского, альтернативного кода.

- Если выбрать получение соответствия для альтернативного кода, альтернативный код нужно определить в таблице `UACI_TrackingType` в таблицах среды выполнения Interact.
- У среды выполнения должен быть доступ к таблицам хронологии контактов Campaign. Можно либо сконфигурировать среду выполнения для получения доступа к таблицам хронологии контактов Campaign, либо создать в среде выполнения копию таблиц хронологии контактов.

  Это доступ только для чтения, он осуществляется отдельно от утилиты хронологии контактов и ответов.

  Если вы создаёте копию таблиц, вам надо убедиться в точности данных в копии хронологии контактов. Можно сконфигурировать промежуток времени хранения службой CrossSessionResponse несопоставленных ответов для сопоставления в соответствии с частотой обновления вами данных в копии таблиц хронологии контактов при помощи свойства `purgeOrphanResponseThresholdInMinutes`. В случае использования модуля хронологии контактов и ответов нужно координировать обновления ETL, чтобы у вас гарантированно были самые последние данные.

Конфигурирование таблиц хронологии контактов и ответов для межсеансового отслеживания ответов

Создаёте ли вы копию таблиц хронологии контактов или используете фактические таблицы в системных таблицах Campaign, нужно выполнить следующие действия по конфигурированию таблиц хронологии контактов и ответов.

Прежде чем начать

Перед выполнением этих действий таблицы хронологии контактов и ответов уже должны быть правильно отображены в Campaign.

Процедура

1. Запустите сценарий SQL `aci_lrnfeature` в каталоге `interactDT/ddl/acifeatures` установки среды разработки Interact для таблиц `UA_DtlContactHist` и `UA_ResponseHistory` в системных таблицах Campaign.

   При этом столбец `RTSelectionMethod` будет добавлен в таблицы `UA_DtlContactHist` и `UA_ResponseHistory`. Запустите сценарий `aci_lrnfeature` для этих таблиц для каждого из используемых уровней аудитории. Отредактируйте этот сценарий должным образом, чтобы он работал с правильной таблицей для каждого используемого уровня аудитории.

2. Если вы хотите скопировать таблицы хронологии контактов в среду выполнения, сделайте это сейчас.

   Если вы создаете копию таблиц хронологии контактов Campaign, доступных среде выполнения, для поддержки межсеансового отслеживания ответов, используйте следующие рекомендации:

   - Для межсеансового отслеживания ответов требуется доступ к этим таблицам только для чтения.
   - Для межсеансового отслеживания ответов требуются следующие таблицы из хронологии контактов Campaign.
     - `UA_DtlContactHist` (для каждого уровня аудитории)
     - `UA_Treatment`

     Для гарантии точности отслеживания ответов нужно обновлять данные в этих таблицах на регулярной основе.

3. Запустите сценарий SQL `aci_crhtab` в каталоге `ddl` установки среды выполнения Interact для источника данных хронологии контактов и ответов.

   Этот сценарий создаёт таблицы `UACI_XsessResponse` и `UACI_CRHTAB_Ver`.

4. Создайте для каждого уровня аудитории версию таблицы `UACI_XSessResponse`.

Результаты

Для повышения производительности межсеансового отслеживания ответов, возможно, вы захотите ограничить объём данных хронологии контактов либо способом, при котором надо скопировать данные хронологии контактов, либо путем конфигурирования представления в таблицы хронологии контактов Campaign. Например, если у вас принято, что никакое из предложений не остается действительным более 30 дней, нужно ограничить данные хронологии контактов последними 30 днями. Чтобы изменить число дней хранения данных хронологии контактов, откройте свойство конфигурации у **Campaign | partitions | partition*n*| Interact | contactAndResponseHistTracking** и задайте значение **daysBackInHistoryToLookupContact**.

Вы не увидите результатов межсеансового отслеживания ответов, пока не будет запущен модуль хронологии контактов и ответов. Например, значение `processSleepIntervalInMinutes` по умолчанию - 60 минут. Поэтому прежде, чем межсеансовые ответы появятся в хронологии ответов Campaign, может пройти более часа.

Таблица UACI_TrackingType

Таблица UACI_TrackingType - это одна из таблиц среды выполнения. Эта таблица определяет коды отслеживания для отслеживания межсеансовых ответов. Код отслеживания определяет, при помощи какого метода среда выполнения сопоставляет текущее предложение в своем сеансе с хронологией контактов и ответов.

Вертикальная столбчатая диаграмма	Тип	Описание
TrackingCodeType	int	Число, представляющее тип кода отслеживания. Это число упоминается в командах SQL, используемых для сопоставления информации данных сеанса с таблицами хронологии контактов и ответов.
Имя	varchar(64)	Имя для типа кода отслеживания. Передается данным сеанса с помощью зарезервированного параметра UACI_TrackingCodeType метода postEvent.
Описание	varchar(512)	Краткое описание типа кода отслеживания. Это необязательное поле.

По умолчанию в среде выполнения заданы два типа кода отслеживания, как показано в следующей таблице. Для любого другого кода необходимо задать уникальный TrackingCodeType.

TrackingCodeType	Имя	Описание
1	Код процедуры	Код процедуры, сгенерированный UACI
2	Код предложения	Код UAC предложения кампании

UACI_XSessResponse

Таблица UACI_XSessResponse - это одна из таблиц среды выполнения. Эта таблица используется для отслеживания межсеансовых ответов.

В источнике данных хронологии контактов и ответов, доступном для отслеживания межсеансовых ответов Interact должен существовать один экземпляр этой таблицы.

Вертикальная столбчатая диаграмма	Тип	Описание
SeqNumber	bigint	Идентификатор для строки данных. Служба CrossSessionResponse обрабатывает все записи в порядке SeqNumber.
ICID	bigint	ID интерактивного канала
AudienceID	bigint	ID аудитории для данного уровня аудитории. Имя этого столбца должно совпадать ID аудитории, заданным в Campaign. Пример таблицы содержит столбец CustomerID.
TrackingCode	varchar(64)	Значение, переданное параметром UACIOfferTrackingCode метода postEvent.
TrackingCodeType	int	Числовое представление кода отслеживания. Значение должно быть действительной записью в таблице UACI_TrackingType.

Вертикальная столбчатая диаграмма	Тип	Описание
OfferID	bigint	ID предложения, как он определен в Campaign.
ResponseType	int	Тип ответа для этой записи. Значение должно быть действительной записью в таблице UA_UsrResponseType.
ResponseTypeCode	varchar(64)	Код типа ответа для этой записи. Значение должно быть действительной записью в таблице UA_UsrResponseType.
ResponseDate	Дата/Время	Дата ответа.
Mark	bigint	Значение в этом поле задает состояние записи. • 1 - В процессе • 2 - Успешно • NULL - Повторить • -1 - Запись находится в базе данных более purgeOrphanResponseThresholdInMinutes минут. В рамках обслуживания этой таблицы администратором базы данных можно проверить это поле на наличие записей без соответствий, то есть записей со значением -1. Все записи со значением 2 будут автоматически удалены службой CrossSessionResponse.
UsrDefinedFields	char(18)	Любые пользовательские поля, которые вы можете добавить при сопоставлении ответов на предложения с хронологией контактов и ответов. Например, если вы собираетесь проводить сопоставление по промокоду, добавьте пользовательское поле промокода.

Включение межсеансового отслеживания ответов

Используйте следующую процедуру для включения межсеансового отслеживания ответов.

Прежде чем начать

Для использования всех преимуществ межсеансового отслеживания ответов нужно сконфигурировать модуль хронологии контактов и ответов.

Для использования межсеансового отслеживания ответов нужно сконфигурировать среду выполнения, чтобы получить доступ для чтения к таблицам хронологии контактов и ответов Campaign. Информацию можно будет читать либо из фактических таблиц хронологии контактов и ответов Campaign в среде разработки, либо из копии таблиц в источниках данных среды выполнения. Конфигурация среды выполнения для получения доступа только для чтения к таблице хронологии контактов и ответов создаётся отдельно от какой-либо конфигурации модуля хронологии контактов и ответов.

В случае получения соответствия для кода, иного чем код процедуры или код предложения, этот код нужно добавить в таблицу UACI_TrackingType.

Процедура

1. Создайте таблицы XSessResponse в таблицах хронологии контактов и ответов, доступных среде выполнения.
2. Определите свойства в категории `contactAndResponseHistoryDataSource` для среды выполнения.
3. Определите свойство `crossSessionResponseTable` для каждого уровня аудитории.
4. Создайте для каждого уровня аудитории категорию `OverridePerAudience`.

Сопоставление предложений для межсеансовых ответов

По умолчанию межсеансовое отслеживание ответов получает соответствия для кодов процедур или кодов предложений. Служба crossSessionResponse использует команды SQL для сопоставления кодов процедур кодов предложений или пользовательского кода из данных сеанса с таблицами хронологии контактов и ответов Campaign. Эти команды SQL можно отредактировать для сопоставления всех настроек, вносимых вами в коды отслеживания, коды предложений или пользовательский код.

Соответствие по коду процедуры

SQL для получения соответствия по коду процедуры должен возвратить все столбцы в таблице XSessResponse для этого уровня аудитории плюс столбец `OfferIDMatch`. В столбце `OfferIDMatch` должно быть значение `offerId`, поступающее с кодом процедуры записи XSessResponse.

Ниже приведён пример генерируемой команды SQL по умолчанию, сопоставляющей коды процедур. Interact генерирует SQL с целью использования правильных имен таблиц для уровня аудитории. Этот SQL используется, если для свойства `Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byTreatmentCode > SQL` задано значение **Использовать SQL, сгенерированный системой**.

```
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
```

Значения `UACI_XsessResponse`, `UA_DtlContactHist`, `CustomerID` и `UA_ContactHistory` определяются заданными вами значениями в Interact. Например, `UACI_XsessResponse` определяется свойством конфигурации `Interact > profile > Audience Levels > [имя_уровня_аудитории] > crossSessionResponseTable`.

Если вами были настроены таблицы хронологии контактов и ответов, возможно, этот SQL потребуется скорректировать для работы с используемыми таблицами. Переопределения SQL задаются в свойстве `Interact > services > crossSessionResponse > OverridePerAudience > *(уровень_аудитории)* > TrackingCodes > byTreatmentCode > OverrideSQL`. При задании какого либо SQL переопределения потребуется также изменить свойство `SQL` на **Override SQL**.

Соответствие по коду предложения

SQL для получения соответствия по коду предложения должен возвратить все столбцы в таблице XSessResponse для этого уровня аудитории плюс столбец TreatmentCodeMatch. Значение в столбце TreatmentCodeMatch - это код обработки, поступающий с ID предложения (и кодом предложения) в записи XSessResponse.

Ниже приведён пример генерируемой команды SQL по умолчанию, сопоставляющей коды предложений. Interact генерирует SQL с целью использования правильных имен таблиц для уровня аудитории. Этот SQL используется, если для свойства Interact > services > crossSessionResponse > OverridePerAudience > *уровень_аудитории* > TrackingCodes > byOfferCode > SQL задано значение **Использовать SQL, сгенерированный системой**.

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

Значения UACI_XsessResponse, UA_DtlContactHist, CustomerID и UA_ContactHistory определяются заданными вами значениями в Interact. Например, UACI_XsessResponse определяется свойством конфигурации Interact > profile > Audience Levels > [имя_уровня_аудитории] > crossSessionResponseTable.

Если вами были настроены таблицы хронологии контактов и ответов, возможно, этот SQL потребуется скорректировать для работы с используемыми таблицами. Переопределения SL задаются в свойстве Interact > services > crossSessionResponse > OverridePerAudience > *(уровень_аудитории)* > TrackingCodes > byOfferCode > OverrideSQL. При задании какого либо SQL переопределения потребуется также изменить свойство SQL на **Override SQL**.

Соответствие по альтернативному коду

Можно определить команду SQL для сопоставления по некоторому дополнительному коду по вашему выбору. Например, у вас могут быть промокоды коды или коды продуктов, отдельные от кодов предложений или процедур.

Этот альтернативный код нужно определить в таблице UACI_TrackingType в таблицах среды выполнения Interact.

Нужно задать SQL или хранимую процедуру в свойстве Interact > services > crossSessionResponse > OverridePerAudience > *(уровень_аудитории)* > TrackingCodes > byAlternateCode > OverrideSQL, возвращающем все столбцы в таблице XSessResponse для этого уровня аудитории плюс столбцы TreatmentCodeMatch и OfferIDMatch. Дополнительно возможен возврат offerCode в OfferIDMatch (в форме offerCode1, offerCode2, ... offerCode*N* - для кодов предложений, состоящих из N частей. Значения в столбце TreatmentCodeMatch и OfferIDMatch (или столбцах кодов предложений) должны соответствовать значению TrackingCode в записи XSessResponse.

Например, следующий псевдокод SQL получает соответствие для столбца AlternateCode в таблице XSessResponse.

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

Где <*x*> - код отслеживания, определяемый в таблице UACI_TrackingType.

Использование утилиты загрузки базы данных со средой выполнения

По умолчанию среда выполнения записывает данные хронологии контактов и ответов из данных сеанса в промежуточные таблицы. Однако в очень активной производственной системе объем памяти, требуемой для записи в кэш всех данных, прежде чем они будут записаны средой выполнения в промежуточные таблицы, может оказаться чрезмерно большим. Для повышения производительности можно сконфигурировать среду выполнения для использования утилиты загрузки базы данных.

При включении утилиты загрузки базы данных вместо хранения всей хронологии контактов и ответов в памяти до ее записи в промежуточные таблицы среда выполнения записывает эти данные в промежуточный файл. Положение каталога,

содержащего эти промежуточные файлы, задается свойством `externalLoaderStagingDirectory`. У этого каталога есть несколько подкаталогов. Первый подкаталог - это каталог экземпляра среды выполнения, который содержит каталоги `contactHist` и `respHist`. Каталоги `contactHist` и `respHist` содержат уникально именованные подкаталоги в формате *имя_уровня_аудитории.уникальный_ID.текущее_состояние*, где содержатся промежуточные файлы.

Текущее состояние	Описание
CACHE	Содержимое каталога, в настоящее время записываемое в файл.
READY	Содержимое каталога, готовое для обработки.
RUN	Содержимое каталога, в настоящее время записываемое в базу данных.
PROCESSED	Содержимое каталога, уже записанное в базу данных.
ERROR	При записи содержимого каталога в базу данных произошла ошибка.
ATTN	Содержимое каталога требует внимания. Это значит, что вам может потребоваться выполнить несколько шагов вручную для завершения записи содержимого этого каталога в базу данных.
RERUN	Содержимое каталога готово для записи в базу данных. После исправления ошибок нужно переименовать каталог `ATTN` или `ERROR` в `RERUN`.

Каталог экземпляра среды выполнения можно определить, задав свойство JVM interact.runtime.instance.name в сценарии запуска сервера прикладных программ. Например, в сценарий запуска сервера веб-программ можно добавить `-Dinteract.runtime.instance.name=instance2`. Если имя не задано, по умолчанию используется имя `DefaultInteractRuntimeInstance`.

Каталог `samples` содержит образцы файлов, помогающие вам в написании собственных файлов управления утилитой загрузки базы данных.

Включение поддержки утилиты загрузки базы данных со средой выполнения

Эта процедура используется для включения поддержки утилиты загрузки базы данных со средой выполнения.

Прежде чем начать

Для утилиты загрузки базы данных нужно определить все файлы команд и файлы управления перед тем, как конфигурировать для их использования среду выполнения. Эти файлы должны существовать в одном и том же положении на всех серверах среды выполнения в одной и той же группе серверов.

Interact предоставляет примеры файлов команд и файлов управления в каталоге `loaderService` установки серверов среды выполнения Interact.

Процедура

1. Подтвердите, что у пользователя среды выполнения есть регистрационные данные для входа в систему для источника данных таблиц среды выполнения, определённого в свойствах конфигурации в `Interact > general > systemTablesDataSource`.

2. Определите свойства конфигурации `Interact > general > systemTablesDataSource > loaderProperties`.
3. Определите свойство `Interact > services >externalLoaderStagingDirectory`.
4. При необходимости скорректируйте свойства конфигурации `Interact > services > responseHist > fileCache`.
5. При необходимости скорректируйте свойства конфигурации `Interact > services > contactHist > fileCache`.
6. Перезапустите сервер среды выполнения.

Процесс ETL паттерна событий

Для обработки больших объемов данных паттерна событий IBM Interact и для того, чтобы сделать эти данными доступными для запросов и отчетов, в целях оптимизации производительности можно установить автономный процесс Extract, Transform, Load (ETL) на любом поддерживаемом сервере.

В Interact все данные паттернов событий для данного AudienceID сохраняются в одном наборе в таблицах базы данных среды выполнения. AudienceID и информации о состоянии паттерна хранятся как большой двоичный объект (Binary Large Object, BLOB). Для выполнения любых запросов SQL или отчетов на основании паттернов событий этот новый процесс ETL необходим, чтобы разбить объект по таблицам в базе данных назначения. Чтобы это осуществить, автономный процесс ETL берет данные о паттернах событий из таблиц базы данных среды выполнения Interact, обрабатывает их по заданному вами расписанию и сохраняет их в базе данных назначения, где они будут доступны для запросов SQL или дополнительных отчетов.

Кроме перемещения и преобразования данных паттернов событий в базу данных назначения, автономный процесс ETL синхронизирует также данные в базе данных назначения с самой новой информацией в вашей базе данных среды выполнения Interact. Например, если удалить паттерн событий в среде выполнения Interact, обработанные данные этого паттерна событий будут удалены из базы данных назначения при следующем запуске процесса ETL. Кроме этого, хранится актуальная информация о состоянии паттернов. Поэтому хранимая в базе данных назначения информация о паттернах событий - это исключительно текущие, а не хронологические данные.

Выполнение автономного процесса ETL

При запуске автономного процесса ETL на сервере он работает непрерывно в фоновом режиме, пока не будет остановлен. Этот процесс следует инструкциям свойств конфигурации Marketing Platform для определения частоты, соединений с базой данных и других подробностей при своем выполнении.

Прежде чем начать

Прежде чем запускать автономный процесс ETL, убедитесь, что вы выполнили следующие задачи:

- У вас должны быть разрешения роли пользователя администратора Interact.
- У вас должен быть установлен процесс на сервере, и файлы на сервере и в Marketing Platform должны быть правильно сконфигурированы для вашей конфигурации.

Примечание:

При запуске процесса ETL в Microsoft Windows с отличным от английского (US English) языком используйте `chcp` в командной строке, чтобы задать кодовую

страницу для используемого языка. Например, можно использовать один из следующих кодов: ja_jp=932, zh_cn=936, ko_kr=949, ru_ru=1251; для de_de, fr_fr, it_it, es_es, pt_br используйте 1252. Для правильного вывода символов используйте `chcp` в командной строке Windows, прежде чем запускать процесс ETL.

Об этой задаче

После установки и конфигурирования автономного процесса ETL вы готовы к запуску самого процесса.

Процедура

1. Откройте командную строку на сервере, где установлен процесс ETL.
2. Перейдите в каталог `<домашний_каталог_Interact>/PatternStateETL/bin`, в котором находятся исполняемые файлы для процесса ETL.
3. Запустите файл `command.bat` (для Microsoft Windows) или `command.sh` (для UNIX-подобных операционных систем) со следующими параметрами:
   - `-u <имя_пользователя>`. Это значение должно представлять допустимого пользователя Marketing Platform, и этот пользователь должен быть сконфигурирован с правом доступа к источникам данных **TargetDS** и **RuntimeDS**, которые будет использовать процесс ETL.
   - `-p <пароль>`. Замените *<пароль>* на пароль, соответствующий заданному вами пользователю. Если пароль для этого пользователя пуст, задайте пару двойных кавычек (как в `-p ""`). При запуске файла command пароль не обязателен; если при вводе command не использовать пароль, поступит предложение ввести его при запуске команды.
   - `-c <имя_профиля>`. Замените *<имя_профиля>* на точное имя, указанное Marketing Platform в созданной вами конфигурации **Interact | PatternStateETL**.

     Введенное вами имя должно совпадать со значением, указанным в поле **Имя новой категории** при создании конфигурации.
   - `start`. Команда start требуется для запуска процесса.

   Тем самым, полная команда для запуска процесса будет выглядеть следующим образом:

   ```
   command.bat -u <имя_пользователя> -p <пароль> -c <имя_профиля> start
   ```

Результаты

Автономный процесс ETL будет запущен и продолжит свое выполнение в фоновом режиме до остановки процесса или до перезапуска сервера.

Примечание:

При первом запуске этого процесса из-за аккумулированных данных паттернов событий может потребоваться существенное время. При следующих запусках процесса будет обрабатываться только самый последний набор данных паттернов событий, и для выполнения потребуется меньшее время.

Помните также, что можно использовать аргумент `help` для файла `command.bat` или `command.sh`, чтобы увидеть все возможные опции, как в следующем примере:

```
command.bat help
```

Остановка автономного процесса ETL

При запуске автономного процесса ETL на сервере он работает непрерывно в фоновом режиме, пока не будет остановлен.

Об этой задаче

Процедура

1. Откройте командную строку на сервере, где установлен процесс ETL.
2. Перейдите в каталог `<домашний_каталог_Interact>/PatternStateETL/bin`, в котором находятся исполняемые файлы для процесса ETL.
3. Запустите файл `command.bat` (для Microsoft Windows) или `command.sh` (для аналогичных UNIX операционных систем) со следующими параметрами:
   - `-u <имя_пользователя>`. Это значение должно представлять допустимого пользователя Marketing Platform, и этот пользователь должен быть сконфигурирован с правом доступа к источникам данных **TargetDS** и **RuntimeDS**, которые будет использовать процесс ETL.
   - `-p <пароль>`. Замените *<пароль>* на пароль, соответствующий заданному вами пользователю. Если пароль для этого пользователя пуст, задайте пару двойных кавычек (как в `-p ""`). При запуске файла command пароль не обязателен; если при вводе command не использовать пароль, поступит предложение ввести его при запуске команды.
   - `-c <имя_профиля>`. Замените *<profileName>* на точное имя, указанное Marketing Platform в созданной вами конфигурации **Interact | PatternStateETL**.

     Введенное вами имя должно совпадать со значением, указанным в поле **Имя новой категории** при создании конфигурации.
   - `stop`. Команда stop требуется для остановки процесса. Если использовать эту команду, любая выполняемая операция ETL будет завершена, прежде чем выключится сам процесс.

     Чтобы завершить процесс ETL, не дожидаясь окончания любой выполняемой операции, используйте `forcestop` вместо `stop`.

   Тем самым, полная команда для остановки процесса будет выглядеть следующим образом:

   ```
   command.bat -u <имя_пользователя> -p <пароль> -c <имя_профиля> stop
   ```

Результаты

Автономный процесс ETL остановится.

Глава 4. Представление предложений

Вы можете сконфигурировать Interact различными способами, чтобы улучшить выбор предложений для представления. Приведенные ниже разделы описывают эти дополнительные возможности подробно.

Соответствие предложений требованиям

Задача Interact - представлять подходящие предложения. По сути дела, Interact представляет оптимальные предложения из всех подходящих, исходя из посетителя, канала и ситуации.

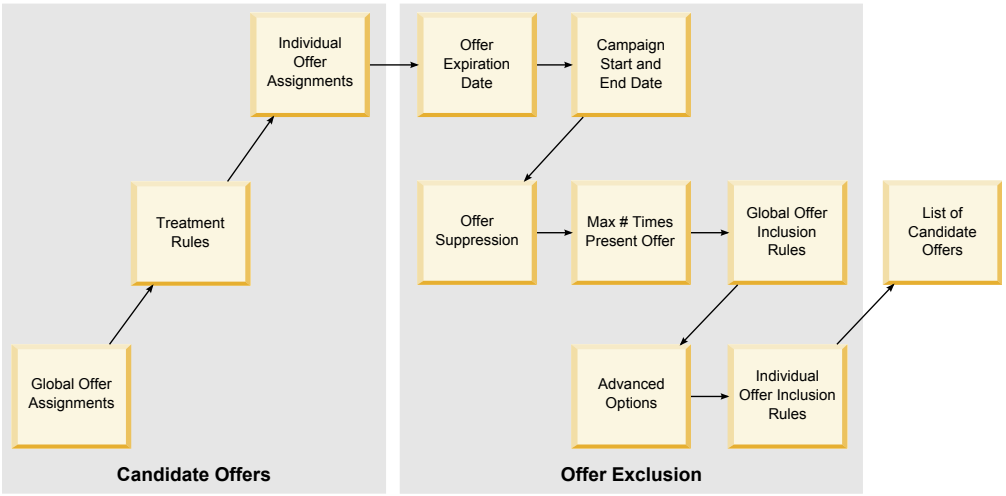
Правила процедуры - это только исходный пункт в процессе определения продуктом Interact, какие предложения подходят для покупателя. Interact содержит несколько дополнительных возможностей, с помощью которых можно улучшить работу среды выполнения по отбору предложений для представления. Ни одна из этих возможностей не гарантирует, что предложение будет представлено клиенту. Эти возможности влияют на вероятность того, что предложение окажется подходящим для представления его покупателю. Для достижения наилучшего решения для вашей среды можно использовать любое число этих возможностей.

Есть три основных области, в которых вы можете оказать влияние на пригодность предложений: создание списка предложений-кандидатов, определение маркетинговой оценки и обучение.

Создание списка предложений-кандидатов

Создание списка предложений-кандидатов состоит из двух основных этапов. На первом этапе генерируется список всех возможных предложений, которые могут подходить для покупателя. На втором этапе отфильтровываются предложения, которые больше не подходят для этого покупателя. На обоих этапах существует несколько точек, в которых можно влиять на создание списка предложений-кандидатов.

На этой диаграмме показаны этапы создания списка предложений-кандидатов. Стрелки показывают порядок очередности. Например, если предложение прошло фильтр **Максимальное число представлений предложения**, но не прошло фильтр **Правила включения глобальных предложений**, среда выполнения исключит это предложение.

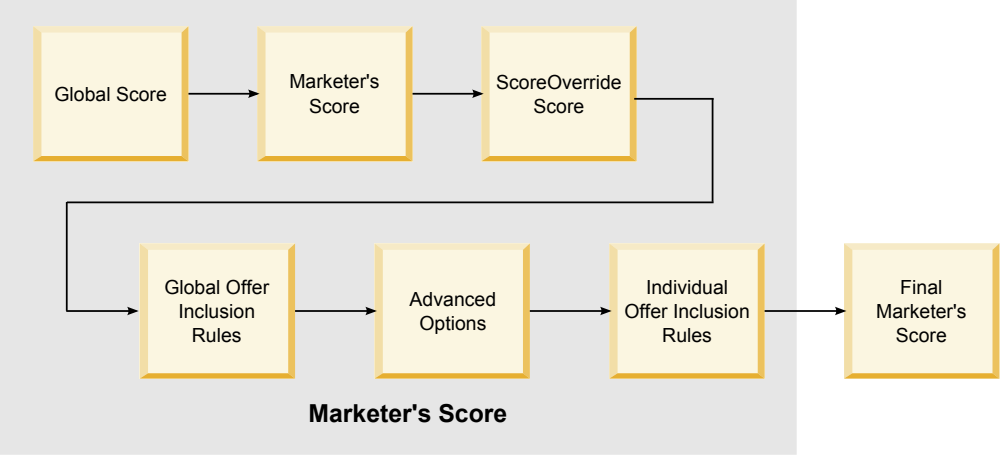


- **Назначения глобальных предложений** - Можно задать глобальные предложения для уровня аудитории при помощи таблицы глобальных предложений.
- **Правила процедур** - Основной метод задания предложений по сегментам и по точкам взаимодействия с помощью вкладки Стратегия взаимодействия.
- **Назначения отдельных предложений** - Можно задать назначения отдельных предложений для покупателей при помощи таблицы переопределения оценок.
- **Дата окончания действия предложения** - При создании предложения в Campaign можно задать дату окончания действия. Если срок действия предложения истек, среда выполнения исключит это предложение.
- **Дата начала и окончания кампании** - При создании кампании в Campaign можно задать даты ее начала и окончания. Если дата начала кампании еще не наступила или же дата окончания уже миновала, среда выполнения исключит это предложение.
- **Подавление предложений** - Можно задать подавление предложений для определенных участников аудитории при помощи таблицы подавления предложений.
- **Максимальное число представлений предложения** - При определении интерактивного канала вы задаете максимальное число раз на сеанс, когда предложение может быть представлено покупателю. Если предложение уже было представлено указанное число раз, среда выполнения исключит это предложение.
- **Правила включения глобальных предложений** - Можно задать логическое выражение для фильтрации предложений на уровне аудитории при помощи таблицы глобальных предложений. Если его результат - false, среда выполнения исключит это предложение.
- **Дополнительные опции** - В правиле процедуры для фильтрации предложений на уровне сегментов можно использовать дополнительную опцию **Считать это правило соответствующим требованиям, если следующее выражение разрешается как true**. Если его результат - false, среда выполнения исключит это предложение.
- **Правила включения отдельных предложений** - Можно задать логическое выражение для фильтрации предложений на уровне отдельных покупателей при помощи таблицы переопределения оценок. Если его результат - false, среда выполнения исключит это предложение.

Вычисление маркетинговой оценки

Существует много способов воздействовать на маркетинговую оценку (при помощи вычислений) или переопределять ее.

На этой диаграмме показаны разные этапы, на которых можно воздействовать на маркетинговую оценку или переопределять ее.



Стрелки показывают порядок очередности. Например, если вы задали на панели Дополнительные опции выражение, определяющее маркетинговую оценку для правила процедуры, и задали выражение в таблице переопределения оценок, то выражение в таблице переопределения оценок будет иметь преимущество.

- **Глобальная оценка** - Можно задать оценку для уровня аудитории при помощи таблицы глобальных предложений.
- **Оценка маркетолога** - Можно задать оценку для сегмента при помощи ползунка в правиле процедуры.
- **Оценка, переопределяющая оценки** - Можно задать оценку для клиента при помощи таблицы переопределения оценок.
- **Правила включения глобального предложения** - Можно задать выражение, вычисляющее оценку для уровня аудитории, при помощи таблицы глобальных предложений.
- **Дополнительные опции** - Можно задать выражение, вычисляющее оценку для сегмента, при помощи дополнительной опции **Использовать следующее выражение в качестве маркетинговой оценки** в правиле процедуры.
- **Правила включения предложения с переопределением оценки** - Можно задать выражение, вычисляющее оценку для клиента, при помощи таблицы переопределения оценок.

Влияние на обучение

Если используется встроенный модуль обучения Interact, вы можете повлиять на результат обучения вне стандартной конфигурации обучения, как, например, список атрибутов обучения или доверительный уровень. При работе с остальными компонентами можно переопределить компоненты алгоритма обучения.

Вы можете переопределить обучение при помощи столбцов `LikelihoodScore` и `AdjExploreScore` предложений по умолчанию и таблиц переопределения оценок. Можно добавить эти столбцы в предложения по умолчанию и таблицы переопределения оценок при помощи сценария возможности `aci_scoringfeature`. Для надлежащего использования этих переопределений надо хорошо разбираться во встроенном обучении Interact.

Модуль обучения оперирует списком предложений-кандидатов с маркетинговыми оценками для каждого кандидата и на этой основе производит окончательные расчеты. Список предложений с атрибутами обучения используется для расчета вероятности того, что покупатель примет предложение. Используя эти вероятности и

хронологию числа представлений для баланса между исследованием и применением, алгоритм обучения определяет вес предложения. Наконец, встроенный механизм обучения умножает вес предложения на конечную маркетинговую оценку и возвращает окончательное значение. Предложения сортируются по этой окончательной оценке.

Подавление предложения

Можно сконфигурировать среду выполнения для подавления предложений.

Существует несколько способов подавления предложений средой выполнения:

- Элемент интерактивного канала **Сколько раз (максимально) следует представить любое предложение в ходе одного посещения**.

  Элемент **Сколько раз (максимально) следует представить любое предложение в ходе одного посещения** задается вами при создании или редактировании интерактивного канала.
- Использование таблицы подавления предложений.

  Вы создаете таблицу подавления предложений в своей базе данных профиля.
- Предложения с истекшим сроком действия.
- Предложения из завершившихся кампаний.
- Предложения, исключенные из-за несоответствия правилам включения (расширенная настройка правил процедур).
- Предложения, уже явно принятые или отклоненные в сеансе Interact. Если клиент явным образом принял или отклонил предложение, это предложение будет подавлено во время сеанса.

Включение подавления предложений

Используйте следующую процедуру для включения подавления предложений.

Об этой задаче

Можно сконфигурировать Interact, задав ссылку на список подавленных предложений.

Процедура

1. Создайте `offerSuppressionTable`, новую таблицу для любой аудитории, содержащую ID аудитории и ID предложения.
2. Задайте для свойства `enableOfferSuppressionLookup` значение **true**.
3. Задайте для свойства `Interact > profile > offerSuppressionTable` в качестве значение имя таблицы подавления предложений для соответствующей аудитории.

Таблица подавления предложений

Таблица подавления предложений позволяет подавлять предложения для конкретного ID аудитории. Например, если ваш уровень аудитории - Покупатель, можно подавить предложение для покупателя Джона Смита. В вашей производственной базе данных профилей должна быть версия этой таблицы хотя бы для одного уровня аудитории. Образец таблицы подавления предложения `UACI_Blacklist` можно создать, запустив сценарий SQL `aci_usrtab` для вашей базы данных профилей. Сценарий SQL `aci_usrtab` расположен в каталоге `ddl` вашего каталога установки среды выполнения.

Для каждой строки надо определить поля `AudienceID` и `OfferCode1`. Если ваш ID аудитории или код предложения содержит несколько столбцов, можно добавить

столбцы. Эти столбцы должны соответствовать именам столбцов, заданным в Campaign. Например, если вы определяете аудиторию Покупатель с помощью полей HHold_ID и MemberNum, надо добавить HHold_ID и MemberNum в таблицу подавления предложений.

Имя	Описание
AudienceID	(обязательный) Имя этого столбца должно соответствовать имени столбца, определяющего ID аудитории в Campaign. Если ваш ID аудитории состоит из нескольких столбцов, можно добавить их к этой таблице. Каждая строка должна содержать ID аудитории, которой вы назначаете предложение по умолчанию, например, customer1.
OfferCode1	(обязательный) Код предложения, которое вы переопределяете. Если ваши коды предложений состоят из нескольких полей, можно добавить дополнительные столбцы, например OfferCode2 и т.д.

Глобальные предложения и индивидуальные назначения

Можно сконфигурировать среду выполнения для назначения определенных предложений вне правил процедур, сконфигурированных на вкладке Стратегия взаимодействия. Можно определить глобальные предложения для любого участника уровня аудитории и индивидуальные назначения для определенных участников аудитории. Например, можно задать глобальное предложение для всех семей, рассматриваемое при отсутствии других предложений, а затем создать индивидуальное назначение предложения для конкретной семьи Смитов.

И глобальные предложения, и индивидуальные назначения можно ограничить зоной, ячейкой и правилами включения предложений. И глобальные предложения, и индивидуальные назначения конфигурируются путем добавления данных в специальные таблицы в вашей производственной базе данных профилей.

Для надлежащего функционирования глобальных предложений и индивидуальных назначений все упомянутые в ссылках ячейки и коды предложений должны существовать во внедрении. Чтобы обеспечить доступность требуемых данных, надо сконфигурировать коды ячеек по умолчанию и таблицу UACI_ICBatchOffers.

Задание кодов ячеек по умолчанию

Если используются предложения по умолчанию или таблицы переопределения оценок для глобальных или индивидуальных назначений предложений, необходимо задать коды ячеек по умолчанию. DefaultCellCode используется, если не заданы коды ячеек в конкретных строках предложений по умолчанию или таблиц переопределения оценок. Этот код ячейки по умолчанию используется при создании отчетов.

Об этой задаче

DefaultCellCode должен соответствовать формату кода ячейки, заданному в Campaign. Этот код ячейки используется в отчетах для всех назначений предложений.

Если вы задали уникальные коды ячеек по умолчанию, можно легко идентифицировать предложения, назначенные предложениями по умолчанию или таблицами переопределения оценок.

Процедура

Задайте свойство DefaultCellCode для каждого уровня аудитории и типа таблицы в категории IndividualTreatment.

Определение предложений, не используемых в правилах процедур

Если используются предложения по умолчанию или таблицы переопределения оценок, нужно обеспечить, чтобы во внедрении присутствовали все коды предложений. Если вам известно, что все предложения, используемые вами в предложениях по умолчанию или таблицах переопределения оценок, используются в ваших правилах процедур, то эти предложения существуют во внедрении. Однако любое предложение, которое не используется в правилах процедур, необходимо задать в таблице `UACI_ICBatchOffers`.

Об этой задаче

Таблица `UACI_ICBatchOffers` относится к системным таблицам Campaign.

Процедура

Заполните таблицу `UACI_ICBatchOffers` кодами предложений, которые используются в предложении по умолчанию или таблицах переопределения оценок. Формат этой таблицы:

Имя столбца +	Тип	Описание
ICName	varchar(64)	Имя интерактивного канала, с которым связано предложение. Если одно предложение используется с двумя различными интерактивными каналами, надо указать строку для каждого канала.
OfferCode1	varchar(64)	Первая часть кода предложения.
OfferCode2	varchar(64)	Вторая часть кода предложения.
OfferCode3	varchar(64)	Третья часть кода предложения.
OfferCode4	varchar(64)	Четвертая часть кода предложения.
OfferCode5	varchar(64)	Пятая часть кода предложения.

О таблице глобальных предложений

Таблица глобальных предложений позволяет определять процедуры на уровне аудитории. Например, можно задать глобальное предложение для каждого участника аудитории Семья.

Можно задать глобальные параметры для следующих элементов представления предложений Interact.

- Глобальное назначение предложений
- Глобальная оценка маркетолога, по числу или по выражению
- Логическое выражение для фильтрации предложений
- Вероятность и вес обучения, если используется встроенное обучение Interact
- Переопределение глобального обучения

Назначение глобальных предложений

С помощью этой процедуры можно сконфигурировать среду выполнения для назначения глобальных предложений для уровня аудитории в обход параметров, заданных в правилах процедур.

Процедура

1. Создайте таблицу с именем UACI_DefaultOffers в вашей базе данных профилей.

   Чтобы создать таблицу UACI_DefaultOffers с правильными столбцами, воспользуйтесь файлом DDL aci_usrtab.
2. Задайте для свойства Interact > profile > enableDefaultOfferLookup значение **true**.

Глобальная таблица предложений

Глобальная таблица предложений должна существовать в вашей базе данных профилей. Можно создать глобальную таблицу предложений UACI_DefaultOffers, запустив сценарий SQL aci_usrtab для вашей базы данных профилей.

Сценарий SQL aci_usrtab расположен в каталоге ddl вашего каталога установки среды выполнения.

Для каждой строки надо определить поля AudienceLevel и OfferCode1. Остальные поля не обязательны; они накладывают дальнейшие ограничения на назначение предложений или влияют на встроенное обучение на уровне аудитории.

Для оптимальной производительности надо создать индекс для этой таблицы по столбцу уровня аудитории.

Имя	Тип	Описание
AudienceLevel	varchar(64)	(обязательный) Имя уровня аудитории, которое вы назначили для предложения по умолчанию, например, покупатель или семья. Это имя должно соответствовать уровню аудитории, определенному в Campaign.
OfferCode1	varchar(64)	(обязательный) Код предложения для предложения по умолчанию. Если ваши коды предложений состоят из нескольких полей, можно добавить дополнительные столбцы, например OfferCode2 и т.д. Если вы добавляете это предложение, чтобы обеспечить глобальное назначение предложения, надо добавить это предложение в таблицу UACI_ICBatchOffers.
Score	float	Число, определяющее маркетинговую оценку для этого назначения предложения.
OverrideTypeID	int	Если задано значение 1 и предложения нет в списке предложений-кандидатов, добавить это предложение в список и использовать данные оценок для этого предложения. В целом, используйте 1, чтобы обеспечить глобальные назначения предложения. Если задано 0, *null* или любое отличное от 1 число, использовать любые данные для этого предложения, только если это предложение есть в списке предложений-кандидатов. В большинстве случаев правило процедуры или отдельное присвоение переопределяют этот параметр.

Имя	Тип	Описание
Predicate	varchar(4000)	Можно ввести выражения в этот столбец для дополнительных опций правил процедур. Можно использовать те же переменные и макрокоманды, которые доступны вам при записи дополнительных опций для правил процедур. Поведение этого столбца зависит от значения в столбце `EnableStateID`. • Если для `EnableStateID` задано 2, этот столбец работает так же, как опция **Считать это правило пригодным, если следующее выражение оценивается как true** в дополнительных опциях для правил процедур, чтобы ограничить назначение этого предложения. Этот столбец должен содержать логическое выражение и разрешаться как true, чтобы предложение было включено. Если вы случайно определили выражение, которое разрешается как число, любое ненулевое число рассматривается как true, а ноль - как false. • Если для `EnableStateID` задано 3, этот столбец работает так же, как опция **Использовать следующее выражение как маркетинговую оценку** в дополнительных опциях для правил процедур, чтобы ограничить назначение этого предложения. Этот столбец должен содержать выражение, которое разрешается как число. • Если для `EnableStateID` задано значение 1, Interact игнорирует любое значение в этом столбце.
FinalScore	float	Число для переопределения итоговой оценки, используется для заказа итогового списка возвращаемых предложений. Этот столбец используется, если вы включили встроенный модуль обучения. Можно реализовать ваш собственный алгоритм обучения, чтобы он использовал этот столбец.
CellCode	varchar(64)	Код ячейки для интерактивного сегмента, которому вы хотите назначить это предложение по умолчанию. Если ваши коды ячеек состоят из нескольких полей, можно добавить дополнительные столбцы. Вы должны задать код ячейки, если для `OverrideTypeID` задано значение 0 или null. Если вы не включаете код ячейки, среда выполнения игнорирует эту строку данных. Если для `OverrideTypeID` задано значение 1, нет необходимости задавать код ячейки в этом столбце. Если вы не задали код ячейки, среда выполнения использует код ячейки, определенный в свойстве `DefaultCellCode` для этого уровня аудитории и таблицы составления отчетов.
Zone	varchar(64)	Имя зоны, к которой вы хотите применить это назначение предложения. Если задано NULL, оно применяется ко всем зонам.

Имя	Тип	Описание
EnableStateID	int	Значение в этом столбце определяет поведение столбца `Predicate`. • **1** - Не использовать столбец `Predicate`. • **2** - Использовать столбец `Predicate` как логическое значение для фильтрации предложения. При этом используются те же правила, что и для дополнительной опции **Считать это правило пригодным, если следующее выражение оценивается как true** в правиле процедур. • **3** - Использовать `Predicate` для определения оценки маркетолога. При этом используются те же правила, что и для дополнительной опции **Использовать следующее выражение в качестве маркетинговой оценки** в правиле процедур. Любая строка, в которой в этом столбце задан Null или любое значение кроме 2 или 3, игнорирует столбец `Predicate`.
LikelihoodScore	float	Этот столбец используется только для влияния на встроенное обучение. Можно добавить этот столбец с помощью DDL `aci_scoringfeature`.
AdjExploreScore	float	Этот столбец используется только для влияния на встроенное обучение. Можно добавить этот столбец с помощью DDL `aci_scoringfeature`.

О таблице переопределения оценок

Таблица переопределения оценок позволяет задавать процедуры на уровне ID аудитории или на индивидуальном уровне. Например, если ваш уровень аудитории - Посетитель, можно создать переопределения для конкретных посетителей.

Можно задать переопределения для следующих элементов представления предложений Interact.

- Назначение отдельного предложения
- Индивидуальная оценка маркетолога в виде числа или выражения
- Логическое выражение для фильтрации предложений
- Вероятность обучения и вес, если используется встроенное обучение
- Переопределение индивидуального обучения

Конфигурирование переопределения оценок

Можно сконфигурировать Interact для работы с оценкой, сгенерированной прикладной программой моделирования, вместо маркетинговой оценки.

Процедура

1. Создайте таблицу переопределения оценок для каждого уровня аудитории, для которого хотите выполнить переопределение.

   Чтобы создать пример таблицы переопределения оценок с правильными столбцами, воспользуйтесь файлом DDL `aci_usrtab`.
2. Задайте для свойства `Interact > Profile > enableScoreOverrideLookup` значение **true**.

3. Задайте для свойства `scoreOverrideTable` значение, равное имени таблицы переопределения оценок для каждого уровня аудитории, для которого хотите выполнить переопределение.

   Предоставлять таблицу переопределения оценок для каждого уровня аудитории нет необходимости.

Таблица переопределения оценок

Таблица переопределения оценок должна существовать в вашей производственной базе данных профилей. Образец таблицы переопределения оценок `UACI_ScoreOverride` можно создать, запустив сценарий SQL `aci_usrtab` для вашей базы данных профилей.

Сценарий SQL `aci_usrtab` расположен в каталоге `ddl` вашего каталога установки среды выполнения.

Для каждой строки надо определить поля *`AudienceID`*, `OfferCode1` и `Score`. Значения в других полях не обязательны и служат для дальнейшего ограничения ваших назначений отдельных предложения или предоставления информацию переопределения оценок для встроенного обучения.

Имя	Тип	Описание
AudienceID	varchar(64)	(обязательный) Имя этого столбца должно соответствовать имени столбца, определяющего ID аудитории в Campaign. Таблица примера, созданная файлом DDL `aci_usrtab`, создает этот столбец как столбец `CustomerID`. Если ваш ID аудитории состоит из нескольких столбцов, можно добавить их к этой таблице. Каждая строка должна содержать ID аудитории, которой вы назначаете отдельное предложение, например, `customer1`. Для оптимальной производительности надо создать индекс по этому столбцу.
OfferCode1	varchar(64)	(обязательный) Код предложения. Если ваши коды предложений состоят из нескольких полей, можно добавить дополнительные столбцы, например `OfferCode2` и т.д. Если вы добавляете это предложение, чтобы обеспечить отдельное назначение предложения, надо добавить это предложение в таблицу `UACI_ICBatchOffers`.
Оценка	float	Число, определяющее маркетинговую оценку для этого назначения предложения.
OverrideTypeID	int	Если задано `0` или *null* (или любое отличное от 1 число), использовать любые данные для этого предложения, только если это предложение есть в списке предложений-кандидатов. В целом, используйте `0`, чтобы обеспечить переопределение оценок. Надо задать код ячейки Если задано значение `1` и предложения нет в списке предложений-кандидатов, добавить это предложение в список и использовать данные оценок для этого предложения. В целом, используйте `1`, чтобы обеспечить отдельные назначения предложения.

Имя	Тип	Описание
Предикат	varchar(4000)	Можно ввести выражения в этот столбец для дополнительных опций правил процедур. Можно использовать те же переменные и макрокоманды, которые доступны вам при записи дополнительных опций для правил процедур. Поведение этого столбца зависит от значения в столбце `EnableStateID`. • Если для `EnableStateID` задано `2`, этот столбец работает так же, как опция **Считать это правило пригодным, если следующее выражение оценивается как true** в дополнительных опциях для правил процедур, чтобы ограничить назначение этого предложения. Этот столбец должен содержать логическое выражение и разрешаться как true, чтобы предложение было включено. Если вы случайно определили выражение, которое разрешается как число, любое ненулевое число рассматривается как true, а ноль - как false. • Если для `EnableStateID` задано `3`, этот столбец работает так же, как опция **Использовать следующее выражение как маркетинговую оценку** в дополнительных опциях для правил процедур, чтобы ограничить назначение этого предложения. Этот столбец должен содержать выражение, которое разрешается как число. • Если для `EnableStateID` задано значение `1`, Interact игнорирует любое значение в этом столбце.
FinalScore	float	Число для переопределения итоговой оценки, используется для заказа итогового списка возвращаемых предложений. Этот столбец используется, если вы включили встроенный модуль обучения. Можно реализовать ваш собственный алгоритм обучения, чтобы он использовал этот столбец.
CellCode	varchar(64)	Код ячейки для интерактивного сегмента, которому вы хотите назначить это предложение. Если ваши коды ячеек состоят из нескольких полей, можно добавить дополнительные столбцы. Вы должны задать код ячейки, если для `OverrideTypeID` задано значение 0 или null. Если вы не включаете код ячейки, среда выполнения игнорирует эту строку данных. Если для `OverrideTypeID` задано значение 1, нет необходимости задавать код ячейки в этом столбце. Если вы не задали код ячейки, среда выполнения использует код ячейки, определенный в свойстве `DefaultCellCode` для этого уровня аудитории и таблицы составления отчетов.
Зона	varchar(64)	Имя зоны, к которой вы хотите применить это назначение предложения. Если NULL, это относится ко всем зонам.

Имя	Тип	Описание
EnableStateID	int	Значение в этом столбце определяет поведение столбца `Predicate`. • **1** - Не использовать столбец `Predicate`. • **2** - Использовать столбец `Predicate` как логическое значение для фильтрации предложения. При этом используются те же правила, что и для дополнительной опции **Считать это правило пригодным, если следующее выражение оценивается как true** в правиле процедур. • **3** - Использовать `Predicate` для определения оценки маркетолога. При этом используются те же правила, что и для дополнительной опции **Использовать следующее выражение в качестве маркетинговой оценки** в правиле процедур. Любая строка, в которой в этом столбце задан Null или любое значение кроме 2 или 3, игнорирует столбец `Predicate`.
LikelihoodScore	float	Этот столбец используется только для влияния на встроенное изучение. Можно добавить этот столбец с помощью DDL `aci_scoringfeature`.
AdjExploreScore	float	Этот столбец используется только для влияния на встроенное изучение. Можно добавить этот столбец с помощью DDL `aci_scoringfeature`.

Обзор встроенного обучения Interact

Хотя в целом вы делаете все возможное, чтобы обеспечить представление нужных предложений в нужные сегменты, вы всегда можете почерпнуть что-то полезное, изучая фактический выбор ваших посетителей. Ваша стратегия не может не зависеть от реального поведения посетителей. Вы можете посмотреть хронологию ответов и пропустить ее через инструменты моделирования, получив в результате оценки, которые можно включить в ваши интерактивные потоковые диаграммы.

Однако это не данные реального времени.

В Interact вам предлагаются две опции обучения на основе действий посетителей в реальном времени:

- Встроенный модуль обучения - среда выполнения содержит модуль обучения на основе наивного байесовского подхода. Этот модуль содержит выбранные вами атрибуты клиентов и с помощью этих данных отбирает предложения для представления.
- API обучения - В среде выполнения есть также API обучения, позволяющий вам написать свой собственный модуль обучения.

Использовать обучение необязательно. По умолчанию обучение отключено.

Модуль обучения Interact

Модуль обучения Interact отслеживает ответы посетителя на предложения и атрибуты посетителя.

Режимы модуля обучения

У модуля обучения есть два основных режима:

- Исследование - модуль обучения представляет предложения, чтобы собрать достаточно данных ответов для оптимизации оценки, используемой в режиме применения. Предложения, представленные в режиме исследования, не обязательно отражают оптимальный выбор.
- Применение - после того, как на стадии исследования собрано достаточно данных, модуль обучения оценивает вероятности успеха для предложений, помогая выбрать предложения для представления.

Для переключения между режимом исследования и режимом применения модуль обучения использует два свойства. Это следующие два свойства:

- доверительный уровень, конфигурируемый при помощи свойства `confidenceLevel`.
- вероятность, что модуль обучения представляет случайное предложение, конфигурируемая при помощи свойства `percentRandomSelection`.

Свойство доверительного уровня

Вы задаете для `confidenceLevel` значение в виде процентов, показывающее, насколько уверенным должен быть модуль обучения в своих оценках предложений, чтобы эти оценки можно было использовать при принятии решений. Вначале, когда обучающий модуль не располагает данными для работы, он основывается целиком на маркетинговой оценке. После того, как каждое предложение представлено столько раз, сколько задано значением `minPresentCountThreshold`, модуль обучения перейдет в режим исследования. При небольшом объеме обрабатываемых данных модуль обучения не может гарантировать правильность вычисленных им процентных значений. Поэтому он остается в режиме исследования.

Модуль обучения назначает вес каждому предложению. Для расчета весов он использует формулу, в которую подставляются сконфигурированный доверительный уровень, хронология принятия данных и данные текущего сеанса. Эта формула предполагает баланс между исследованием и применением и возвращает соответствующий вес.

Свойство случайного выбора

Чтобы устранить возможное предпочтение системой предложений, получивших наилучшие оценки на ранних этапах отбора, Interact представляет случайное предложение в течение `percentRandomSelection` процентов всего времени. В течение времени, определяемого этим процентом случайных предложений, модуль обучения вынужден рекомендовать другие, менее успешные предложения в расчете на то, что при большем времени оценки эти предложения получат преимущество. Например, если задать для `percentRandomSelection` значение 5, модуль обучения будет представлять в течение 5% всего времени случайно выбранные предложения и добавит в свои расчеты данные ответов.

Можно задать **% случайных**, вероятность случайной выборки возвращаемого предложения без учета оценок, для каждой зоны на вкладке Точки взаимодействия в окне Интерактивный канал.

Как модуль обучения определяет предложения

Модуль обучения определяет, какие предложения представляются, следующим образом.

1. Вычисляет вероятность выбора предложения посетителем.
2. Вычисляет вес предложения, используя вероятность на шаге 1, и решает, работать ли в режиме исследования или применения.
3. Вычисляет окончательную оценку для каждого предложения, используя маркетинговую оценку и вес предложения из шага 2.
4. Сортирует предложения по значениям оценки, полученным на шаге 3, и возвращает требуемое число наиболее подходящих предложений.

Пусть, например, модуль обучения определил, что посетитель с вероятность 30% примет предложение A и с вероятностью 70% примет предложение B и собирается использовать эту информацию. В правилах процедур указано, что маркетинговая оценка для предложения A - 75, а для предложения B - 55. Однако после расчетов на шаге 3 окончательная оценка для предложения B получается выше, чем для предложения A, поэтому среда выполнения рекомендует предложение B.

Свойства весовых коэффициентов

Обучение основано также на свойствах `recencyWeightingFactor` и `recencyWeightingPeriod`. Эти свойства позволяют присвоить больший вес более свежим данным по сравнению с более старыми. `recencyWeightingFactor` - это весовая доля, присваиваемая свежим данным. `recencyWeightingPeriod` - это период времени, в течение которого данные считаются свежими. Например, вы задали для `recencyWeightingFactor` значение 0.30, а для `recencyWeightingPeriod` - значение 24. Эти параметры означают, что данные за последние 24 часа будут составлять 30% от всех рассматриваемых данных. Для данных за неделю совокупность данных, обработанных за первые шесть дней, составит 70% всех данных, а за последние сутки - 30% всех данных.

Запись данных в промежуточную таблицу

Каждый сеанс записывает в промежуточную таблицу обучения следующие данные:

- Контакт предложения
- Принятие предложения
- Атрибуты обучения

В конфигурируемом интервале агрегатор считывает данные из промежуточной таблицы, компилирует их и записывает в таблицу. Модуль обучения считывает эти агрегированные данные и использует их в вычислениях.

Включение модуля обучения

На всех серверах среды выполнения есть встроенный модуль обучения. По умолчанию модуль обучения отключен. Для включения модуля обучения нужно изменить свойства конфигурации.

Процедура

В Marketing Platform для среды выполнения отредактируйте следующие свойства конфигурации в категории `Interact > offerserving`.

Свойство конфигурации	Значение
`optimizationType`	**BuiltInLearning**

Атрибуты обучения

Модуль обучения работает с использованием атрибутов посетителя и данных о принятии предложения. Вы можете выбрать, какие из атрибутов посетителей отслеживать. Эти атрибуты посетителя могут быть элементами профиля покупателя, включая параметры событий, собранные вами в режиме реального времени.

Атрибуты из таблиц измерений при обучении не поддерживаются.

Хотя можно сконфигурировать любое число атрибутов для отслеживания, IBM рекомендует конфигурировать не более десяти атрибутов обучения (между статическими и динамическими), а также соблюдать следующие правила.

- Выбирайте независимые атрибуты.

  Не следует выбирать похожие атрибуты. Например, если вы создали атрибут HighValue и этот атрибут определяется расчетами, основанными на заработной плате, не выбирайте одновременно HighValue и Salary. Похожие атрибуты бесполезны для алгоритма обучения.
- Выбирайте атрибуты с дискретными значениями.

  Если у атрибута есть диапазон значений, надо выбрать точное значение. Например, если нужно в качестве атрибута использовать заработную плату, надо присвоить отдельное значение каждому диапазону зарплаты: для диапазона 20000-30000 это будет A, для диапазона 30001-40000 - B и так далее.
- Ограничьте число отслеживаемых атрибутов, чтобы не ухудшать производительность.

  Число атрибутов, которые вы в состоянии отслеживать, зависит от ваших требований производительности и вашей установки Interact. Если возможно, используйте другой инструмент моделирования (например, PredictiveInsight), чтобы определить первые десять предиктивных атрибутов. Можно сконфигурировать модуль обучения, чтобы он автоматически отсекал атрибуты с малой предиктивностью, но при этом приводящие к снижению производительности.

Можно управлять производительностью, задав одновременно число отслеживаемых вами атрибутов и число значений на отслеживаемый атрибут. Свойство `Campaign > partitions > partition1 > Interact > learning > maxAttributeNames` задает максимальное число отслеживаемых вами атрибутов посетителя. Свойство `maxAttributeValues` задает максимальное число значений, отслеживаемых вами на атрибут. Все прочие значения назначены категории, определяемой значением свойства `otherAttributeValue`. Однако механизм обучения отслеживает только первые встреченные им значения. Например, вы отслеживаете атрибут посетителя - цвет глаз. Вас интересуют только значения голубые, карие и зеленые, поэтому вы задали для `maxAttributeValues` значение 3. Однако у первых трех посетителей эти значения - голубые, карие и серые. Это означает, что всем посетителям с зелеными глазами будет назначено свойство `otherAttributeValue`.

Можно также воспользоваться атрибутами динамического обучения, позволяющими конкретизировать ваши критерии обучения. Атрибуты динамического обучения дают возможность использовать в обучении комбинацию двух атрибутов как единую запись. Рассмотрим в качестве примера следующую информации о профиле.

ID посетителя	Тип карты	Баланс карты
1	Золотая карта	1000 долларов США
2	Золотая карта	9000 долларов США
3	Бронзовая карта	1000 долларов США

ID посетителя	Тип карты	Баланс карты
4	Бронзовая карта	9000 долларов США

При использовании стандартных атрибутов обучения можно проводить обучение только индивидуально, по типу карты или балансу. Посетители 1 и 2 будут объединены в одну группу на основе типа карты, а посетители 2 и 4 - на основе баланса карты. Этот предиктор принятия предложений может оказаться неточным. Если обладатели Золотой карты обычно имеют более высокий баланс, поведение Посетителя 2 может кардинально отличаться от поведения Посетителя 4, что приведет к перекосу для стандартных атрибутов обучения. Однако если используются атрибуты динамического обучения, каждый из этих посетителей будет рассматриваться при обучении отдельно, и прогноз окажется более точным.

Если используются атрибуты динамического обучения и у посетителя есть два допустимых значения для атрибута, модуль обучения выберет первое из найденных им значений.

Если для свойства `enablePruning` задано значение `yes`, обучающий модуль алгоритмически определит, какие из атрибутов не обладают предиктивностью, и перестанет учитывать их при расчете весов. Например, если вы отслеживаете атрибут, представляющий цвет волос, и обучающий модуль определит, что паттерн для принятия предложения на основе цвета волос отсутствует, модуль обучения прекратит учитывать атрибут цвет волос. Атрибуты подвергаются переоценке при каждом запуске процесса агрегирования обучения (задается свойством `aggregateStatsIntervalInMinutes`). Динамические атрибуты обучения также усекаются.

Определение атрибута обучения

Атрибут обучения задается при помощи следующей процедуры.

Об этой задаче

Можно сконфигурировать до `maxAttributeNames` атрибутов посетителя.

`(learningAttributes)` служит шаблоном для создания новых атрибутов обучения. Нужно ввести новое имя для каждого атрибута. Нельзя создать две категории с одинаковыми именами

Процедура

В Marketing Platform для среды разработки отредактируйте следующие свойства конфигурации из категории `Campaign > partitions > partition`*`n`* `> Interact > learning`.

Свойство конфигурации	Значение
`attributeName`	`attributeName` должно соответствовать имени пары имя - значение в данных профиля. Регистр символов в этом имени не учитывается.

Определение атрибутов динамического обучения

Чтобы определить атрибуты динамического обучения, надо заполнить таблицу `UACI_AttributeList` в источнике данных обучения.

Все столбцы в этой таблице имеют тип varchar(64).

Вертикальная столбчатая диаграмма	Описание
AttributeName	Имя динамического атрибута, на котором должно быть основано обучение. Это значение должно быть фактическим значением, допустимым в AttributeNameCol.
AttributeNameCol	Полное имя столбца (иерархическая структура, начиная от таблицы профилей), где находится AttributeName. Имя этого столбца не обязательно должно быть стандартным атрибутом обучения.
AttributeValueCol	Полное имя столбца (иерархическая структура, начиная от таблицы профилей), где находится связанное значение для AttributeName.

Например, рассмотрим следующую таблицу профилей и связанную с ней таблицу измерений.

Таблица 6. MyProfileTable

VisitorID	KeyField
1	Ключ1
2	Ключ2
3	Ключ3
4	Ключ4

Таблица 7. MyDimensionTable

KeyField	CardType	CardBalance
Ключ1	Золотая карта	1000
Ключ2	Золотая карта	9000
Ключ3	Бронзовая карта	1000
Ключ4	Бронзовая карта	9000

Ниже приведен пример таблицы UACI_AttributeList для совпадений по типу карты и балансу.

Таблица 8. UACI_AttributeList

AttributeName	AttributeNameCol	AttributeValueCol
Золотая карта	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance
Бронзовая карта	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance

Конфигурирование среды выполнения для распознавания внешних модулей обучения

API обучения Java™ можно использовать для написания вашего собственного модуля обучения. Среду выполнения надо сконфигурировать для распознавания вашей утилиты обучения в Marketing Platform.

Об этой задаче

Чтобы изменения вступили в силу, нужно перезапустить сервер среды выполнения Interact.

Процедура

1. В Marketing Platform для среды выполнения отредактируйте следующие свойства конфигурации в категории `Interact > offerserving`. Свойства конфигурации для API оптимизатора обучения содержатся в категории `Interact > offerserving > External Learning Config`.

Свойство конфигурации	Значение
`optimizationType`	**ExternalLearning**
`externalLearningClass`	имя класса для внешнего обучения
`externalLearningClassPath`	Путь к классу или файлам JAR на сервере среды выполнения для внешнего обучения. Если используется группа серверов и все серверы среды выполнения ссылаются на один и тот же экземпляр Marketing Platform, на каждом сервере должна быть копия этого класса или файлы JAR в том же каталоге.

2. Чтобы изменения вступили в силу, перезапустите сервер среды выполнения Interact.

Глава 5. Что такое API Interact

Interact динамически распределяет предложения по разнообразным точкам взаимодействия. Например, можно сконфигурировать среду выполнения и вашу точку контакта для отправки сообщений сотрудникам колл-центра с информацией о наилучших перспективах дополнительных или перекрестных продаж для клиента, обратившегося за конкретной справкой по обслуживанию. Можно также сконфигурировать среду выполнения и вашу точку контакта для предоставления предложений клиенту (посетителю), зашедшему в определенную область вашего сайта, в соответствии с его потребностями.

Интерфейс прикладного программирования (API) Interact позволяет сконфигурировать вашу точку контакта и сервер среды выполнения для совместной работы по представлению посетителям наилучших предложений. При помощи API точка контакта может запрашивать информацию у сервера среды выполнения для назначения посетителя некоторой группе (сегменту) и представления предложений с учетом специфики этого сегмента. Вы можете также заносить данные в журнал для последующего анализа с целью совершенствования стратегии представления предложений.

API Interact также позволяет клиенту конечного пользователя связываться с сервером через JavaScript.

Для обеспечения максимально возможной гибкости при интеграции Interact с вашими средами IBM содержит веб-службу, доступную посредством API Interact.

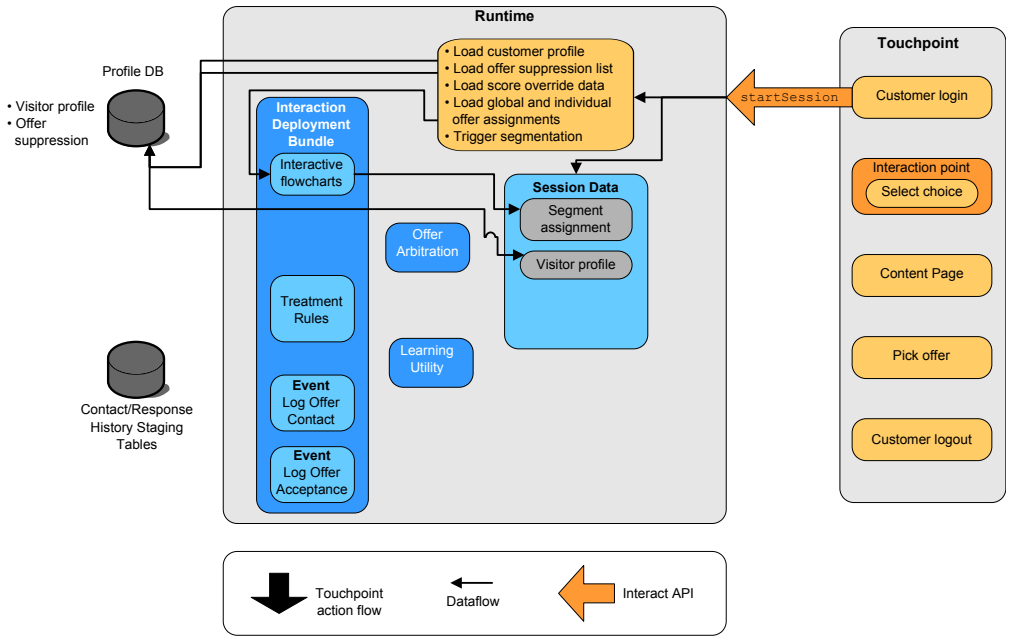
Поток данных API Interact

В этом примере показано, как работает API между вашей точкой контакта и средой выполнения. Посетитель может выполнять только четыре действия - вход в систему, переход на страницу с предложениями, выбор предложения и выход из системы. В рамках ваших требований к производительности можно разработать интеграцию любой нужной вам сложности.

На этой диаграмме показана простая реализация API Interact.

Посетитель входит в систему на сайте и находит страницу с предложениями. Затем он выбирает предложение и выходит из системы. Несмотря на простоту взаимодействия, и в точке контакта, и на сервере среды выполнения происходит несколько событий:

1. Запуск сеанса
2. Переход к странице
3. Выбор предложения
4. Закрытие сеанса



Запуск сеанса

Когда посетитель входит в систему, он инициирует метод `startSession`.

Метод `startSession` выполняет четыре функции:

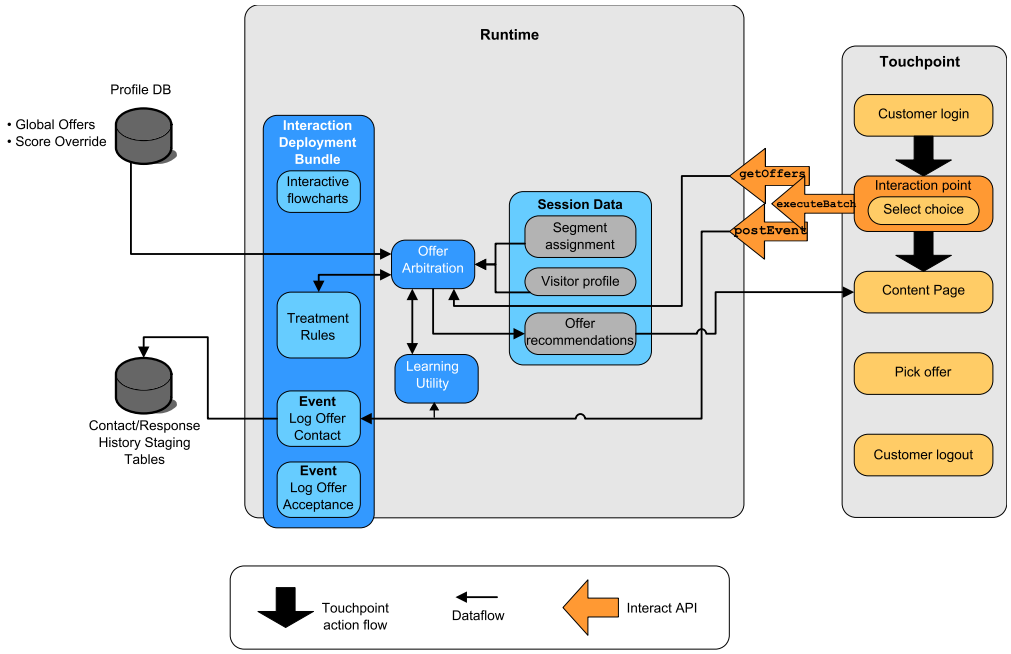
1. Создает новый сеанс среды выполнения
2. Отправляет требование загрузки данных профиля покупателя в сеанс
3. Отправляет требование использования данных профиля и запуска интерактивной потоковой диаграммы для помещения покупателей в определенные сегменты. Запуск этой потоковой диаграммы выполняется асинхронно.
4. Сервер среды выполнения загружает в сеанс информацию о всех подавлениях предложений, а также обработке глобальных и отдельных предложений. В течение сеанса данные сеанса сохраняются в памяти.

Переход к странице

Посетитель будет перемещаться по сайту до тех пор, пока не достигнет заранее заданной точки взаимодействия. На рисунке вторая точка взаимодействия (Выбрать вариант) - это место, в котором посетитель щелкает по ссылке, представляющей набор предложений. Менеджер точек контакта сконфигурировал эту ссылку для запуска метода `executeBatch`, чтобы выбрать предложение.

Выбор предложения

На этой диаграмме показан вызов API, инициирующий метод `executeBatch`.

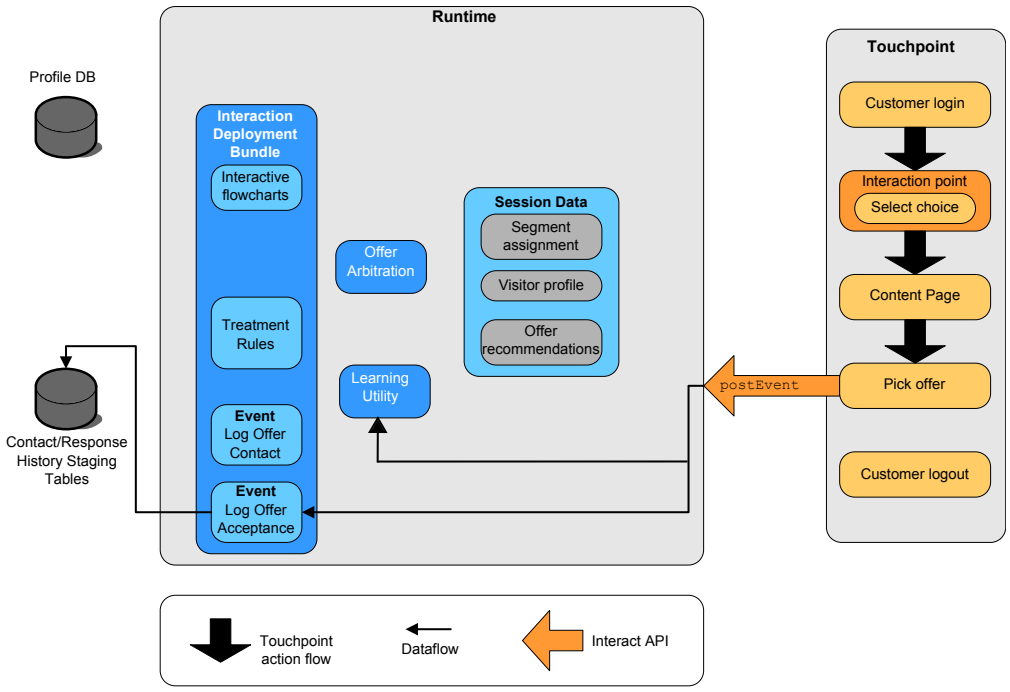


Метод `executeBatch` позволяет вызывать несколько методов в одном вызове сервера среды выполнения. Данный метод executeBatch вызывает два других метода, `getOffers` и `postEvent`. Метод `getOffers` запрашивает список предложений. Сервер среды выполнения использует для представления набора предложений данные сегментации, список подавлений предложений, правила процедур и модуль обучения. Сервер среды выполнения возвращает набор предложений, которые выводятся на странице содержимого.

Метод `postEvent` инициирует одно из событий, определенных в среде разработки. В этом конкретном случае событие посылает требование записи в журнал предложений, представленных в хронологии контактов.

Посетитель выбирает одно из этих предложений (Выбрать предложение).

На этой диаграмме показан метод `postEvent`.

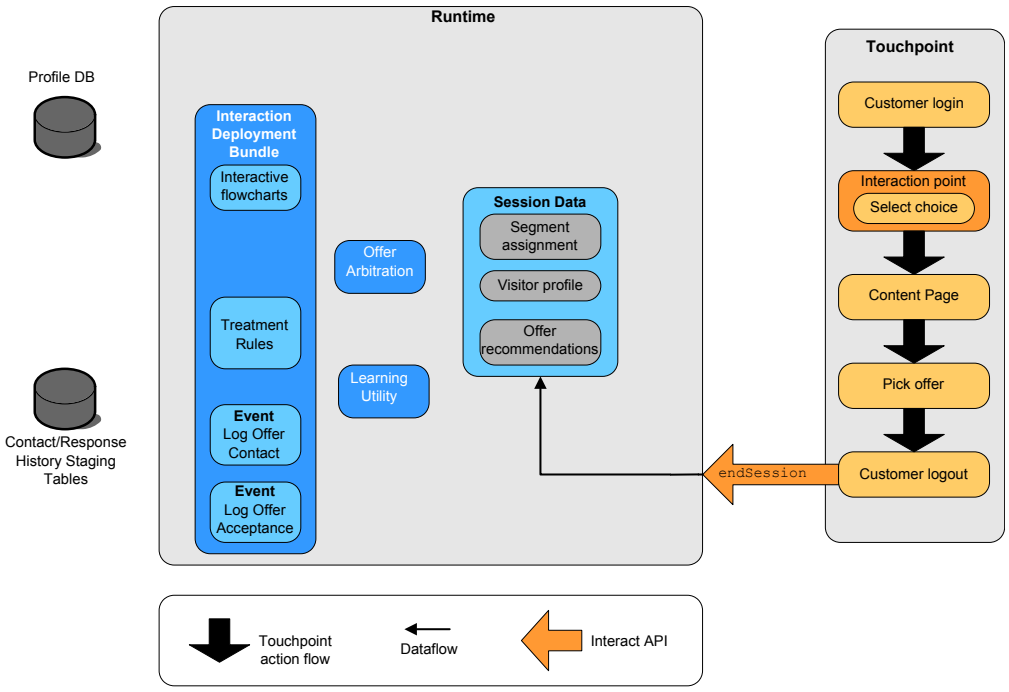


Элемент управления пользовательского интерфейса, связанный с выбором предложения, сконфигурирован для отправки еще одного метода `postEvent`. Это событие посылает требование записи подтверждения, что предложение принято, в хронологию ответов.

Закрытие сеанса

После того, как посетитель выбрал предложение, он заканчивает работу с сайтом и выходит из системы. Команда выхода из системы связана с методом `endSession`.

На этой диаграмме показан метод `endSession`.



Метод `endSession` закрывает сеанс. Если посетитель забыл выйти из сеанса, существует конфигурируемый срок бездействия для сеанса, обеспечивающий закрытие любого сеанса. Если требуется сохранить какие-либо данные, переданные в сеанс, например, информацию, включенную в параметры в методах `startSession` или `setAudience`, обратитесь к сотруднику, создающему интерактивные потоковые диаграммы. Сотрудник, создающий интерактивную потоковую диаграмму, может записать эти данные в базу данных при помощи процесса Снимок до того, как сеанс завершится и данные будут утеряны. Можно затем с помощью метода `postEvent` вызвать интерактивную потоковую диаграмму, содержащую процесс Снимок.

Простой пример планирования взаимодействия

В этом примере вы разрабатываете взаимодействие для веб-сайта оператора сотовой связи. Вы создаете три различных предложения, конфигурируете запись в журнал для этих предложений, назначаете им коды процедур и демонстрируете набор изображений, связанных с предложениями.

Процесс разработки

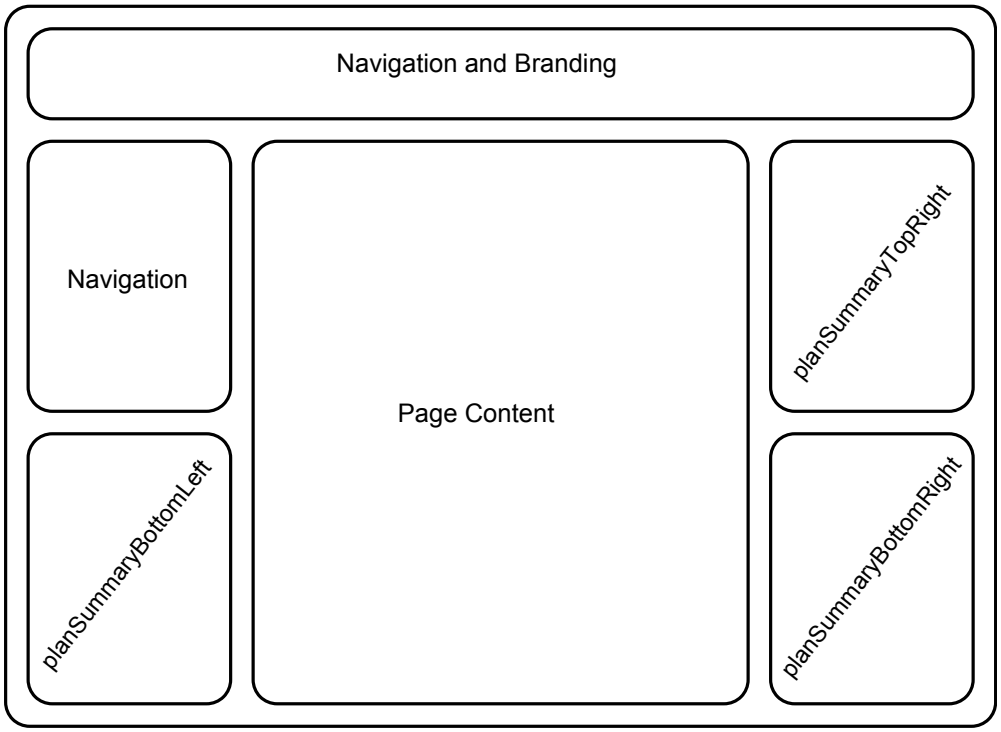
Для разработки взаимодействия для этого клиента нужно:

1. Определить требования для сводной страницы клиента
2. Создать точки взаимодействия для требований к предложениям
3. Сконфигурировать запись в журнал для предложений
4. Создать коды процедур
5. Связать ряды циклически сменяющихся изображений с предложениями

Приведенный ниже пример демонстрирует лишь суть подхода и не указывает наилучший способ написания интеграции. Например, ни один из этих примеров не содержит проверки ошибок, используемой классом Response.

Определить требования к сводной странице плана по сотовым телефонам

На следующей диаграмме показан макет сводной страницы плана по сотовым телефонам



Чтобы соответствовать требованиям к сводной странице плана по сотовым телефонам, вы определяете следующие элементы:

Требования	**Реализация**
Одно предложение нужно показать в зоне, отведенной предложениям по обновлению На странице, содержащей предложение по обновлению, должна быть задана область. Кроме того, после того, как Interact получит предложение для вывода, информацию необходимо записать в журнал.	• Точка взаимодействия: `ip_planSummaryBottomRight` • Событие: `evt_logOffer`
Два предложения по обновлению телефонов Необходимо задать каждую из областей на странице, где показаны обновления телефонов.	• Точка взаимодействия: `ip_planSummaryTopRight` • Точка взаимодействия: `ip_planSummaryBottomLeft`
Для анализа требуется записывать в журнал, какие из предложений были приняты и какие отклонены.	• Событие: `evt_offerAccept` • Событие: `evt_offerReject`
Не забудьте каждый раз передавать код процедуры для предложения при записи в журнал контакта по предложению, его принятия или отклонения.	`NameValuePair`

Требования	Реализация
Показывайте по три циклически сменяющихся изображения на странице. Свяжите изображения с предложениями.	

Создайте точки взаимодействия

Теперь можно попросить пользователя среды разработки создать для вас точки взаимодействия и события и начать писать код для интеграции с вашей областью взаимодействия.

Для каждой точки взаимодействия, содержащей предложение, необходимо вначале получить это предложение, а затем извлечь информацию, которая необходима для его вывода на экран. Например, затребуйте предложение для правой нижней области вашей веб-страницы (`planSummaryBottomRight`)

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

Этот вызов Response возвращает объект Response, содержащий ответ `OfferList`. Однако ваша веб-страница не может использовать объект `OfferList`. Для предложения требуется файл изображения, который, как вам известно, является одним из атрибутов предложения (`offerImg`). Вам нужно извлечь из `OfferList` нужный атрибут предложения.

```
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"))
  {
   /* Используйте для страницы это значение в своем коде,
   например: stringHtml = "<img src="+attribute.getValueAsString()+ " >   */
  }
 }
}
```

Сконфигурируйте запись в журнал

Теперь, когда вы показываете предложение, вам нужно записать его в журнал в качестве контакта.

```
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)
```

Вместо того, чтобы вызывать каждый из этих методов по отдельности, можно использовать метод `executeBatch`, как показано в следующем примере для области `planSummaryBottomLeft` веб-страницы.

```
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);
```

```
/** Построить массив команд */
Command[] commands =
{
 getOffersCommand,
 postEventCommand
};

/** Выполнить вызов */
BatchResponse batchResponse = api.executeBatch(sessionId, commands);
```

Определять UACIOfferTrackingCode в этом примере не требуется. Если вы не укажете UACIOfferTrackingCode, сервер среды выполнения Interact автоматически записывает в журнал последний рекомендованный список процедур в качестве контактов.

Создайте коды процедур

Там, где это необходимо, создайте пару NameValuePair для хранения кода процедуры, как показано в следующем примере.

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

Свяжите изображения с предложениями

Для второй области на странице, где показано обновление телефона, напишите что-нибудь, чтобы изменить изображение, выводимое каждые 30 секунд. Вы решили чередовать три изображения и собираетесь извлекать набор предложений в кэш, чтобы использовать его для смены изображений, с помощью следующего кода.

```
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) {
   // захватить значение атрибута предложения и сохранить его в другом месте;
   // это будет первое изображение для вывода
  }
  else if(x==1)
  {
   // захватить значение атрибута предложения и сохранить его в другом месте;
   // это будет второе изображение для вывода
  }
  else if(x==2)
  {
   // захватить значение атрибута предложения и сохранить его в другом месте;
   // это будет третье изображение для вывода
  }
 }
}
```

Необходимо записать выборку вашего клиентского кода из локального кэша и сделать только одну запись в контактах для каждого предложения после того, как показано соответствующее ему изображение. Для записи в контакты параметр UACITrackingCode необходимо разместить, как раньше. У каждого предложения есть свой код отслеживания.

```
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);
  }
 }
}
```

Для каждого предложения при щелчке по нему в журнал записываются принятые и отклоненные предложения. (В этом сценарии предложения, которые не выбраны явным образом, рассматриваются как отклоненные.) В приведенном ниже примере выбрано предложение `ip_planSummaryTopRight`:

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

На практике было бы лучше всего отправить эти три вызова `postEvent` с помощью метода `executeBatch`.

Разработка интеграции API Interact

Построение интеграции API Interact с вашей точкой контакта требует уделить внимание разработке, прежде чем вы сможете приступить к реализации. Чтобы решить, где в вашей точке контакта среда выполнения должна представлять предложения (определить точки взаимодействия) и какие еще виды отслеживания и интерактивных функций нужно использовать (определить ваши события), требуется совместная работа с вашей маркетинговой командой.

На стадии разработки это могут быть просто наброски. Например, для телекоммуникационного сайта сводная страница плана клиента должна содержать одно предложение по обновлению плана и два предложения по обновлению телефонов.

Как только ваша компания решила, где и как они хотят взаимодействовать с клиентами, надо уточнить детали с помощью Interact. Автору потоковой диаграммы нужно разработать интерактивные потоковые диаграммы, которые будут использоваться при наступлении событий пересегментации. Необходимо определить число точек взаимодействия и событий и задать их имена, а также решить, какие данные нужно передать для правильной сегментации, размещения событий и получения предложений. Пользователь среды разработки определяет точки взаимодействия и события для интерактивного канала. Далее вы используете эти

имена при написании кода для интеграции с вашей точкой контакта в среде выполнения. Чтобы задать, когда вам потребуется запись в журнал данных контактов и ответов для предложений, определите, какая нужна информация о показателях.

Вопросы для рассмотрения

При разработке взаимодействий имейте в виду возможное влияние на взаимодействие таких факторов, как не удовлетворяющее требованиям предложение, недоступный сервер среды выполнения и временной график процесса. Будьте внимательны, определяя предложения как отклоненные. Подумайте, какие дополнительные возможности продукта могут улучшить взаимодействие.

При разработке взаимодействия:

Создайте заполняющее содержимое по умолчанию
Создайте заполняющее содержимое по умолчанию, например, стандартное сообщение о торговой марке или пустое содержимое, для всех точек взаимодействия, где могут быть представлены предложения. Это заполняющее содержимое используется в случае отсутствия подходящих предложений, которые можно представить посетителю в текущей ситуации. Заполняющее содержимое по умолчанию назначается точке взаимодействия в виде строки по умолчанию.

Подготовьте резервный метод представления содержимого
Предусмотрите дополнительный метод представления содержимого на случай, если ваша точка контакта в силу непредвиденных обстоятельств не сможет обратиться к группе серверов среды выполнения.

Примите во внимание время запуска потоковых диаграмм
При инициировании событий, приводящих к пересегментированию посетителей, включая `postEvent` и `setAudience`, имейте в виду, что запуск потоковых диаграмм отнимает некоторое время. Метод `getOffers` ожидает окончания сегментации перед запуском метода `getOffers`. Слишком частая пересегментация может привести к ухудшению производительности ответов на вызовы `getOffers`.

Определите для себя, что означает "отклонить предложение"
Отклоненные предложения неоднократно упоминаются в нескольких отчетах, например, Сводке производительности предложений канала. В этом отчете показано много случаев, когда событие `postEvent` инициирует действие Записать отклонение предложения. Необходимо определить, относится ли действие Записать отклонение предложения к фактическому отклонению, вызванному, например, щелчком по ссылке **Нет, спасибо**. Или же действие Записать отклонение предложения относится к предложению, которое игнорируется, например, на странице выводится три различных рекламных баннера, ни один из которых не выбран.

Решите, какие возможности выбора предложений использовать
Есть несколько необязательных возможностей, с помощью которых можно улучшить выбор предложений Interact. К этим возможностям относятся:

- Обучение
- Подавление предложений
- Назначения отдельных предложений
- Другие элементы представления предложений

Необходимо определить, какие из этих дополнительных возможностей помогут улучшить ваши взаимодействия.

Глава 6. Управление API IBM Interact

Каждый раз при использовании метода `startSession` вы создаете на сервере среды выполнения сеанс среды выполнения Interact. Можно использовать свойства конфигурации для управления сеансами на сервере среды выполнения.

Возможно, вам понадобится сконфигурировать эти параметры при реализации интеграции Interact с вашей точкой контакта.

Эти свойства конфигурации содержатся в категории `sessionManagement`.

Локаль и API Interact

Interact можно использовать для точек контакта с языком, отличным от английского. Точка контакта и все строки в API используют локаль, заданную для пользователя среды выполнения.

Можно выбрать только одну локаль на группу серверов.

Пусть, например, вы создаете двух пользователей в среде выполнения, asm_admin_en с английской локалью и asm_admin_ru с русской локалью. Если ваша точка контакта разработана для носителей русского языка, задайте для свойства `asmUserForDefaultLocale` среды выполнения значение asm_admin_ru.

О мониторинге JMX

В Interact есть служба мониторинга Java Management Extensions (JMX), доступная в любой программе мониторинга JMX. При помощи мониторинга JMX можно отслеживать серверы среды выполнения и управлять ими.

Атрибуты JMX охватывают целый ряд подробностей о сервере среды выполнения. Например, атрибут JMX `ErrorCount` содержит число сообщений об ошибках с момента последнего сброса или запуска системы. Эту информацию можно использовать, чтобы узнать, часто ли возникают ошибки в системе. Кроме того, если вы запрограммировали сайт так, что окончание сеанса вызывается, только если кто-то завершит транзакцию, то можно сравнить `startSessionCount` с `endSessionCount` и увидеть, много ли есть незавершенных транзакций.

Interact поддерживает протоколы RMI и JMXMP, как определено в JSR 160. Соединиться со службой мониторинга JMX можно при помощи любого клиента JMX, удовлетворяющего требованиям JSR160.

Отслеживать интерактивные потоковые диаграммы можно только при помощи мониторинга JMX. Информация о потоковых диаграммах Interactive не выводится в мониторинге Campaign.

Примечание: Если используется IBM WebSphere с менеджером узлов, для поддержки мониторинга JMX нужно определить универсальный аргумент JVM.

Конфигурирование Interact для использования мониторинга JMX с протоколом RMI

Используйте данную процедуру, чтобы сконфигурировать Interact для использования мониторинга JMX с протоколом RMI

Об этой задаче

Адрес по умолчанию для мониторинга с протоколом RMI - `service:jmx:rmi:///jndi/rmi://`*`сервер_среды_выполнения`*`:`*`порт`*`/interact`.

Процедура

В Marketing Platform для среды выполнения отредактируйте следующие свойства конфигурации в категории `Interact > monitoring`.

Свойство конфигурации	Значение
`protocol`	**RMI**
`port`	Номер порта для службы JMX
`enableSecurity`	**False** Реализация Interact протокола RMI не поддерживает защиту.

Конфигурирование Interact для использования мониторинга JMX с протоколом JMXMP

Используйте данную процедуру, чтобы сконфигурировать Interact для использования мониторинга JMX с протоколом JMXMP

Прежде чем начать

Для протокола JMXMP требуется две дополнительных библиотеки в следующем порядке в пути классов: `InteractJMX.jar` и `jmxremote_optional.jar`. Оба этих файла можно найти в каталоге `lib` вашей установки среды выполнения.

Об этой задаче

Если вы включаете защиту, имя пользователя и пароль должны соответствовать пользователю в Marketing Platform для среды выполнения. Пустой пароль использовать нельзя.

Адрес по умолчанию для мониторинга с протоколом JMXMP - `service:jmx:jmxmp://`*`сервер_среды_выполнения`*`:`*`порт`*.

Процедура

1. Убедитесь, что библиотеки `InteractJMX.jar` и `jmxremote_optional.jar` включены в путь классов в правильном порядке. Если их нет в пути классов, добавьте их.
2. В Marketing Platform для среды выполнения отредактируйте следующие свойства конфигурации в категории `Interact > monitoring`.

Свойство конфигурации	Значение
`protocol`	**JMXMP**
`port`	номер порта для службы JMX

Свойство конфигурации	Значение
enableSecurity	**False** отключает защиту, **True** включает защиту

Конфигурирование Interact для использования сценариев jconsole при мониторинге JMX

Если у вас нет отдельной программы мониторинга JMX, можете использовать консоль jconsole, установленную вместе с JVM. Можно запускать консоль jconsole при помощи сценариев запуска в каталоге Interact/tools.

Об этой задаче

По умолчанию сценарий jconsole использует для мониторинга протокол JMXMP. Значение по умолчанию для jconsole.bat:

Соединение JMXMP

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

Соединение RMI

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

Процедура

1. Откройте в текстовом редакторе файл Interact\tools\jconsole.bat (Windows) или Interact/tools/jconsole.sh (UNIX).
2. Задайте в INTERACT_LIB полный путь к каталогу *InteractInstallationDirectory*/lib.
3. Задайте в HOST имя хоста сервера времени выполнения, который нужно отслеживать.
4. Задайте в PORT порт, на котором сконфигурировали ожидание приема в JMX в свойстве Interact > monitoring > port.
5. Необязательно: Если для мониторинга используется протокол RMI, добавьте метку комментария перед соединением JMXMP и раскомментируйте соединение RMI.

Атрибуты JMX

Есть много атрибутов, доступных для мониторинга JMX. Атрибуты среды разработки охватывают мониторинг ETL хронологии контактов и ответов. Атрибуты среды выполнения охватывают исключения, ряд других атрибутов потоковой диаграммы, локаль, программу журнала и статистику пула потоков. Кроме того, доступен ряд атрибутов статистики обслуживания. Все данные от мониторинга JMX относятся ко времени после последней перезагрузки или запуска системы. Например, все счетчики показывают число случаев с последней последней перезагрузки или запуска системы, а не с момента установки.

Атрибуты монитора ETL хронологии контактов и ответов

Атрибуты монитора ETL хронологии контактов и ответов относятся к среде разработки. Остальные атрибуты относятся к среде выполнения.

Таблица 9. Монитор ETL хронологии контактов и ответов

Атрибут	Описание
AvgCHExecutionTime	Среднее число миллисекунд, израсходованных модулем хронологии контактов и ответов на запись в таблицу хронологии контактов. Это среднее вычисляется только для операций, которые успешно завершились с выполнением по меньшей мере одной записи в таблицу хронологии контактов.
AvgETLExecutionTime	Среднее число миллисекунд, израсходованных модулем хронологии контактов и ответов на чтение данных из среды выполнения. Это среднее время включает в себя как успешно завершенные операции, так и завершившиеся неудачно.
AvgRHExecutionTime	Среднее число миллисекунд, израсходованных модулем хронологии контактов и ответов на запись в таблицу хронологии ответов. Это среднее вычисляется только для операций, которые успешно завершились с выполнением по меньшей мере одной записи в таблицу хронологии ответов.
ErrorCount	Число сообщений об ошибках с момента последнего сброса или запуска системы, если они были.
HighWaterMarkCHExecutionTime	Максимальное число миллисекунд, израсходованных модулем хронологии контактов и ответов на запись в таблицу хронологии контактов. Это значение вычисляется только для операций, которые успешно завершились с выполнением по меньшей мере одной записи в таблицу хронологии контактов.
HighWaterMarkETLExecutionTime	Максимальное число миллисекунд, израсходованных модулем хронологии контактов и ответов на чтение данных из среды выполнения. Это значение включает в себя как успешно завершенные операции, так и завершившиеся неудачно.
HighWaterMarkRHExecutionTime	Максимальное число миллисекунд, израсходованных модулем хронологии контактов и ответов на запись в таблицу хронологии ответов. Это значение вычисляется только для операций, которые успешно завершились с выполнением по меньшей мере одной записи в таблицу хронологии ответов.
LastExecutionDuration	Время в минутах, израсходованное модулем хронологии контактов и ответов на последнее копирование.

Таблица 9. Монитор ETL хронологии контактов и ответов (продолжение)

Атрибут	Описание
NumberOfExecutions	Сколько раз запускался модуль хронологии контактов и ответов с момента инициализации.
LastExecutionStart	Время последнего запуска модуля хронологии контактов и ответов.
LastExecutionSuccessful	Если значение - true, последний запуск модуля хронологии контактов и ответов завершился успешно. Если значение - false, возникла ошибка.
NumberOfContactHistoryRecordsMarked	Число записей хронологии контактов в таблице `UACI_CHStaging`, которые перемещаются во время текущего выполнения модуля хронологии контактов и ответов. Это значение больше нуля, только если модуль хронологии контактов и ответов запущен.
NumberOfResponseHistoryRecordsMarked	Число записей хронологии ответов в таблице `UACI_RHStaging`, которые перемещаются во время текущего выполнения модуля хронологии контактов и ответов. Это значение больше нуля, только если модуль хронологии контактов и ответов запущен.

Атрибуты исключительных ситуаций

Атрибуты исключительных ситуаций - часть среды выполнения.

Таблица 10. Исключительные ситуации

Атрибут	Описание
errorCount	Число сообщений об ошибках с момента последнего сброса или запуска системы.
warningCount	Число сообщений предупреждения с момента последнего сброса или запуска системы.

Атрибуты статистики механизма потоковых диаграмм

Атрибуты статистики механизма потоковых диаграмм - часть среды выполнения.

Таблица 11. Статистика механизма потоковых диаграмм

Атрибут	Описание
activeProcessBoxThreads	Число потоков обработки потоковой диаграммы, активных в настоящее время (во всех выполнениях).
activeSchedulerThreads	Число потоков планировщика потоковой диаграммы, активных в настоящее время.

Таблица 11. Статистика механизма потоковых диаграмм (продолжение)

Атрибут	Описание
avgExecutionTimeMillis	Среднее время выполнения потоковой диаграммы в миллисекундах.
CurrentJobsInProcessBoxQueue	Число заданий, ожидающих запуска потоками обработки потоковой диаграммы.
CurrentJobsInSchedulerQueue	Число заданий, ожидающих запуска потоками планировщика потоковой диаграммы.
maximumProcessBoxThreads	Максимальное число потоков обработки потоковой диаграммы, которое может быть запущено (во всех выполнениях).
maximumSchedulerThreads	Максимальное число потоков обработки планировщика потоковой диаграммы, которое может быть запущено (по одному потоку на каждое выполнение).
numExecutionsCompleted	Общее число завершенных выполнений потоковой диаграммы.
numExecutionsStarted	Общее число запущенных выполнений потоковой диаграммы.

Атрибуты конкретных потоковых диаграмм для интерактивного канала

Атрибуты конкретных потоковых диаграмм для интерактивного канала относятся к среде выполнения.

Таблица 12. Конкретные потоковые диаграммы для интерактивного канала

Атрибут	Описание
AvgExecutionTimeMillis	Среднее время выполнения в миллисекундах для этой потоковой диаграммы в этом интерактивном канале.
HighWaterMarkForExecutionTime	Максимальное время выполнения в миллисекундах для этой потоковой диаграммы в этом интерактивном канале.
LastCompletedExecutionTimeMillis	Время выполнения в миллисекундах для последнего завершения этой потоковой диаграммы в этом интерактивном канале.
NumExecutionsCompleted	Общее число выполнений, завершившихся для этой потоковой диаграммы в этом интерактивном канале.
NumExecutionsStarted	Общее число выполнений, запущенных для этой потоковой диаграммы в этом интерактивном канале.

Атрибуты локали

Атрибуты локали - часть среды выполнения.

Таблица 13. Локаль

Атрибут	Описание
locale	Параметр локали для клиента JMX.

Атрибуты конфигурации регистратора

Атрибуты конфигурации регистратора - часть среды выполнения.

Таблица 14. Конфигурация регистратора

Атрибут	Описание
category	Измените категорию журнала, для которой можно управлять уровнем журнала.

Атрибуты статистики пула потоков служб

Атрибуты статистики пула потоков служб - часть среды выполнения.

Таблица 15. Статистика пула потоков служб

Атрибут	Описание
activeContactHistThreads	Приблизительное число активных потоков, которыми выполняются задачи для хронологии контактов и хронологии ответов.
activeFlushCacheToDBThreads	Приблизительное число активных потоков, которыми выполняются задачи перемещения кэшированной статистики на склад данных.
activeOtherStatsThreads	Приблизительное число активных потоков, которыми выполняются задачи для статистики соответствия требованиям, статистики интенсивности событий и статистики использования строки по умолчанию.
CurrentHighWaterMarkInContactHistQueue	Самое большое число записей, поставленных в очередь на запись в журнал службой, которая собирает данные хронологии контактов и ответов.
CurrentHighWaterMarkInFlushCachetoDBQueue	Самое большое число записей, поставленных в очередь на запись в журнал службой, которая записывает данные кэша в таблицы базы данных.

Таблица 15. Статистика пула потоков служб (продолжение)

Атрибут	Описание
CurrentHighWaterMarkInOtherStatsQueue	Самое большое число записей, поставленных в очередь на запись в журнал службой, которая собирает статистику соответствия требованиям по предложениям, статистику использования строки по умолчанию, статистику интенсивности событий и пользовательские данные регистрации в таблицу.
currentMsgsInContactHistQueue	Число заданий в очереди для пула потоков, использующегося для хронологии контактов и ответов.
currentMsgsInFlushCacheToDBQueue	Число заданий в очереди для пула потоков, использующегося для перемещения кэшированной статистики на склад данных.
currentMsgsInOtherStatsQueue	Число заданий в очереди для пула потоков, использующегося для статистики соответствия требованиям, статистики интенсивности событий и статистики использования строки по умолчанию.
maximumContactHistThreads	Наибольшее число одновременных потоков в пуле, использующемся для хронологии контактов и хронологии ответов.
maximumFlushCacheToDBThreads	Наибольшее число одновременных потоков в пуле, использующемся для перемещения кэшированной статистики на склад данных.
maximumOtherStatsThreads	Наибольшее число одновременных потоков в пуле, использующемся для статистики соответствия требованиям, статистики интенсивности событий и статистики использования строки по умолчанию.

Атрибуты статистики служб

Статистические показатели служб состоят из набора атрибутов для каждой службы.

- ContactHistoryMemoryCacheStatistics - Служба, собирающая данные для промежуточных таблиц хронологии контактов.
- CustomLoggerStatistics - Служба, собирающая пользовательские данные для записи в таблицу (событие, использующее параметр события `UACICustomLoggerTableName`).
- Default Statistics - служба, собирающая статистические данные о числе использований строки по умолчанию для точки взаимодействия.
- Eligibility Statistics - служба, записывающая статистику соответствия требованиям по предложениям.

- Event Activity Statistics - служба, собирающая статистику интенсивности событий, включая и системные события, такие как `getOffer` или `startSession`, и пользовательские события, инициированные триггером `postEvent`.
- Response History Memory Cache Statistics - служба, записывающая статистику в промежуточные таблицы хронологии ответов.
- Cross-session Response Statistics - служба, собирающая данные отслеживания межсеансовых ответов.

Таблица 16. Статистика службы

Атрибут	Описание
Count	Число обработанных сообщений.
ExecTimeInsideMutex	Время в миллисекундах, израсходованное на обработку сообщений для этой службы, не считая времени, израсходованного на ожидание других потоков. Если существует большая разница между ExecTimeInsidMutex и ExecTimeMillis, то, возможно, есть смысл изменить размер пула буферов для этой службы.
ExecTimeMillis	Время в миллисекундах, израсходованное на обработку сообщений для этой службы, включая время, израсходованное на ожидание других потоков.
ExecTimeOfDBInsertOnly	Время в миллисекундах, израсходованное только на обработку порции пакетной вставки.
HighWaterMark	Максимальное число сообщений, обработанных для этой службы.
NumberOfDBInserts	Общее число запусков пакетных вставок.
TotalRowsInserted	Общее число строк, вставленных в базу данных.

Атрибуты статистики служб - утилиты загрузки баз данных

Атрибуты статистики служб - утилиты загрузки баз данных - часть среды выполнения.

Таблица 17. Статистика служб - утилита загрузки баз данных

Атрибут	Описание
ExecTimeOfWriteToCache	Время в миллисекундах, израсходованное на запись в файловый кэш, включая запись в файл и, при необходимости, получение первичного ключа от базы данных.
ExecTimeOfLoaderDBAccessOnly	Время в миллисекундах, израсходованное только на выполнение порции загрузчика базы данных.
ExecTimeOfLoaderThreads	Время в миллисекундах, израсходованное потоками загрузчика базы данных.
ExecTimeOfFlushCacheFiles	Время в миллисекундах, израсходованное на очистку кэша и воссоздание новых файлов.

Таблица 17. Статистика служб - утилита загрузки баз данных (продолжение)

Атрибут	Описание
ExecTimeOfRetrievePKDBAccess	Время в миллисекундах, израсходованное на получение доступа к базе данных первичного ключа.
NumberOfDBLoaderRuns	Общее число запусков загрузчика базы данных.
NumberOfLoaderStagingDirCreated	Общее число созданных промежуточных каталогов
NumberOfLoaderStagingDirRemoved	Общее число удаленных промежуточных каталогов
NumberOfLoaderStagingDirMovedToAttention	Общее число промежуточных каталогов, переименованных в attention.
NumberOfLoaderStagingDirMovedToError	Общее число промежуточных каталогов, переименованных в error.
NumberOfLoaderStagingDirRecovered	Общее число восстановленных промежуточных каталогов, включая случаи во время запуска и повторное выполнение фоновыми потоками.
NumberOfTimesRetrievePKFromDB	Общее число получений первичного ключа из базы данных.
NumberOfLoaderThreadsRuns	Общее число запусков потока загрузчика базы данных.
NumberOfFlushCacheFiles	Общее число очисток файлового кэша.

Атрибуты статистики API

Атрибуты статистики API - часть среды выполнения.

Таблица 18. Статистика API

Атрибут	Описание
endSessionCount	Число вызовов API `endSession` с момента последнего сброса или запуска системы.
endSessionDuration	Время, истекшее с момента последнего вызова API `endSession`, в миллисекундах.
executeBatchCount	Число вызовов API `executeBatch` с момента последнего сброса или запуска системы.
executeBatchDuration	Время, истекшее с момента последнего вызова API `executeBatch`, в миллисекундах.
getOffersCount	Число вызовов API `getOffers` с момента последнего сброса или запуска системы.
getOffersDuration	Время, истекшее с момента последнего вызова API `getOffer`, в миллисекундах.
getProfileCount	Число вызовов API `getProfile` с момента последнего сброса или запуска системы.
getProfileDuration	Время, истекшее с момента последнего вызова API `getProfileDuration`, в миллисекундах.
getVersionCount	Число вызовов API `getVersion` с момента последнего сброса или запуска системы.

Таблица 18. Статистика API (продолжение)

Атрибут	Описание
getVersionDuration	Время, истекшее с момента последнего вызова API `getVersion`, в миллисекундах.
loadOfferSuppressionDuration	Время, истекшее с момента последнего вызова API `loadOfferSuppression`.
LoadOffersBySQLCount	Число вызовов API `LoadOffersBySQL` с момента последнего сброса или запуска системы.
LoadOffersBySQLDuration	Время, истекшее с момента последнего вызова API `LoadOffersBySQL`, в миллисекундах.
loadProfileDuration	Время, истекшее с момента последнего вызова API `loadProfile`, в миллисекундах.
loadScoreOverrideDuration	Время, истекшее с момента последнего вызова API `loadScoreOverride`, в миллисекундах.
postEventCount	Число вызовов API `postEvent` с момента последнего сброса или запуска системы.
postEventDuration	Время, истекшее с момента последнего вызова API `postEvent`, в миллисекундах.
runSegmentationDuration	Время, истекшее с момента последнего вызова API `runSegmentation`, в миллисекундах.
setAudienceCount	Число вызовов API `setAudience` с момента последнего сброса или запуска системы.
setAudienceDuration	Время, истекшее с момента последнего вызова API `setAudience`, в миллисекундах.
setDebugCount	Число вызовов API `setDebug` с момента последнего сброса или запуска системы.
setDebugDuration	Время, истекшее с момента последнего вызова API `setDebug`, в миллисекундах.
startSessionCount	Число вызовов API `startSession` с момента последнего сброса или запуска системы.
startSessionAverage	Среднее время, истекшее с момента последнего вызова API `startSession`, в миллисекундах.
ActiveSessionCount	Число сеансов, которые в настоящее время активны в экземпляре среды выполнения interact. **Примечание:** ActiveSessionCount в JMX MBean com.unicacorp.interact:type=api, group=Statistics не учитывает события с истекшим сроком ожидания и, следовательно, не всегда совпадает с числом активных сеансов.

Атрибуты статистики оптимизатора обучения

Атрибуты статистики оптимизатора обучения - часть среды выполнения.

Таблица 19. Статистика оптимизатора обучения

Атрибут	Описание
LearningOptimizerAcceptCalls	Число событий принятия, переданных в модуль обучения.
LearningOptimizer AcceptTrackingDuration	Общее число миллисекунд, израсходованных на запись событий принятия в журнал модуля обучения.
LearningOptimizerContactCalls	Число событий контакта, переданных в модуль обучения.
LearningOptimizer ContactTrackingDuration	Общее число миллисекунд, израсходованных на запись событий контакта в журнал модуля обучения.
LearningOptimizerLogOtherCalls	Число событий, отличных от контакта или принятия, переданных в модуль обучения.
LearningOptimizer LogOtherTrackingDuration	Время в миллисекундах, израсходованное на запись других событий (не контактов и не принятий) в журнал модуля обучения.
LearningOptimizer NonRandomCalls	Сколько раз была применена сконфигурированная реализация обучения.
LearningOptimizer RandomCalls	Сколько раз была пропущена сконфигурированная реализация обучения и применена случайная выборка.
LearningOptimizer RecommendCalls	Число требований рекомендации, переданных в модуль обучения.
LearningOptimizer RecommendDuration	Общее число миллисекунд, израсходованных на обучение алгоритма рекомендаций.

Атрибуты статистики предложений по умолчанию

Атрибуты статистики предложений по умолчанию - часть среды выполнения.

Таблица 20. Статистика предложений по умолчанию

Атрибут	Описание
LoadDefaultOffersDuration	Время на загрузку предложений по умолчанию.
DefaultOffersCalls	Сколько раз загружались предложения по умолчанию.

Атрибуты Диспетчеры инициируемых сообщений

Атрибуты диспетчеров инициируемых сообщений - часть среды выполнения.

Таблица 21. Диспетчеры инициируемых сообщений

Атрибут	Описание
NumRequested	Общее число предложений, для которых запрашивалась диспетчеризация этим диспетчером.

Таблица 21. Диспетчеры инициируемых сообщений (продолжение)

Атрибут	Описание
NumDispatched	Общее число предложений, успешно распределенных этим диспетчером.
AvgExecutionTime	Среднее время в миллисекундах, расходуемое этим диспетчером на диспетчеризацию предложения. В этом значении учитываются только предложения, успешно распределенные по шлюзам.
CurrentQueueSize	Число предложений, которые в настоящее время ожидают диспетчеризации.
GatewayInvocation	Число предложений и среднее время диспетчеризации в миллисекундах, уделенные каждому шлюзу этим диспетчером. Формат этого значения `{имя шлюза=[число предложений, среднее время диспетчеризации]}`.

Атрибуты Шлюзы инициируемых сообщений

Атрибуты шлюзов инициируемых сообщений - часть среды выполнения.

Таблица 22. Шлюзы инициируемых сообщений

Атрибут	Описание
NumValidationRequested	Общее число предложений, для которых этот шлюз запросил проверку правильности.
NumValidated	Общее число предложений, для которых этот шлюз успешно подтвердил правильность.
AvgValidationTime	Среднее время в миллисекундах, расходуемое этим шлюзом на проверку правильности предложения. В этом значении учитываются только предложения, правильность которых была успешно подтверждена.
NumDeliveryRequested	Общее число предложений, для которых этот шлюз запросил доставку.
NumDelivered	Общее число предложений, которые этот шлюз успешно доставил.
AvgDeliveryTime	Среднее время в миллисекундах, расходуемое этим шлюзом на доставку предложения. В этом значении учитываются только предложения, которые были успешно доставлены.

Атрибуты Сообщения инициируемых сообщений

Атрибуты сообщений инициируемых сообщений - часть среды выполнения.

Таблица 23. Сообщения инициируемых сообщений

Атрибут	Описание
ProcessSuccessCount	Общее число успешных выполнений этого инициированного сообщения.
AvgSuccessProcessTime	Среднее время в миллисекундах, расходуемое на успешное выполнение этого инициированного сообщения.
ProcessErrorCount	Общее число неудачных выполнений этого инициированного сообщения.
AvgErrorProcessTime	Среднее время в миллисекундах, расходуемое на неудачное выполнение этого инициированного сообщения.
SelectBranchCount	Общее число выполнений выбора ветви при обработке инициированных сообщений.
AvgSelectBranchTime	Среднее время в миллисекундах на выполнение выбора ветви при обработке инициированных сообщений.
SelectOfferCount	Общее число выполнений выбора предложения при обработке инициированных сообщений.
AvgSelectOfferTime	Среднее время в миллисекундах на выполнение выбора предложения при обработке инициированных сообщений.
SelectChannelCount	Общее число выполнений выбора канала при обработке инициированных сообщений.
AvgSelectChannelTime	Среднее время в миллисекундах на выполнение выбора канала при обработке инициированных сообщений.
FlowchartWaitCount	Общее число случаев, когда это инициированное сообщение ожидало завершения сегментации.
AvgFlowchartWaitTime	Среднее время в миллисекундах, которое это инициированное сообщение ожидало завершения в каждом выполнении.
WaitFlowchartTimeoutCount	Общее число истечений срока ожидания, когда это инициированное сообщение не дождалось завершения сегментации.

Операции JMX

Есть ряд операций, доступных при мониторинге JMX.

В следующей таблице описаны операции, доступные для мониторинга JMX.

Группа	Атрибут	Описание
Конфигурация регистратора	activateDebug	Задайте уровню записи в журнал, указанный в `Interact/conf/interact_log4j.properties`, значение debug.
Конфигурация регистратора	activateError	Задайте уровню записи в журнал, указанный в `Interact/conf/interact_log4j.properties`, значение error.
Конфигурация регистратора	activateFatal	Задайте уровню записи в журнал, указанный в `Interact/conf/interact_log4j.properties`, значение fatal.
Конфигурация регистратора	activateInfo	Задайте уровню записи в журнал, указанный в `Interact/conf/interact_log4j.properties`, значение info.
Конфигурация регистратора	activateTrace	Задайте уровню записи в журнал, указанный в `Interact/conf/interact_log4j.properties`, значение trace.
Конфигурация регистратора	activateWarn	Задайте уровню записи в журнал, указанный в `Interact/conf/interact_log4j.properties`, значение warn.
Локаль	changeLocale	Измените локаль клиента JMX. Локали, поддерживаемые Interact, - `de`, `en`, `es` и `fr`.
ContactResponseHistory ETLMonitor	сброс	Сбросьте все счетчики.
Статистика предложений по умолчанию	updatePollPeriod	Период defaultOfferUpdatePollPeriod обновлений. Это значение задает время ожидания в секундах, после которого система обновляет предложения по умолчанию в кэше. Если задать `-1`, система читает заданное число предложений по умолчанию только при запуске.

Глава 7. Классы и методы для API IBM Interact Java, SOAP и REST

В следующих разделах перечислены требования и другие подробности, которые нужно знать перед началом работы с API Interact.

Примечание: В этом разделе предполагается, что вы знакомы с вашей точкой взаимодействия, языком программирования Java и API на основе Java.

У API Interact есть адаптер клиента Java, использующий сериализацию Java через HTTP. Кроме того, Interact содержит WSDL для поддержки клиентов SOAP. WSDL располагает тем же набором функций, что и адаптер клиента Java, поэтому следующие ниже разделы применимы к нему, за исключением приведенных примеров.

Примечание: Несколько вхождений одного параметра в вызове API не поддерживается.

Классы API Interact

API Interact основан на классе `InteractAPI`.

Существует 6 интерфейсов поддержки.

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

У этих интерфейсов есть три поддерживающих конкретных класса. Для двух следующих конкретных классов необходимо создать экземпляры и передать их в качестве аргументов в методы API Interact:

- NameValuePairImpl
- CommandImpl

Третий конкретный класс с именем `AdvisoryMessageCode` содержит константы, которые используются для различения кодов сообщений, возвращаемых сервером, везде, где это применимо.

В остальной части этого раздела описаны методы, связанные с работой API Interact.

Предварительные требования сериализации Java через HTTP

Адаптер клиента Java использует сериализацию Java через HTTP.

Предварительные требования для работы с адаптером клиента Java для сериализации Java через HTTP:

1. Добавьте следующий файл в ваш `CLASSPATH`:

домашний_каталог_Interact/lib/interact_client.jar

2. Все объекты, передаваемые в обе стороны между клиентом и сервером, можно найти в пакете `com.unicacorp.interact.api`. Дополнительную информацию смотрите в API Javadoc Interact, установленном на сервере среды выполнения в `домашний_каталог_Interact/docs/apiJavaDoc`. Javadoc можно просмотреть, открыв файл `index.html` в этом каталоге при помощи любого браузера.
3. Чтобы получить экземпляр класса `InteractAPI`, вызовите статический метод `getInstance`, используя URL сервера среды выполнения Interact.

Предварительные требования SOAP

Прежде чем вы сможете получить доступ к серверу среды выполнения при помощи SOAP, необходимо выполнить несколько подготовительных задач по конфигурированию вашей среды.

Важное замечание: Тестирование производительности показывает, что адаптер сериализации Java работает значительно быстрее, чем сгенерированный клиент SOAP. Для наилучшей производительности пользуйтесь адаптером сериализации Java везде, где это возможно.

Для доступа к серверу среды выполнения с помощью SOAP нужно сделать следующее:

1. Преобразуйте WSLD API Interact при помощи выбранного вами инструментария SOAP.

   WSLD API Interact устанавливается вместе с Interact в каталоге `Interact/conf`.

   При конфигурировании SOAP с помощью файлов WSDL XML необходимо изменить ваши URL, указав в них имя хоста и порт сервера среды выполнения.

   Текст WSDL доступен в конце Руководства по администрированию Interact.
2. Установите и сконфигурируйте сервер среды выполнения.

   Сервер среды выполнения должен работать, чтобы полностью протестировать вашу интеграцию.
3. Проверьте использование правильной версии SOAP.

   Interact использует axis2 1.3 в качестве инфраструктуры SOAP на серверах среды выполнения Interact. Подробную информацию о том, какие версии SOAP поддерживает axis2 1.3, смотрите на сайте:

   Apache Axis2

   Interact был протестирован при помощи клиентов axis2, XFire, JAX-WS-Ri, DotNet, SOAPUI и IBM RAD SOAP.

Предварительные требования для REST

Один из методов вызова API Interact, называемый здесь API REST, - это использование вызовов формата JSON (JavaScript Object Notation) через HTTP. API REST имеет перед SOAP преимущество в производительности, хотя адаптер сериализации Java по-прежнему остается наиболее быстрым методом вызова API Interact .

Прежде чем начать работу с API REST, учтите следующее:

- URL, поддерживающий вызовы REST для API Interact:

  `http://сервер_среды_выполнения_Interact:порт/interact/servlet/RestServlet`, замещающий фактическое имя хоста или IP-адрес сервера среды выполнения Interact и порта, на котором внедрен Interact.

- Существует два класса Interact, связанных с API REST: `RestClientConnector`, который служит помощником для соединения с экземпляром среды выполнения Interact через REST в формате JSON, и `RestFieldConstants`, который описывает основной формат сообщения JSON, используемого для требований и ответов API.
- В файле `домашний_каталог_Interact/samples/javaApi/InteractRestClient.java` приведен пример клиента REST. Хотя пример кода - это просто пример, он должен давать хорошую отправную точку для демонстрации того, как используется API REST.
- Полное описание классов API REST вместе со всей прочей информацией об API Interact смотрите в документе Javadoc, установленном на сервере среды выполнения в каталоге `домашний_каталог_Interact/docs/apiJavaDoc`.
- API REST возвращает SessionID и сообщения в формате с эскейп-выделением управляющих символов HTML, а не в формате Unicode.

Помимо приведенной здесь информации, API REST поддерживает все методы, которые поддерживаются другими протоколами для работы с API Interact.

API JavaDoc

В дополнение к руководству администратора Interact вместе с сервером среды выполнения устанавливается Javadoc для API Interact. Javadoc устанавливается для вашего сведения в каталоге `домашний_каталог_Interact/docs/apiJavaDoc`.

Примеры API

Все примеры в этом руководстве созданы с использованием сериализации Java через адаптер HTTP. Классы, созданные при помощи WSDL, могут различаться в зависимости от инструментария SOAP и выбранных вами опций. Если используется SOAP, эти примеры могут не работать должным образом в вашей среде.

Работа с данными сеанса

Если вы инициируете сеанс с помощью метода `startSession`, данные сеанса будут загружены в память. На протяжении сеанса можно считывать и записывать данные сеанса (которые являются надмножеством данных статического профиля).

Сеанс содержит следующие данные:

- Данные статического профиля
- Назначения сегмента
- Данные реального времени
- Рекомендации предложения

Все данные сеанса доступны, пока вы не вызовете метод `endSession` или пока не истечет время `SessionTimeout`. По окончании сеанса все данные, не сохраненные в явном виде в хронологии контактов, хронологии ответов или какой-либо другой таблице базы данных, будут потеряны.

Данные хранятся в виде набора пар имя - значение. Если данные считаны из таблицы базы данных, используется имя столбца таблицы.

Можно создать эти пары имя - значение, когда вы работаете с API Interact. Необязательно объявлять все пары имя - значение в глобальной области. При задании новых параметров событий в виде пар имя - значение среда выполнения добавляет пары имя - значение к данным сеанса. Например, если вы используете параметры событий с методом `postEvent`, среды выполнения добавит параметры

событий в данные сеанса, даже если параметры событий были недоступны в данных профиля. Эти данные существуют только в данных сеанса.

В любое время можно перезаписать данные сеанса. Например, если часть профиля покупателя включает `creditScore`, вы можете передать параметр события с помощью пользовательского типа `NameValuePair`. В классе `NameValuePair` можно изменить значение при помощи методов `setName` и `setValueAsNumeric`. Имена должны соответствовать друг другу. В данных сеанса имя не является регистрозависимым. Таким образом, оценка `creditScore` будет перезаписана вариантами `creditscore` или `CrEdItScOrE`.

Сохраняются только последние данные, записанные в сеанс. Например, метод `startSession` загружает данные профиля для значения `lastOffer`. Метод `postEvent` перезаписывает `lastOffer`. Затем второй метод `postEvent` перезаписывает `lastOffer`. В среде выполнения сохраняются только данные, записанные в данных сеанса вторым методом `postEvent`.

По завершении сеанса данные будут потеряны, если не предпринять специальных шагов, например, не записать данные в таблицу базы данных с помощью процесса Снимок в вашей интерактивной потоковой диаграмме. Если вы планируете использовать процесс Снимок, учтите, что для имен должны соблюдаться ограничения вашей базы данных. Например, если разрешено использовать в именах столбцов не более 256 символов, имя для пары имя - значение не может содержать более 256 символов.

О классе InteractAPI

Класс InteractAPI содержит методы, используемые для интеграции точки взаимодействия с сервером среды выполнения. Все прочие классы и методы в API Interact поддерживает методы в этом классе.

Надо скомпилировать реализацию с помощью файла `interact_client.jar`, расположенного в каталоге `lib` вашей установки Interact среды выполнения.

endSession

Метод `endSession` отмечает конец сеанса во время выполнения. Когда сервер среды выполнения получает этот метод, сервер среды выполнения записывает информацию в журнал хронологии, очищает память и так далее.

`endSession(String sessionID)`

- **sessionID**-Уникальная строка, определяющая сеанс.

Если метод endSession не вызван, сеансы среды выполнения завершаются по истечению срока бездействия. Срок ожидания конфигурируется с помощью свойства `sessionTimeout`.

Возвращаемое значение

Сервер среды выполнения отвечает на метод `endSession` объектом `Response` со следующими заполненными атрибутами:

- SessionID
- ApiVersion
- StatusCode
- AdvisoryMessages

Пример

В следующем примере показан метод endSession и возможный анализ ответа. sessionId - это та же строка для идентификации сеанса, что и используемая вызовом startSession, который запустил этот сеанс.

```
response = api.endSession(sessionId);
    // проверить успешность ответа
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("Вызов endSession обработан без предупреждений или ошибок");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов endSession обработан с предупреждением");
    }
    else
    {
        System.out.println("Вызов endSession обработан с ошибкой");
    }
    // В случае каких-либо неудач должны быть поясняющие рекомендации
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("endSession",
   response.getAdvisoryMessages());
```

executeBatch

Метод executeBatch позволяет выполнить несколько методов за одно требование к серверу среды выполнения.

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

- **sessionID** - строка, определяющая ID сеанса. Этот ID сеанса используется для всех команд, выполняемых в этом вызове метода.
- **commandImpl[]** - Массив объектов CommandImpl, по одному для каждой команды, которую вы хотите выполнить.

Результат вызова этого метода эквивалентен прямому вызову каждого метода в массиве Command. Этот метод минимизирует число фактических требований к серверу среды выполнения. Сервер среды выполнения выполняет каждый метод последовательно; для каждого вызова любая ошибка или предупреждения захватываются объектом Response, соответствующем этому вызову метода. Если возникает ошибкой, executeBatch продолжает выполнять оставшуюся часть вызовов в пакете. Если выполнение какого-либо метода приводит к ошибке, состояние высокого уровня для объекта BatchResponse отражает ту ошибку. Если ошибок не происходит, состояние высокого уровня отражает любые возможные предупреждения. Если предупреждений нет, состояние высокого уровня отражает успешное выполнение пакета.

Возвращаемое значение

Сервер среды выполнения отвечает на executeBatch объектом BatchResponse.

Пример

Следующий пример показывает, как вызвать все методы getOffer и postEvent в одном вызове executeBatch и как обработать ответ.

```
/** Определяем все переменные для всех элементов executeBatch*/
String sessionId="MySessionID-123";
String interactionPoint = "Баннер Страницы Обзора 1";
int numberRequested=1;
String eventName = "logOffer";
```

```
/** строим команду getOffers */
Command getOffersCommand = new CommandImpl();
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(interactionPoint);
getOffersCommand.setNumberRequested(numberRequested);

/** строим команду postEvent */
Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEventParameters(postEventParameters);
postEventCommand.setEvent(eventName);

/** Построить массив команд */
Command[] commands =
{
  getOffersCommand,
  postEventCommand,
};

/** Выполнить вызов */
BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Обработать ответ соответствующим образом */
// Код состояния высшего уровня - простой способ определить, были ли
// неудачи в массиве объектов Response
if(batchResponse.getBatchStatusCode() == Response.STATUS_SUCCESS)
{
  System.out.println("ExecuteBatch отработал прекрасно!");
}
else if(batchResponse.getBatchStatusCode() == Response.STATUS_WARNING)
{
  System.out.println("Вызов ExecuteBatch обработан хотя бы с одним предупреждением");
}
else
{
  System.out.println("Вызов ExecuteBatch обработан хотя бы с одной ошибкой");
}

// Итерация по массиву и печать сообщения для любой неудачи
for(Response response : batchResponse.getResponses())
{
  if(response.getStatusCode() != Response.STATUS_SUCCESS)
  {
  printDetailMessageOfWarningOrError("executeBatchCommand",
 response.getAdvisoryMessages());
  }
}
```

Запись требований XML executeBatch() для API SOAP Interact

Используйте следующие действия для записи требований XML `executeBatch ()` для API SOAP Interact.

Об этой задаче

XML требования для вызовов API SOAP с единичной операцией (`startSession`, `getOffers`, `setAudience`, `endSession` и т.п.) нельзя прямо копировать и вставлять в вызов `executeBatch()` с несколькими операциями. Подкоманды в вызовах `executeBatch ()` используют слегка отличающиеся структуры требований WSDL и XML по сравнению с используемыми в вызовах API с единственной операцией. Если скопировать и вставить элементы XML из требований API с одной операцией в требования `executeBatch` с несколькими операциями, это отличие в структурах приведет к ошибкам.

Пример ответов с сообщениями об ошибках:

```
** Элемент ответа XML: <ns0:faultstring>org.apache.axis2.databinding.ADBException:
Неожиданный audienceID подэлемента</ns0:faultstring>
** Исключительная ситуация сервера Interact: java.lang.Exception: org.apache.axis2.databinding.
ADBException: Неожиданный audienceID подэлемента в
*** ... com.unicacorp.interact.api.soap.service.v1.xsd.CommandImpl$Factory.parse
(CommandImpl.java:1917) в
```

Используйте следующие действия для записи требования XML `executeBatch ()`. Для этих действий можно использовать значения параметров из требований вызовов API с единственными операциями, но нельзя копировать и вставлять элементы XML.

Процедура

1. Используйте инструмент обработки WSDL (например, SoapUI) для создания правильно построенного требования XML `executeBatch ()` из файла WDSL Interact.
2. Добавьте подкоманды к требованию после определения WSDL для дочерних элементов `executeBatch()`.
3. Допишите аргументы подкоманды после определения WSDL для дочерних элементов `executeBatch()`.

getInstance

Метод `getInstance` создает экземпляр API Interact, который связывается с указанным сервером среды выполнения.

```
getInstance(String URL)
```

Важное замечание: Каждая написанная вами программа, использующая API Interact, должна вызвать `getInstance` для создания экземпляра объекта InteractAPI, отображаемого на сервер среды выполнения, который задан параметром URL.

Для групп серверов, если вы используете балансировщик нагрузки, используйте имя хоста и порт, который вы сконфигурировали для балансировщика нагрузки. Если у вас нет балансировщика нагрузки, надо включить в программу логический механизм для ротации доступных серверов среды выполнения.

Этот метод применим только для сериализации Java с адаптером HTTP. Для SOAP WSDL соответствующий метод не определен. Каждая реализация клиента SOAP устанавливает URL конечной точки своим способом.

- **URL** - Строка, определяющая URL для экземпляра среды выполнения. Например, `http://localhost:7001/Interact/servlet/InteractJSService`.

Возвращаемое значение

Сервер среды выполнения возвращает InteractAPI.

Пример

В следующем примере показано, как создать экземпляр InteractAPI, который указывает на экземпляр сервера среды выполнения, работающий на том же компьютере, что и ваша точка контакта.

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

getOffers

Метод getOffers позволяет затребовать предложения от сервера среды выполнения.

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

- **sessionID** - строка, определяющая текущий сеанс.
- **interactionPoint** - строка, задающая имя точки взаимодействия, на которую ссылается этот метод.

  Примечание: Это имя должно точно соответствовать имени точки взаимодействия, определенной в интерактивном канале.
- **numberOfOffers** - целое число, определяющее число затребованных предложений.

Метод getOffers ожидает срок (в миллисекундах), заданный свойством segmentationMaxWaitTimeInMS, чтобы вся пересегментация завершилась до запуска. Поэтому, если вы вызываете метод postEvent, который запускает пересегментацию, или метод setAudience непосредственно перед вызовом getOffers, возможна задержка.

Возвращаемое значение

Сервер среды выполнения отвечает на getOffers объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

Пример

Этот пример показывает требование единственного предложения для точки взаимодействия Overview Page Banner 1 и способ обработки ответа.

sessionId - это та же строка для идентификации сеанса среды выполнения, что и используемая вызовом startSession, который запустил этот сеанс.

```
String interactionPoint = "Баннер Страницы Обзора 1";
int numberRequested=1;

/** Выполнить вызов */
response = api.getOffers(sessionId, interactionPoint, numberRequested);

/** Обработать ответ соответствующим образом */
 // проверить успешность ответа
 if(response.getStatusCode() == Response.STATUS_SUCCESS)
 {
  System.out.println("Вызов getOffers обработан без предупреждений или ошибок");

        /** Проверка, есть ли другие предложения */
        OfferList offerList=response.getOfferList();

        if(offerList.getRecommendedOffers() != null)
        {
            for(Offer offer : offerList.getRecommendedOffers())
            {
                // печать предложения
                System.out.println("Имя предложения:"+offer.getOfferName());
            }
        }
```

```
        else // строка предложения по умолчанию
        System.out.println("Предложение по умолчанию:"+offerList.getDefaultString());
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов getOffers обработан с предупреждением");
    }
    else
    {
        System.out.println("Вызов getOffers обработан с ошибкой");
    }
    // В случае каких-либо неудач должны быть поясняющие рекомендации
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getOffers",
   response.getAdvisoryMessages());
```

getOffersForMultipleInteractionPoints

Метод `getOffersForMultipleInteractionPoints` позволяет затребовать предложения от сервера среды выполнения для нескольких IP с устранением дублирования.

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

- **sessionID** - строка, определяющая текущий сеанс.
- **requestStr** - строка, задающая массив объектов `GetOfferRequest`.

  Каждый объект `GetOfferRequest` задает:

  – **ipName** - Имя точки взаимодействия (interaction point, IP), для которой объект запрашивает предложения

  – **numberRequested** - Число уникальных предложений, требуемых для указанной IP

  – **offerAttributes** - Требования атрибутов присылаемых предложений, используемых экземпляром `OfferAttributeRequirements`

  – **duplicationPolicy** - ID политики дубликатов для присылаемых предложений

  Политики дубликатов определяют, будут ли возвращаться предложения-дубликаты для различных точек взаимодействия в одном вызове метода. (*В пределах* одной точки взаимодействия предложения-дубликаты никогда не возвращаются.) В настоящее время поддерживаются две политики дубликатов.

  - NO_DUPLICATION (значение ID = 1). Никакое из предложений, включенных в предыдущие экземпляры `GetOfferRequest`, не будут включены в этот экземпляр `GetOfferRequest` (то есть Interact применяет дедупликацию).
  - ALLOW_DUPLICATION (значение ID = 2). Любое из предложений, удовлетворяющих заданным в этом экземпляре `GetOfferRequest` требованиям, будет включено. Предложения, включенные в предыдущие экземпляры `GetOfferRequest`, не будут согласовываться.

  Порядок требований в параметре массива - это порядок приоритетов при представлении предложений.

  Например, предположим, что точки взаимодействия в требовании - IP1, а затем IP2, что предложения-дубликаты запрещены (ID политики дубликатов = 1), и для каждой точки затребованы два предложения. Если Interact обнаруживает предложения A, B и C для IP1 и предложения A и D для IP2, ответ будет содержать предложения A и B для IP1 и только предложение D для IP2.

  Также заметим, что когда ID политики дубликатов - 1, предложения, представленные для точки с более высоким приоритетом, не будут представлены для данной точки.

Метод getOffersForMultipleInteractionPoints ожидает срок (в миллисекундах), заданный свойством segmentationMaxWaitTimeInMS, чтобы вся пересегментация завершилась до запуска. Поэтому, если вы вызываете метод postEvent, который запускает пересегментацию, или метод setAudience непосредственно перед вызовом getOffers, возможна задержка.

Возвращаемое значение

Сервер среды выполнения отвечает на getOffersForMultipleInteractionPoints объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- массив OfferList
- SessionID
- StatusCode

Пример

```
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) {
  // Проверка, есть ли другие предложения
  OfferList[] allOfferLists = response.getAllOfferLists();
  if (allOfferLists != null) {
   for (OfferList ol : allOfferLists) {
    System.out.println
("Следующие предложения предоставляются для точки
               взаимодействия " + ol.getInteractionPointName() + ":");
    for (Offer o : ol.getRecommendedOffers()) {
     System.out.println(o.getOfferName());
    }
   }
  }
 }
 else {
  System.out.println("Вызов метода getOffersForMultipleInteractionPoints()
          возвратил ошибку с кодом: " + response.getStatusCode());
 }
```

Обратите внимание на то, что синтаксис requestStr следующий:

```
требования_для_IP[<требования_для_IP]
```

где

```
<требования_для_IP> = {имя_ip,число_затребованных_предложений_для_этой_ip,
     политика_дубликатов[,дочерние_требования]]}
требования_атрибутов = (число_затребованных_для_этих_требований_атрибутов
     [,требование_атрибута[;требование_отдельного_атрибута])
     [,(требования_атрибутов))
требование_отдельного_атрибута = имя_атрибута=значение_атрибута | тип_атрибута
```

В показанном выше примере requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))}) означает, что для точки взаимодействия с именем IP1 представляется до 5 различных предложений, и они не могут быть представлены ни для какой другой точки взаимодействия в том же вызове метода. Все 5 этих предложений должны иметь числовой атрибут с именем attr1 и значением 1, а также строковый атрибут с именем attr2 и значением *value2*. Из этих 5 предложений максимум 3 должны иметь строковый атрибут с именем attr3 и значением *value3*, и максимум 3 должны иметь числовой атрибут attr4 со значением *4*.

Разрешенные типы атрибутов - числовой, строковый и дата-время, значение атрибута даты-времени должно быть задано в формате ММ/дд/гггг ЧЧ:мм:сс. Для получения возвращенных предложений используйте метод Response.getAllOfferLists (). Чтобы помочь вам понять синтаксис, пример в setGetOfferRequests строит тот же самый экземпляр GetOfferRequests при использовании объектов Java, что предпочтительнее.

getProfile

Метод getProfile позволяет получить профиль и временную информацию о посетителе, посещающем точку контакта.

getProfile(String *sessionID*)

- **sessionID**-строка, определяющая ID сеанса.

Возвращаемое значение

Сервер среды выполнения отвечает на getProfile объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

Пример

Далее приводится пример использования getProfile и способа обработки ответа.

sessionId - это та же строка для идентификации сеанса, что и используемая вызовом startSession, который запустил этот сеанс.

```
response = api.getProfile(sessionId);
/** Обработать ответ соответствующим образом */
    // проверить успешность ответа
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("Вызов getProfile обработан без предупреждений или ошибок");
        // Печатаем профиль - это просто массив объектов NameValuePair
        for(NameValuePair nvp : response.getProfileRecord())
        {
            System.out.println("Имя:"+nvp.getName());
            if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
            {
                System.out.println("Значение:"+nvp.getValueAsDate());
            }
            else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
            {
                System.out.println("Значение:"+nvp.getValueAsNumeric());
```

```
            }
            else
            {
                System.out.println("Значение:"+nvp.getValueAsString());
            }
        }
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов getProfile обработан с предупреждением");
    }
    else
    {
        System.out.println("Вызов getProfile обработан с ошибкой");
    }
    // В случае каких-либо неудач должны быть поясняющие рекомендации
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getProfile",
   response.getAdvisoryMessages());
```

getVersion

Метод `getVersion` возвращает версию текущей реализации сервера среды выполнения Interact.

```
getVersion()
```

Рекомендуется использовать этот метод при инициализации точки контакта с API Interact.

Возвращаемое значение

Сервер среды выполнения отвечает на `getVersion` объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- StatusCode

Пример

Этот пример показывает простой способ вызова `getVersion` и обработки результатов.

```
response = api.getVersion();
/** Обработать ответ соответствующим образом */
    // проверить успешность ответа
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("Вызов getVersion обработан без предупреждений или ошибок");
        System.out.println("Версия API:" + response.getApiVersion(););
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов getVersion обработан с предупреждением");
    }
    else
    {
        System.out.println("Вызов getVersion обработан с ошибкой");
    }

    // В случае каких-либо неудач должны быть поясняющие рекомендации
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getVersion",
   response.getAdvisoryMessages());
```

postEvent

Метод `postEvent` позволяет выполнять любое событие, определенное в интерактивном канале.

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

- **sessionID**: строка, определяющая ID сеанса.
- **eventName**: строка, задающая имя события.

  Примечание: Имя события должно соответствовать имени события, как определено в интерактивном канале. Регистр символов в этом имени не учитывается.
- **eventParameters**. Объекты `NameValuePairImpl`, идентифицирующие параметры, которые требуется передать с этим событием. Эти значения хранятся в данных сеанса.

  Если это событие инициирует пересегментацию, надо убедиться, что все данные, требуемые интерактивными потоковыми диаграммами, доступны в данных сеанса. Если какое-либо из этих значений не было заполнено предшествующими действиями (например, `startSession` или `setAudience` или загрузкой таблицы профилей), надо включить eventParameter для каждого отсутствующего значения. Например, если вы сконфигурировали все таблицы профиля для загрузки в память, надо включить NameValuePair для временных данных, требуемых для интерактивных потоковых диаграмм.

  Если вы используете несколько уровней аудитории, скорее всего у вас есть различные наборы eventParameters для каждого уровня аудитории. Надо включать некоторую логику, чтобы обеспечить выбор правильного набора параметров для уровня аудитории.

  Важное замечание: Если это событие записывается в хронологию ответов, надо передать код процедуры для предложения. Надо задать имя для NameValuePair как `"UACIOfferTrackingCode"`.

  Для каждого события можно передать только один код обработки. Если вы не передаете код процедуры для контакта предложения, Interact записывает в журнал контакт предложения для каждого предложения в последнем рекомендованном списке предложений. Если вы не передаете код процедуры для ответа, Interact возвращает ошибку.
- Есть несколько других зарезервированных параметров, используемых с `postEvent`, и другие методы; они обсуждаются далее в этом разделе.

Любое требование пересегментации или записи в хронологию контактов или ответов не ожидает ответа.

Пересегментация не очищает результаты предыдущей сегментации для текущего уровня аудитории. Можно использовать параметр `UACIExecuteFlowchartByName`, чтобы задать конкретные потоковые диаграммы для выполнения. Метод `getOffers` ожидает завершения пересегментации перед выполнением. Поэтому, если вы вызываете метод `postEvent`, который запускает пересегментацию, непосредственно перед вызовом `getOffers`, возможна задержка.

Возвращаемое значение

Сервер среды выполнения отвечает на `postEvent` объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Пример

Следующий пример `postEvent` показывает отправку новых параметров для события, инициировавшего пересегментацию, и способ обработки ответа.

`sessionId` - это та же строка для идентификации сеанса, что и используемая вызовом `startSession`, который запустил этот сеанс.

```
String eventName = "SearchExecution";

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

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

NameValuePair parmB3 = new NameValuePairImpl();
parmB3.setName("Браузер");
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
};

/** Выполнить вызов */
response = api.postEvent(sessionId, eventName, postEventParameters);

/** Обработать ответ соответствующим образом */
    // проверить успешность ответа
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("Вызов postEvent обработан без предупреждений или ошибок");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов postEvent обработан с предупреждением");
```

```
}
else
{
    System.out.println("Вызов postEvent обработан с ошибкой");
}

// В случае каких-либо неудач должны быть поясняющие рекомендации
if(response.getStatusCode() != Response.STATUS_SUCCESS)
    printDetailMessageOfWarningOrError("postEvent",
response.getAdvisoryMessages());
```

setAudience

Метод setAudience позволяет задавать ID и уровень аудитории для посетителя.

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

- **sessionID** - строка, определяющая ID сеанса.
- **audienceID** - массив объектов NameValuePairImpl, которые определяют уровень аудитории.
- **audienceLevel** - строка, определяющая уровень аудитории.
- **parameters** - объекты NameValuePairImpl, идентифицирующие параметры, которые требуется передать с setAudience. Эти значения хранятся в данных сеанса и могут использоваться для сегментации.

  У вас должно быть значение для каждого столбца в вашем профиле. Это надмножество всех столбцов во всех таблицах, определенных для интерактивного канала и любых данных реального времени. Если вы уже заполнили все данные сеанса с помощью startSession или postEvent, нет необходимости посылать новые параметры.

Метод setAudience запускает пересегментацию. Метод getOffers ожидает завершения пересегментации перед выполнением. Поэтому, если вы вызываете метод setAudience непосредственно перед вызовом getOffers, возможна задержка.

Метод setAudience также загружает данные профиля для ID аудитории. Можно использовать метод setAudience для принудительной перезагрузки тех же данных профиля, которые загружены методом startSession.

Возвращаемое значение

Сервер среды выполнения отвечает на setAudience объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Пример

Для этого примера уровень аудитории остается тем же, но ID меняется, как при случае, когда анонимный пользователь входит в систему и становится известным.

sessionId и audienceLevel - те же самые строки идентификации сеанса и уровня аудитории, что и использовавшиеся вызовом startSession при запуске сеанса.

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

```
NameValuePair[] newAudienceId = { custId2 };

/** Можно передавать также параметры. Для этого примера параметров нет,
* так что передается null */
NameValuePair[] noParameters=null;

/** Выполнить вызов */
response = api.setAudience(sessionId, newAudienceId, audienceLevel, noParameters);

 /** Обработать ответ соответствующим образом */
    // проверить успешность ответа
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("Вызов setAudience обработан без предупреждений или ошибок");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов setAudience обработан с предупреждением");
    }
    else
    {
        System.out.println("Вызов setAudience обработан с ошибкой");
    }

    // В случае каких-либо неудач должны быть поясняющие рекомендации
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setAudience",
   response.getAdvisoryMessages());
```

setDebug

Метод `setDebug` позволяет задать уровень подробности записи в журнал для всех путей кода в сеансе.

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

- **sessionID** - строка, определяющая ID сеанса.
- **debug** - логическое значение, которое включает или выключает запись отладочной информации. Допустимые значения: `true` и `false`. Если задано true, Interact записывает отладочную информацию в журналы сервера среды выполнения.

Возвращаемое значение

Сервер среды выполнения отвечает на `setDebug` объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Пример

В следующем примере показано изменение уровня отладки сеанса.

`sessionId` - это та же строка для идентификации сеанса, что и используемая вызовом `startSession`, который запустил этот сеанс.

```
boolean newDebugFlag=false;
/** выполняем вызов */
response = api.setDebug(sessionId, newDebugFlag);

/** Обработать ответ соответствующим образом */
    // проверить успешность ответа
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("Вызов setDebug обработан без предупреждений или ошибок");
    }
```

```
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов setDebug обработан с предупреждением");
    }
    else
    {
        System.out.println("Вызов setDebug обработан с ошибкой");
    }

    // В случае каких-либо неудач должны быть поясняющие рекомендации
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setDebug",
   response.getAdvisoryMessages());
```

startSession

Метод startSession создает и определяет сеанс среды выполнения.

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

startSession может инициировать до пяти действий:

- создать сеанс среды выполнения.
- загрузить в сеанс выполнения данные профиля посетителя для текущего уровня аудитории, в том числе любые таблицы измерений, помеченные для загрузки в отображении таблиц, которое определено для интерактивного канала.
- запустить сегментацию, выполняя все интерактивные потоковые диаграммы для текущего уровня аудитории.
- загрузить в сеанс данные подавления предложений, если для свойства enableOfferSuppressionLookup задано значение true.
- загрузить в сеанс данные переопределения оценок, если для свойства enableScoreOverrideLookup задано значение true.

Метод startSession требует следующих параметров:

- **sessionID**-строка, определяющая ID сеанса. Надо задать ID сеанс. Например, вы могли использовать сочетание ID покупателя и отметки времени.

  Чтобы определить, что составляет сеанс выполнения, надо указать ID сеанса. Этим значением управляет клиент. Все вызовы методов для одного и того же ID сеанса должны быть синхронизированы клиентом. Поведение для одновременных вызовов API с одним и тем ID сеанса не определено.
- **relyOnExistingSession** - логическое значение, определяющее, использует ли данный сеанс новый или же существующий сеанс. Допустимые значения: true и false. При значении true в методе startSession надо задать ID существующего сеанса. При значении false надо задать новый ID сеанса.

  Если вы задали для relyOnExistingSession значение true, и сеанс существует, среда выполнения использует данные существующего сеанса, не перезагружает никакие данные и не запускает сегментацию. Если этот сеанс не существует, среда выполнения создает новый сеанс, загружает данные и запускает сегментацию. Задание для relyOnExistingSession значения true и его использование для всех вызовов startSession полезно, если длительность сеанса для вашей точки контакта больше, чем длительность сеанса среды выполнения. Например, сеанс сайта длится 2 часа, но сеанс среды выполнения длится только 20 минут.

Если вы вызываете startSession дважды с одним и тем же самым ID сеанса, при значении relyOnExistingSession false все данные сеанса первого вызова startSession теряются.

- **debug** - логическое значение, которое включает или выключает запись отладочной информации. Допустимые значения: true и false. Если задано true, Interact записывает отладочную информацию в журналы сервера среды выполнения. Флаг отладки задается для каждого сеанса отдельно. Таким образом, вы можете отслеживать данные отладки для отдельного сеанса.
- **interactiveChannel** - строка, определяющая имя интерактивного канала, к которому относится данный сеанс. Это имя должно точно соответствовать имени интерактивного канала, определенного в Campaign.
- **audienceID** - массив объектов NameValuePairImpl, в котором имена должны соответствовать именам физических столбцов любой таблицы, содержащей ID аудитории.
- **audienceLevel** - строка, определяющая уровень аудитории.
- **parameters** - объекты NameValuePairImpl, идентифицирующие параметры, которые требуется передать с startSession. Эти значения хранятся в данных сеанса и могут использоваться для сегментации.

  Если у вас есть несколько интерактивных потоковых диаграмм для одного и того же уровня аудитории, надо включить надмножество всех столбцов во всех таблицах. Если вы конфигурируете среду выполнения для загрузки таблицы профилей, и эта таблица профилей содержит все нужные вам столбцы, не требуется передавать никакие параметры, если только вы не хотите перезаписать данные в таблице профилей. Если ваша таблица профилей содержит только подмножество необходимых столбцов, надо включить недостающие столбцы как параметры.

Если audienceID или audienceLevel недопустимы и значение relyOnExistingSession - false, вызов startSession завершается неудачно. Если interactiveChannel недопустим, startSession завершается неудачно, независимо от того, какое значение задано для relyOnExistingSession - true или false.

Если для relyOnExistingSession задано значение true, и вы делаете второй вызов startSession с тем же sessionID, а срок ожидания для первого сеанса истек, Interact создает новый сеанс.

Если для relyOnExistingSession задано значение true, и вы делаете второй вызов startSession с тем же sessionID, но с другим audienceID или audienceLevel, сервер среды выполнения меняет аудиторию для существующего сеанса.

Если для relyOnExistingSession задано значение true, и вы делаете второй вызов startSession с тем же sessionID, но с другим interactiveChannel, сервер среды выполнения создает новый сеанс.

Возвращаемое значение

Сервер среды выполнения отвечает на startSession объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages (если StatusCode не равен 0)
- ApiVersion
- SessionID
- StatusCode

Пример

В следующем примере показан возможный способ вызова startSession.

```
String sessionId="MySessionID-123";
String audienceLevel="Покупатель";
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("Отметка времени");
parm2.setValueAsDate(new Date());
parm2.setValueDataType(NameValuePair.DATA_TYPE_DATETIME);

NameValuePair parm3 = new NameValuePairImpl();
parm3.setName("Браузер");
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);

/** Задание параметров (не обязательно) */
NameValuePair[] initialParameters = { parm1,
 parm2,
 parm3,
 parm4,
 parm5,
 parm6
 };

/** Выполнить вызов */
response = api.startSession(sessionId, relyOnExistingSession, initialDebugFlag,
 interactiveChannel, initialAudienceId, audienceLevel, initialParameters);

/** Обработать ответ соответствующим образом */
processStartSessionResponse(response);
```

processStartSessionResponse - это метод обработки объекта ответа, возвращенного startSession.

```
public static void processStartSessionResponse(Response response)
{
  // проверить успешность ответа
  if(response.getStatusCode() == Response.STATUS_SUCCESS)
```

```
 {
  System.out.println("Вызов startSession обработан без предупреждений или ошибок");
 }
  else if(response.getStatusCode() == Response.STATUS_WARNING)
 {
  System.out.println("Вызов startSession обработан с предупреждением");
 }
  else
 {
  System.out.println("Вызов startSession обработан с ошибкой");
 }

 // В случае каких-либо неудач должны быть поясняющие рекомендации
 if(response.getStatusCode() != Response.STATUS_SUCCESS)
  printDetailMessageOfWarningOrError("StartSession",
   response.getAdvisoryMessages());
}
```

Дедупликация предложений по атрибутам предложений

При использовании интерфейса прикладного программирования (application programming interface, API) Interact предложения доставляют два вызова API: `getOffers` и `getOffersForMultipleInteractionPoints`. Вызов `getOffersForMultipleInteractionPoints` позволяет не допустить возврата дубликатов предложений на уровне *OfferID*, но не может дедуплицировать предложения в категории предложений. Поэтому, например, для Interact, чтобы возвратить только одно предложение из каждой категории предложений, ранее требовался обходной путь. За счет добавления двух параметров в вызов API `startSession` теперь стала возможна дедупликация предложений по другим атрибутам предложений, например, категории.

Этот список содержит сводку параметров, которые были добавлены к вызову API `startSession`. Дополнительную информацию об этих параметрах и о любых аспектах API Interact смотрите в *Руководстве администратора IBM Interact* или в файлах Javadoc, включенных в вашу установку Interact в `<домашний_каталог_Interact>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Чтобы создать вызов API `startSession` с дедупликацией предложений, так что последующие вызовы `getOffer` возвращали только по одному предложению из каждой категории, надо включить параметр `UACIOfferDedupeAttribute` как часть вызова API. Параметр можно задать в формате `имя,значение,тип`, как показано здесь:

  `UACIOfferDedupeAttribute,<имя_атрибута>,string`

  В этом примере вам надо заменить *<имя_атрибута>* на имя атрибута предложения, который вы хотите использовать в качестве критерия дедупликации, например, Category.

  Примечание: Interact проверяет предложения с одинаковыми значениями заданного вами атрибута и удаляет все предложения, кроме предложения с максимальной оценкой. Если у предложений с атрибутом-дубликатом идентичные оценки, Interact возвращает случайный выбор из соответствующих предложений.

- `UACINoAttributeDedupeIfFewerOf`. Когда вы включаете `UACIOfferDedupeAttribute` в вызов `startSession`, можно также задать параметр `UACINoAttributeDedupeIfFewerOf`, чтобы определить поведение в случаях, когда список предложений после дедупликации больше не содержит достаточно предложений для удовлетворения исходного требования.

Например, если вы задали UACIOfferDedupeAttribute, чтобы использовать категорию предложения для дедупликации предложений, и ваш последующий вызов getOffers требует возврата восьми предложений, в результате дедупликации пригодных предложений может стать меньше восьми. В этом случае задание для параметра UACINoAttributeDedupeIfFewerOf значения true привело бы к добавлению некоторых дубликатов в список пригодных предложений, чтобы соблюсти требуемое число предложений. В этом примере, если вы задали для параметра значение false, число возвращенных предложений было бы меньше, чем требуемое.

По умолчанию в качестве значения UACINoAttributeDedupeIfFewerOf задается true.

Например, предположим, что вы задали в параметрах startSession, что критерием дедупликации служит категория предложения, как показано здесь:

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

Эти параметры вместе приводят к тому, что Interact дедуплицирует предложения на основе атрибута предложения "Category" и возвращает только предложения без дубликатов, даже если полученных предложений меньше, чем затребовано (для UACINoAttributeDedupeIfFewerOffer задано false).

Когда вы выполняете вызов API getOffers, исходный набор пригодных предложений может включать в себя следующие предложения:

- Category=Электроника: Предложение A1 с оценкой 100 и Предложение A2 с оценкой 50.
- Category=Смартфоны: Предложение B1 с оценкой 100, Предложение B2 с оценкой 80 и Предложение B3 с оценкой 50.
- Category=MP3-плееры: Предложение C1 с оценкой 100, Предложение C2 с оценкой 50.

В данном случае есть два предложения-дубликата в первой категории, три предложения-дубликата во второй категории и два предложения-дубликата в третьей категории. Возвращаются предложения с максимальной оценкой в каждой категории, то есть Предложение A1, Предложение B1 и Предложение C1.

Если в вызове API getOffers затребовано шесть предложений, в этом наборе примера для UACINoAttributeDedupeIfFewerOffer задано false, таким образом были бы возвращены только три предложения.

Если вызов API getOffers требует шесть предложений, и в этом примере параметр UACINoAttributeDedupeIfFewerOffer опущен или задан как true, некоторые предложения-дубликаты были бы включены в результат, чтобы обеспечить требуемое количество.

Зарезервированные параметры

Есть несколько зарезервированных параметров, используемых с API Interact. Некоторые требуются для сервера среды выполнения, а других можно использовать для дополнительных возможностей.

Возможности postEvent

Возможность	Параметр	Описание
Запись в пользов. таблицу	UACICustomLoggerTableName	Имя таблицы в источнике данных таблиц среды выполнения. Если вы задаете этот параметр с допустимым именем таблицы, среда выполнения записывает все данные сеанса в выбранную таблицу. Все имена столбцов в этой таблице, соответствующие NameValuePair данных сеанса, заполняются. Среда выполнения заполняет любой столбец, не соответствующей паре значение-имя в сеансе, значением null. Процессом записи в базу данных можно управлять при помощи свойства конфигурации customLogger.
Несколько типов ответов	UACILogToLearning	Целое со значением 1 или 0. 1 указывает, что среда выполнения должна записать в журнал событие как принятие для системы обучения или включить подавление предложений в сеансе. 0 указывает, что среда выполнения не должна записывать в журнал событие для системы обучения или включать подавление предложений в сеансе. Этот параметр позволяет создать несколько различных методов postEvent, записывающих в журнал различные типы ответов, не влияя при этом на обучение. Не требуется определять этот параметр для набора событий, чтобы записывать в журнал контакт, принятие или отклонение. Этот параметр надо использовать вместе с UACIResponseTypeCode. Если вы не определяете UACILOGTOLEARNING, среда выполнения предполагает значение по умолчанию 0 (кроме случая, когда событие инициировало запись в журнал контакта, принятия или отклонения).
	UACIResponseTypeCode	Значение, представляющее код типа ответа. Значение должно быть допустимой записью в таблице UA_UsrResponseType
Отслеживание ответов	UACIOfferTrackingCode	Код процедуры для предложения. Этот параметр надо определить, если событие записывается в журнал хронологии контактов или ответов. Для каждого события можно передать только один код процедуры. Если вы не передаете код процедуры для контакта предложения, среда выполнения записывает в журнал контакт предложения для каждого предложения в последнем рекомендуемом списке предложений. Если вы не передаете код процедуры для ответа, среда выполнения возвращает ошибку. Если вы сконфигурировали отслеживание межсеансовых ответов, можно использовать параметр UACIOfferTrackingcodeType для определения, какой тип кода отслеживания вы используете помимо кода процедуры.

Возможность	Параметр	Описание
Межсеансовое отслеживание ответов	UACIOfferTrackingCodeType	Число, определяющее тип кода отслеживания. 1 - код процедуры по умолчанию, 2 - код предложения. Все коды должны быть допустимыми записями в таблице UACI_TrackingType. В эту таблицу можно добавить другие, пользовательские коды.
Выполнение определенной потоковой диаграммы	UACIExecuteFlowchartByName	Если вы определяете этот параметр для какого-либо метода, запускающего сегментацию (startSession, setAudience или postEvent, запускающий пересегментацию), вместо запуска всех потоковых диаграмм для текущего уровня аудитории Interact запускает только указанные потоковые диаграммы. Можно задать список потоковых диаграмм, разделенных символом (|).

Зарезервированные параметры среды выполнения

Среда выполнения использует следующие зарезервированные параметры. Не используйте эти имена для своих параметров событий.

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

О классе AdvisoryMessage

Класс advisoryMessage содержит методы, которые определяют объект сообщения рекомендации. Объект сообщения рекомендации содержится в объекте Response. Каждый метод в InteractAPI возвращает объект Response. (Исключение - метод executeBatch, который возвращает объект batchResponse.)

В случае ошибки или предупреждения сервер Interact заполняет объект сообщения рекомендации. Объект сообщения рекомендации содержит следующие атрибуты:

- **DetailMessage** - подробное описание сообщения рекомендации. Этот атрибут может быть доступен не для всех сообщений рекомендации. Если атрибут DetailMessage доступен. он может не быть локализован.
- **Message** - краткое описание сообщения рекомендации.
- **MessageCode** - числовой код сообщения рекомендации.
- **StatusLevel** - числовой код серьезности сообщения рекомендации.

Вы получаете объекты advisoryMessage при помощи метода getAdvisoryMessages.

getDetailMessage

Метод getDetailMessage возвращает подробное детализированное описание объекта сообщения рекомендации. Не для всех сообщений есть детализированное сообщение.

getDetailMessage()

Возвращаемое значение

Объект сообщения рекомендации возвращает строку.

Пример

```
// В случае каких-либо неудач должны быть поясняющие рекомендации
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
 for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
  System.out.println(msg.getMessage());
  // Некоторые сообщения рекомендации могут иметь дополнительные подробности:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessage

Метод getMessage возвращает краткое описание объекта сообщения рекомендации.

getMessage()

Возвращаемое значение

Объект сообщения рекомендации возвращает строку.

Пример

Следующий метод выводит сообщение и подробное сообщение объекта AdvisoryMessage.

```
// В случае каких-либо неудач должны быть поясняющие рекомендации
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
 for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
  System.out.println(msg.getMessage());
  // Некоторые сообщения рекомендации могут иметь дополнительные подробности:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessageCode

Метод getMessageCode возвращает внутренний код ошибки, связанный с объектом сообщения рекомендации, если уровень состояния - 2 (STATUS_LEVEL_ERROR).

getMessageCode()

Возвращаемое значение

Объект AdvisoryMessage возвращает целое число.

Пример

Следующий метод выводит код сообщения объекта AdvisoryMessage.

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

getStatusLevel

Метод `getStatusLevel` возвращает уровень состояния объекта сообщения рекомендации.

```
getStatusLevel()
```

Возвращаемое значение

Объект сообщения рекомендации возвращает целое число.

- 0 - STATUS_LEVEL_SUCCESS - Вызванный метод завершился без ошибок.
- 1 - STATUS_LEVEL_WARNING - Вызванный метод завершился по крайней мере с одним предупреждением (но без ошибок).
- 2 - STATUS_LEVEL_ERROR - Вызванный метод не завершился успешно, и есть по крайней мере одно сообщение об ошибке.

Пример

Следующий метод выводит уровень состояния объекта AdvisoryMessage.

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

О классе AdvisoryMessageCode

Класс `advisoryMessageCode` содержит методы, которые определяют коды сообщений рекомендации. Коды сообщений рекомендации получают при помощи метода `getMessageCode`.

Коды сообщений рекомендации

Вы получаете коды сообщений рекомендации с помощью метода `getMessageCode`.

Эта таблица содержит список кодов сообщений рекомендации и их описаний.

Код	Текст сообщения	Описание
1	INVALID_SESSION_ID	ID сеанса не относится к допустимому сеансу.
2	ERROR_TRYING_TO_ABORT_ SEGMENTATION	Произошла ошибка при попытке прервать сегментацию во время вызова `endSession`.
3	INVALID_INTERACTIVE_CHANNEL	Аргумент, переданный для интерактивного канала, не ссылается на допустимый интерактивный канал.
4	INVALID_EVENT_NAME	Аргумент, переданный для события, не ссылается на допустимое событие для текущего интерактивного канала.
5	INVALID_INTERACTION_POINT	Аргумент, переданный для точки взаимодействия, не ссылается на допустимую точку взаимодействия для текущего интерактивного канала.
6	ERROR_WHILE_MAKING_INITIAL_ SEGMENTATION_REQUEST	Ошибка при передаче требования сегментации.
7	SEGMENTATION_RUN_FAILED	Сегментация частично обработана, но привела к ошибке.

Код	Текст сообщения	Описание
8	PROFILE_LOAD_FAILED	Попытка загрузить таблицы профилей или измерений завершилась неудачно.
9	OFFER_SUPPRESSION_LOAD_FAILED	Попытка загрузить таблицу подавления предложений завершилась неудачно.
10	COMMAND_METHOD_ UNRECOGNIZED	Метод команды, заданный для команды в вызове `executeBatch`, недопустим.
11	ERROR_TRYING_TO_POST_EVENT _PARAMETERS	Произошла ошибка при отправке параметров события.
12	LOG_SYSTEM_EVENT_EXCEPTION	Исключительная ситуация при попытке передать системное событие (Окончание сеанса, Получить предложение, Получить профиль, Задать аудиторию, Задать отладку или Запустить Сеанс) для записи в журнал.
13	LOG_USER_EVENT_EXCEPTION	Исключительная ситуация при попытке передать пользовательское событие для записи в журнал.
14	ERROR_TRYING_TO_LOOK_UP_EVENT	Ошибка при попытке поиска имени события.
15	ERROR_TRYING_TO_LOOK_UP _INTERACTIVE_CHANNEL	Ошибка при попытке поиска имени интерактивного канала.
16	ERROR_TRYING_TO_LOOK_UP _INTERACTION_POINT	Ошибка при попытке поиска имени точки взаимодействия.
17	RUNTIME_EXCEPTION_ ENCOUNTERED	Возникла непредвиденная исключительная ситуация времени выполнения.
18	ERROR_TRYING_TO_EXECUTE_ ASSOCIATED_ACTION	Произошла ошибка при попытке выполнить связанное действие (Инициировать пересегментацию, Запись контакта предложения, Запись восприятия предложения или Запись отклонения предложения).
19	ERROR_TRYING_RUN_FLOWCHART	Ошибка при попытке запустить потоковую диаграмму.
20	FLOWCHART_FAILED	Запуск поточной диаграммы завершился неудачно.
21	FLOWCHART_ABORTED	Выполнение потоковой диаграммы прервано.
22	FLOWCHART_NEVER_RUN	Указанная потоковая диаграмма никогда не выполнялась.
23	FLOWCHART_STILL_RUNNING	Потоковая диаграмма еще выполняется.
24	ERROR_WHILE_READING_ PARAMETERS	Произошла ошибка при чтении параметров.
25	ERROR_WHILE_LOADING_ RECOMMENDED_OFFERS	Ошибка при загрузке рекомендуемых предложений
26	ERROR_WHILE_LOGGING_DEFAULT _TEXT_STATISTICS	Ошибка при записи в журнал текстовой статистики по умолчанию (выводится число раз строки по умолчанию для точки взаимодействия).
27	SCORE_OVERRIDE_LOAD_FAILED	Не удалось загрузить таблицу переопределения оценки.

Код	Текст сообщения	Описание
28	NULL_AUDIENCE_ID	Идентификатор аудитории пуст.
29	UNRECOGNIZED_AUDIENCE_LEVEL	Был задан нераспознанный уровень аудитории.
30	MISSING_AUDIENCE_FIELD	Поле аудитории отсутствует.
31	INVALID_AUDIENCE_FIELD_TYPE	Указан недопустимый тип поля аудитории.
32	UNSUPPORTED_AUDIENCE_FIELD _TYPE	Неподдерживаемый тип поля аудитории
33	TIMEOUT_REACHED_ON_GET_ OFFERS_CALL	Для вызова `getOffers` истек срок ожидания, но предложения не возвращены.
34	INTERACT_INITIALIZATION_NOT_ COMPLETED_SUCCESSFULLY	Инициализация сервера среды выполнения не завершилась успешно.
35	SESSION_ID_UNDEFINED	Идентификатор сеанса не определен.
36	INVALID_NUMBER_OF_OFFERS_ REQUESTED	Было затребовано недопустимое число предложений.
37	NO_SESSION_EXIST_BUT_WILL_ CREATE_NEW_ONE	Сеансов не существует, но один был создан.
38	AUDIENCE_ID_NOT_FOUND_IN_ PROFILE_TABLE	Заданный идентификатор аудитории отсутствует в таблице профилей.
39	LOG_CUSTOM_LOGGER_EVENT_ EXCEPTION	Исключительная ситуация при попытке передать пользовательское событие данных записи в журнал.
40	SPECIFIED_FLOWCHART_FOR_ EXECUTION_DOES_NOT_EXIST	Нельзя запустить указанную потоковую диаграмму, так как она не существует.
41	AUDIENCE_NOT_DEFINED_IN_ CONFIGURATION	Указанная аудитория не определена в конфигурации.

О классе BatchResponse

Класс `BatchResponse` содержит методы, определяющие результаты метода `executeBatch`.

Объект BatchResponse содержит следующие атрибуты:

- **BatchStatusCode**- Максимальное значение кода состояния для всех ответов, затребованных методом `executeBatch`.
- **Responses** - массив объектов Response, затребованных методом `executeBatch`.

getBatchStatusCode

Метод `getBatchStatusCode` возвращает максимальный код состояния из массива команд, выполненных методом `executeBatch`.

```
getBatchStatusCode()
```

Возвращаемое значение

Метод `getBatchStatusCode` возвращает целое число.

- 0 - STATUS_SUCCESS - вызванный метод завершился без ошибок.

- 1 - STATUS_WARNING - Вызванный метод завершился по крайней мере с одним предупреждением (но без ошибок).
- 2 - STATUS_ERROR - Вызванный метод не завершился успешно, и есть по крайней мере одно сообщение об ошибке.

Пример

Следующий код - пример получения BatchStatusCode.

```
// Код состояния высшего уровня - простой способ определить, были ли
// неудачи в массиве объектов Response
if(batchResponse.getBatchStatusCode() == Response.STATUS_SUCCESS)
{
  System.out.println("ExecuteBatch отработал прекрасно!");
}
else if(batchResponse.getBatchStatusCode() == Response.STATUS_WARNING)
{
  System.out.println("Вызов ExecuteBatch обработан хотя бы с одним предупреждением");
}
else
{
  System.out.println("Вызов ExecuteBatch обработан хотя бы с одной ошибкой");
}

// Итерация по массиву и печать сообщения для любой неудачи
for(Response response : batchResponse.getResponses())
{
  if(response.getStatusCode() != Response.STATUS_SUCCESS)
  {
      printDetailMessageOfWarningOrError("executeBatchCommand",
   response.getAdvisoryMessages());
  }
}
```

getResponses

Метод getResponses возвращает максимальный код состояния из массива объектов ответа, соответствующих массиву команд, которые выполнены методом executeBatch.

```
getResponses()
```

Возвращаемое значение

Метод getResponses возвращает массив объектов Response.

Пример

Следующий пример выбирает все ответы и выводит любые сообщения рекомендации, если команда завершилась неудачно.

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

О командном интерфейсе

Метод executeBatch требует передачи массива объектов, реализующего командный интерфейс. Для передачи объектов команд воспользуйтесь реализацией по умолчанию, CommandImpl.

В следующей таблице перечислены команды, методы класса InteractAPI, которые каждая команда представляет, и методы командного интерфейса, которые надо использовать для каждой команды. Указывать ID сеанса не требуется, поскольку метод executeBatch уже содержит ID сеанса.

Команда	Метод API Interact	Методы командного интерфейса
COMMAND_ENDSESSION	endSession	Нет.
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	Нет.
COMMAND_GETVERSION	getVersion	Нет.
COMMAND_POSTEVENT	postEvent	• setEvent • setEventParameters
COMMAND_SETAUDIENCE	setAudience	• setAudienceID • setAudienceLevel • setEventParameters
COMMAND_SETDEBUG	setDebug	setDebug
COMMAND_STARTSESSION	startSession	• setAudienceID • setAudienceLevel • setDebug • setEventParameters • setInteractiveChannel • setRelyOnExistingSession

setAudienceID

Метод setAudienceID определяет AudienceID для команд setAudience и startSession.

setAudienceID(*audienceID*)

- **audienceID** - массив объектов NameValuePair, который определяет ID аудитории.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода executeBatch, вызывающего startSession и setAudience.

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

```
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceID(newAudienceId);
. . .
/** Построить массив команд */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Выполнить вызов */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Обработать ответ соответствующим образом */
    processExecuteBatchResponse(batchResponse);
```

setAudienceLevel

Метод setAudienceLevel определяет уровень аудитории для команд setAudience и startSession.

setAudienceLevel(*audienceLevel*)

- *audienceLevel* - строка, содержащая уровень аудитории.

  Важное замечание: Имя *audienceLevel* должно точно соответствовать имени аудитории, как определено в Campaign. Регистр символов учитывается.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода executeBatch, вызывающего startSession и setAudience.

```
String audienceLevel="Покупатель";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceLevel(audienceLevel);
. . .
/** Построить массив команд */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Выполнить вызов */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Обработать ответ соответствующим образом */
    processExecuteBatchResponse(batchResponse);
```

setDebug

Метод setDebug определяет уровень отладки для команды startSession.

setDebug(*debug*)

Если задано `true`, сервер среды выполнения записывает отладочную информацию в журналы сервера среды выполнения. Если задано `false`, сервер среды выполнения не записывает никакой отладочной информации. Флаг отладки задается для каждого сеанса отдельно. Таким образом, вы можете отслеживать данные отладки для отдельного сеанса выполнения.

- **debug** - логическое значение (`true` или `false`).

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода `executeBatch`, вызывающего `startSession` и `setDebug`.

```
boolean initialDebugFlag=true;
boolean newDebugFlag=false;
. . .
/* строим команду startSession */
Command startSessionCommand = new CommandImpl();
startSessionCommand.setDebug(initialDebugFlag);
. . .

/* строим команду setDebug */
Command setDebugCommand = new CommandImpl();
setDebugCommand.setMethodIdentifier(Command.COMMAND_SETDEBUG);
setDebugCommand.setDebug(newDebugFlag);

/** Построить массив команд */
Command[] commands =
    {
        startSessionCommand,
        setDebugCommand,
    };
/** Выполнить вызов */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Обработать ответ соответствующим образом */
    processExecuteBatchResponse(batchResponse);
```

setEvent

Метод `setEvent` определяет имя события, используемое командой `postEvent`.

`setEvent(`*`event`*`)`

- **event** - Строка, содержащая имя события.

Важное замечание: Имя *event* должно точно соответствовать имени события, как определено в интерактивном канале. Регистр символов учитывается.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода `executeBatch`, вызывающего `postEvent`.

```
String eventName = "SearchExecution";

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

setEventParameters

Метод `setEventParameters` определяет параметры события, используемые командой `postEvent`. Эти значения хранятся в данных сеанса.

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

- **eventParameters** - массив объектов NameValuePair, определяющих параметры события.

Например, если событие - это регистрация предложения в хронологии контактов, надо включать код обработки предложения.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода `executeBatch`, вызывающего `postEvent`.

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

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

NameValuePair parmB3 = new NameValuePairImpl();
parmB3.setName("Браузер");
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

Метод **setGetOfferRequests** задает параметр для получения предложений, используемых командой getOffersForMultipleInteractionPoints.

setGetOfferRequests(*numberRequested*)

- **numberRequested** - массив объектов GetOfferRequest, определяющих параметр для получения предложений.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода GetOfferRequest, вызывающего setGetOfferRequests.

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

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

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

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

setInteractiveChannel

Метод setInteractiveChannel определяет имя интерактивного канала, используемого командой startSession.

setInteractiveChannel(*interactiveChannel*)

- **interactiveChannel** - строка, содержащая имя интерактивного канала.

Важное замечание: *interactiveChannel* должно точно соответствовать имени интерактивного канала, определенного в Campaign. Регистр символов учитывается.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода executeBatch, вызывающего startSession.

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

setInteractionPoint

Метод setInteractionPoint определяет имя точки взаимодействия, используемой командами getOffers и postEvent.

setInteractionPoint(*interactionPoint*)

- **interactionPoint** - строка, содержащая имя точки взаимодействия.

Важное замечание: *interactionPoint* должно точно соответствовать имени точки взаимодействия, определенному в интерактивном канале. Регистр символов учитывается.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода executeBatch, вызывающего getOffers.

```
String interactionPoint = "Баннер Страницы Обзора 1";
int numberRequested=1;

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

setMethodIdentifier

Метод setMethodIdentifier определяет тип команды, содержащейся в объекте команды.

setMethodIdentifier(*methodIdentifier*)

- **methodIdentifier** - строка, содержащая тип команды.

  Допустимые значения:

  – **COMMAND_ENDSESSION**-представляет метод endSession.

- **COMMAND_GETOFFERS**-представляет метод getOffers.
- **COMMAND_GETPROFILE**-представляет метод getProfile.
- **COMMAND_GETVERSION**-представляет метод getVersion.
- **COMMAND_POSTEVENT**-представляет метод postEvent.
- **COMMAND_SETAUDIENCE**-представляет метод setAudience.
- **COMMAND_SETDEBUG**-представляет метод setDebug.
- **COMMAND_STARTSESSION**-представляет метод startSession.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода executeBatch, вызывающего getVersion и endSession.

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

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

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

setNumberRequested

Метод setNumberRequested определяет число предложений, затребованных командой getOffers.

setNumberRequested(*numberRequested*)

- **numberRequested** - целое число, определяющее число предложений, которые затребованы командой getOffers.

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода executeBatch, вызывающего getOffers.

```
String interactionPoint = "Баннер Страницы Обзора 1";
int numberRequested=1;

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

setRelyOnExistingSession

Метод setRelyOnExistingSession определяет логическое значение, задающее, использует ли команда startSession существующий сеанс или нет.

setRelyOnExistingSession(*relyOnExistingSession*)

При значении true ID сеанса для executeBatch должен соответствовать ID существующего сеанса. При значении false надо задать новый ID сеанса с методом executeBatch.

- **relyOnExistingSession** - логическое значение (true или false).

Возвращаемое значение

Нет.

Пример

Следующий пример - это отрывок из метода executeBatch, вызывающего startSession.

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

Об интерфейсе NameValuePair

Многие методы в API Interact либо возвращают объекты NameValuePair, либо требуют передать объекты NameValuePair в качестве аргументов. При передаче в качестве аргументов в метод воспользуйтесь реализацией по умолчанию - NameValuePairImpl.

getName

Метод getName возвращает компонент имени объекта NameValuePair.

```
getName()
```

Возвращаемое значение

Метод getName возвращает строку.

Пример

Следующий пример - это отрывок из метода, обрабатывающего объект ответа для getProfile.

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

getValueAsDate

Метод getValueAsDate возвращает значение объекта NameValuePair.

```
getValueAsDate()
```

Надо использовать getValueDataType перед использованием getValueAsDate, чтобы подтвердить, что вы ссылаетесь на правильный тип данных.

Возвращаемое значение

Метод getValueAsDate возвращает дату.

Пример

Следующий пример - это отрывок из метода, обрабатывающего NameValuePair и печатающего значение, если это дата.

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

getValueAsNumeric

Метод `getValueAsNumeric` возвращает значение объекта NameValuePair.

```
getValueAsNumeric()
```

Надо использовать `getValueDataType` перед использованием `getValueAsNumeric`, чтобы подтвердить, что вы ссылаетесь на правильный тип данных.

Возвращаемое значение

Метод `getValueAsNumeric` возвращает число двойной точности. Если, например, вы получаете значение, первоначально сохраненное в вашей таблице профилей как Integer, `getValueAsNumeric` возвращает число двойной точности.

Пример

Следующий пример - это отрывок из метода, обрабатывающего NameValuePair и печатающего значение, если это число.

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

getValueAsString

Метод `getValueAsString` возвращает значение объекта NameValuePair.

```
getValueAsString()
```

Надо использовать `getValueDataType` перед использованием `getValueAsString`, чтобы подтвердить, что вы ссылаетесь на правильный тип данных.

Возвращаемое значение

Метод `getValueAsString` возвращает строку. Если, например, вы получаете значение, первоначально сохраненное в вашей таблице профилей как char, varchar или char[10], `getValueAsString` возвращает строку.

Пример

Следующий пример - это отрывок из метода, обрабатывающего NameValuePair и печатающего значение, если это строка.

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

getValueDataType

Метод getValueDataType возвращает тип данных объекта NameValuePair.

```
getValueDataType()
```

Надо использовать getValueDataType перед использованием getValueAsDate, getValueAsNumeric или getValueAsString, чтобы подтвердить, что вы ссылаетесь на правильный тип данных.

Возвращаемое значение

Метод getValueDataType возвращает строку, указывающую, содержит ли NameValuePair данные, число или строку.

Допустимые значения:

- **DATA_TYPE_DATETIME** - дата, содержащая значение даты и времени.
- **DATA_TYPE_NUMERIC** - число двойной точности, содержащее числовое значение.
- **DATA_TYPE_STRING** - строка, содержащая текстовое значение.

Пример

Следующий пример - это отрывок из метода, обрабатывающего объект ответа от метода getProfile.

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

setName

Метод setName определяет компонент имени объекта NameValuePair.

```
setName(name)
```

- **name** - строка, содержащая компонент имени объекта NameValuePair.

Возвращаемое значение

Нет.

Пример

Следующий пример показывает, как определить компонент имени NameValuePair.

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

setValueAsDate

Метод `setValueAsDate` определяет значение объекта NameValuePair.

`setValueAsDate(`*`valueAsDate`*`)`

- **valueAsDate** - дата, содержащая значение даты и времени объекта NameValuePair.

Возвращаемое значение

Нет.

Пример

Следующий пример показывает, как определить компонент значения NameValuePair, если это значение - дата.

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

setValueAsNumeric

Метод `setValueAsNumeric` определяет значение объекта NameValuePair.

`setValueAsNumeric(`*`valueAsNumeric`*`)`

- **valueAsNumeric** - число двойной точности, содержащее значение объекта NameValuePair

Возвращаемое значение

Нет.

Пример

Следующий пример показывает, как определить компонент значения NameValuePair, если это значение - числовое.

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

setValueAsString

Метод `setValueAsString` определяет значение объекта NameValuePair.

`setValueAsString(`*`valueAsString`*`)`

- **valueAsString** - строка, содержащая значение объекта NameValuePair

Возвращаемое значение

Нет.

Пример

Следующий пример показывает, как определить компонент значения NameValuePair, если это значение - числовое.

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

setValueDataType

Метод `setValueDataType` определяет тип данных объекта NameValuePair.

`getValueDataType(`*`valueDataType`*`)`

Допустимые значения:

- **DATA_TYPE_DATETIME** - дата, содержащая значение даты и времени.
- **DATA_TYPE_NUMERIC** - число двойной точности, содержащее числовое значение.
- **DATA_TYPE_STRING** - строка, содержащая текстовое значение.

Возвращаемое значение

Нет.

Пример

Следующие примеры показывают, как задать тип данных значения NameValuePair.

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

NameValuePair parm3 = new NameValuePairImpl();
parm3.setName("Браузер");
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);
```

О классе Offer

Класс `Offer` содержит методы, определяющие объект Offer. Этот объект предложения содержит многие из аналогичных свойств предложения в Campaign.

Объект предложения содержит следующие атрибуты:

- **AdditionalAttributes** - NameValuePairs, содержащие пользовательские атрибуты предложения, которые вы определили в Campaign.
- **Description** - Описание предложения.
- **EffectiveDate** - Дата вступления предложения в силу.
- **ExpirationDate** - Дата истечения срока предложения.
- **OfferCode** - Код предложения.
- **OfferName** - Имя предложения.
- **TreatmentCode** - Код обработки предложения.

- **Score** - маркетинговая оценка предложения или оценка, заданная ScoreOverrideTable, если значение свойства enableScoreOverrideLookup - true.

getAdditionalAttributes

Метод getAdditionalAttributes возвращает пользовательские атрибуты предложения, определенные в Campaign.

```
getAdditionalAttributes()
```

Возвращаемое значение

Метод getAdditionalAttributes возвращает массив объектов NameValuePair.

Пример

Следующий пример сортирует все дополнительные атрибуты, проверяя дату вступления в силу и дату истечения срока, и печатает остальные атрибуты.

```
for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
{
    // проверяет, задана ли дата вступления в силу
    if(offerAttribute.getName().equalsIgnoreCase("effectiveDate"))
    {
        System.out.println("Найдена дата вступления в силу");
    }
    // проверяет, задана ли дата истечения срока
    else if(offerAttribute.getName().equalsIgnoreCase("expirationDate"))
    {
        System.out.println("Найдена дата истечения срока");
    }
    printNameValuePair(offerAttribute);
    }
}
public static void printNameValuePair(NameValuePair nvp)
{
    // печать имени:
    System.out.println("Имя:"+nvp.getName());

    // на основе типе данных вызываем соответствующий метод для получения значения
    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

Метод getDescription возвращает описание предложения, определенного в Campaign.

```
getDescription()
```

Возвращаемое значение

Метод getDescription возвращает строку.

Пример

Следующий пример печатает описание предложения.

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

getOfferCode

Метод getOfferCode возвращает код предложения, как он определен в Campaign.

```
getOfferCode()
```

Возвращаемое значение

Метод getOfferCode возвращает массив строк, содержащих код предложения.

Пример

Следующий пример печатает код предложения.

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

getOfferName

Метод getOfferName возвращает имя предложения, как оно определено в Campaign.

```
getOfferName()
```

Возвращаемое значение

Метод getOfferName возвращает строку.

Пример

Следующий пример печатает имя предложения.

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

getScore

Метод getScore возвращает оценку на основе сконфигурированного вами предложения.

```
getScore()
```

Метод getScore возвращает одно из следующих значений:

- Если вы не включили таблицу предложений по умолчанию, таблицу переопределения оценок или встроенное обучение, этот метод возвращает маркетинговую оценку предложения, определенную на вкладке стратегии взаимодействия.
- Если вы включили таблицу предложений по умолчанию или таблицу переопределения оценок, но не включили встроенное обучение, этот метод возвращает оценку предложения, определенную в порядке предпочтения между таблицей предложений по умолчанию, оценкой маркетолога и таблицей переопределения оценок.

- Если вы включении встроенное обучение, этот метод возвращает окончательную оценку, которую встроенное обучение использовало для заказа предложения.

Возвращаемое значение

Метод `getScore` возвращает целое число, представляющее оценку предложения.

Пример

Следующий пример печатает оценку предложения.

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

getTreatmentCode

Метод `getTreatmentCode` возвращает код процедуры, как он определен в Campaign.

```
getTreatmentCode()
```

Поскольку Campaign использует код процедуры для идентификации экземпляра представленного предложения, этот код должен быть возвращен как параметр события при использовании метода `postEvent` для записи в журнал события контакта, принятия или отклонения предложения. Если вы записывает в журнал принятие или отклонение предложения, надо задать значение имени NameValuePair, представляющего код процедуры `UACIOfferTrackingCode`.

Возвращаемое значение

Метод `getTreatmentCode` возвращает строку.

Пример

Следующий пример печатает код процедуры предложения.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // печать предложения
    System.out.println("Код обработки предложения:"+offer.getTreatmentCode());
}
```

О классе OfferList

Класс `OfferList` содержит методы, определяющие результаты метода `getOffers`.

Объект OfferList содержит следующие атрибуты:

- **DefaultString** - Строка по умолчанию, определенная для точки взаимодействия в канале взаимодействия.
- **RecommendedOffers** - Массив объектов Offer, затребованных методом `getOffers`.

Класс `OfferList` работает со списками предложений. Этот класс не связан со списками предложений Campaign.

getDefaultString

Метод `getDefaultString` возвращает строку по умолчанию для точки взаимодействия, как определено в Campaign.

```
getDefaultString()
```

Если объект RecommendedOffers пуст, надо сконфигурировать точку для представления этой строки, чтобы обеспечить вывод некоторого содержания. Interact заполняет объект DefaultString, только если объект RecommendedOffers пуст.

Возвращаемое значение

Метод `getDefaultString` возвращает строку.

Пример

Следующий пример получает строку по умолчанию, если объект offerList не содержит никаких предложений.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
  System.out.println("Имя предложения:"+offer.getOfferName());
    }
}
else // строка предложения по умолчанию
    System.out.println("Предложение по умолчанию:"+offerList.getDefaultString());
```

getRecommendedOffers

Метод `getRecommendedOffers` возвращает массив объектов Offer, затребованных методом `getOffers`.

```
getRecommendedOffers()
```

Если ответ на `getRecommendedOffer` пуст, точка должна представить результат `getDefaultString`.

Возвращаемое значение

Метод `getRecommendedOffers` возвращает объект Offer.

Пример

Следующий пример обрабатывает объект OfferList и печатает имена всех рекомендуемых предложений.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
        // печать предложения
        System.out.println("Имя предложения:"+offer.getOfferName());
    }
}
else // строка предложения по умолчанию
System.out.println("Предложение по умолчанию:"+offerList.getDefaultString());
```

О классе Response

Класс `Response` содержит методы, определяющие результаты методов класса InteractAPI.

Объект Response содержит следующие атрибуты:

- **AdvisoryMessages** - массив сообщений рекомендации. Этот атрибут заполняется, только если при запуске метода возникли предупреждения или ошибки.
- **ApiVersion** - строка, содержащая версию API. Этот атрибут заполняется методом `getVersion`.
- **OfferList** - объект OfferList, содержащий предложения, которые затребованы методом `getOffers`.
- **ProfileRecord** - массив NameValuePairs, содержащий данные профиля. Этот атрибут заполняется методом `getProfile`.
- **SessionID** - строка, определяющая ID сеанса. Возвращается любым методом класса `InteractAPI`.
- **StatusCode** - число, указывающее, как работает метод - без ошибок, с предупреждениями или с ошибками. Возвращается любым методом класса `InteractAPI`.

getAdvisoryMessages

Метод `getAdvisoryMessages` возвращает массив сообщений рекомендации от объекта Response.

```
getAdvisoryMessages()
```

Возвращаемое значение

Метод `getAdvisoryMessages` возвращает массив объектов сообщений рекомендации.

Пример

Следующий пример получает объекты AdvisoryMessage от объекта Response и в цикле по этим сообщениям выводит их на печать.

```
AdvisoryMessage[] messages = response.getAdvisoryMessages();
    for(AdvisoryMessage msg : messages)
    {
        System.out.println(msg.getMessage());
        // Некоторые сообщения рекомендации могут содержать дополнительные подробности:
        System.out.println(msg.getDetailMessage());
    }
```

getApiVersion

Метод `getApiVersion` возвращает версию API объекта Response.

```
getApiVersion()
```

Метод `getVersion` заполняет атрибут ApiVersion объекта Response.

Возвращаемое значение

Объект Response возвращает строку.

Пример

Следующий пример - это отрывок из метода, обрабатывающего объект ответа для getVersion.

```
if(response.getStatusCode() == Response.STATUS_SUCCESS)
{
    System.out.println("Вызов getVersion обработан без предупреждений или ошибок");
    System.out.println("Версия API:" + response.getApiVersion());
}
```

getOfferList

Метод getOfferList возвращает объект OfferList объекта Response.

```
getOfferList()
```

Метод getOffers заполняет объект OfferList объекта Response.

Возвращаемое значение

Объект Response возвращает объект OfferList.

Пример

Следующий пример - это отрывок из метода, обрабатывающего объект ответа для getOffers.

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

getAllOfferLists

Метод getAllOfferLists возвращает массив всех OfferLists объекта Response.

```
getAllOfferLists()
```

Это используется методом getOffersForMultipleInteractionPoints, который заполняет объект массива OfferList объекта Response.

Возвращаемое значение

Объект Response возвращает массив OfferList.

Пример

Следующий пример - это отрывок из метода, обрабатывающего объект ответа для getOffers.

```
OfferList[] allOfferLists = response.getAllOfferLists();
 if (allOfferLists != null) {
  for (OfferList ol : allOfferLists) {
   System.out.println("Следующие предложения поставляются для точки взаимодействия "
     + ol.getInteractionPointName() + ":");
   for (Offer o : ol.getRecommendedOffers()) {
```

```
    System.out.println(o.getOfferName());
   }
  }
 }
```

getProfileRecord

Метод `getProfileRecord` возвращает записи профиля для текущего сеанса как массив объектов NameValuePair. Эти записи профилей также включают в себя любые параметры `eventParameters`, добавленные ранее в сеансе выполнения.

```
getProfileRecord()
```

Метод `getProfile` заполняет запись профиля объектами NameValuePair объекта Response.

Возвращаемое значение

Объект Response возвращает массив объектов NameValuePair.

Пример

Следующий пример - это отрывок из метода, обрабатывающего объект ответа для `getOffers`.

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

getSessionID

Метод `getSessionID` возвращает ID сеанса.

```
getSessionID()
```

Возвращаемое значение

Метод `getSessionID` возвращает строку.

Пример

Следующий пример показывает сообщение, которое можно вывести в конце или в начале обработки ошибок, чтобы указать, к какому сеансу относятся ошибки.

```
System.out.println("Этот ответ относится к ID сеанса:"+response.getSessionID());
```

getStatusCode

Метод `getStatusCode` возвращает код состояния объекта Response.

```
getStatusCode()
```

Возвращаемое значение

Объект Response возвращает целое число.

- 0 - STATUS_SUCCESS - Вызванный метод завершился без ошибок. Возможно, есть сообщения рекомендации.
- 1 - STATUS_WARNING - Вызванный метод завершился по крайней мере с одним предупреждением (но без ошибок). Запросите сообщения рекомендации, чтобы узнать подробности.
- 2 - STATUS_ERROR - Вызванный метод не завершился успешно, и есть по крайней мере одно сообщение об ошибке. Запросите сообщения рекомендации, чтобы узнать подробности.

Пример

Далее приводится пример использования `getStatusCode` при обработке ошибок.

```
public static void processSetDebugResponse(Response response)
{
    // проверить успешность ответа
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("Вызов setDebug обработан без предупреждений или ошибок");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("Вызов setDebug обработан с предупреждением");
    }
    else
    {
        System.out.println("Вызов setDebug обработан с ошибкой");
    }

    // В случае каких-либо неудач должны быть поясняющие рекомендации
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setDebug",
    response.getAdvisoryMessages());
}
```

Глава 8. Классы и методы для API IBM Interact JavaScript

В следующих разделах перечислены требования и другие сведения, которые нужно знать перед началом работы с API JavaScript Interact.

API Interact поддерживает разновидность JavaScript, позволяющую клиенту конечного пользователя (браузеру) взаимодействовать с сервером.

Примечание: В этом разделе предполагается, что вы знакомы с API на основе JavaScript.

Примечание: Многократные вхождений одного параметра в одном вызове API не поддерживаются.

Предварительные требования JavaScript

Прежде, чем использовать API JavaScript Interact на сайте, надо включить файл `interactapi.js` на ваших веб-страницах.

Работа с данными сеанса

Если вы инициируете сеанс с помощью метода `startSession`, данные сеанса будут загружены в память. На протяжении сеанса можно считывать и записывать данные сеанса (которые являются надмножеством данных статического профиля).

Сеанс содержит следующие данные:

- Данные статического профиля
- Назначения сегмента
- Данные реального времени
- Рекомендации предложения

Все данные сеанса доступны, пока вы не вызовете метод `endSession` или пока не истечет время `SessionTimeout`. По окончании сеанса все данные, не сохраненные в явном виде в хронологии контактов, хронологии ответов или какой-либо другой таблице базы данных, будут потеряны.

Данные хранятся в виде набора пар имя - значение. Если данные считаны из таблицы базы данных, используется имя столбца таблицы.

Можно создать эти пары значение-имя, когда вы работаете с API Interact. Необязательно объявлять все пары значение-имя в глобальной области. При задании новых параметров событий в виде пар имя - значение среда выполнения добавляет пары имя - значение к данным сеанса. Например, если вы используете параметры событий с методом `postEvent`, среды выполнения добавит параметры событий в данные сеанса, даже если параметры событий были недоступны в данных профиля. Эти данные существуют только в данных сеанса.

В любое время можно перезаписать данные сеанса. Например, если часть профиля покупателя включает `creditScore`, вы можете передать параметр события с помощью пользовательского типа `NameValuePair`. В классе `NameValuePair` можно изменить значение при помощи методов `setName` и `setValueAsNumeric`. Имена

должны соответствовать друг другу. В данных сеанса имя не является регистрозависимым. Таким образом, оценка `creditScore` будет перезаписана вариантами `creditscore` или `CrEdItScOrE`.

Сохраняются только последние данные, записанные в сеанс. Например, метод `startSession` загружает данные профиля для значения `lastOffer`. Метод `postEvent` перезаписывает `lastOffer`. Затем второй метод `postEvent` перезаписывает `lastOffer`. В среде выполнения сохраняются только данные, записанные в данных сеанса вторым методом `postEvent`.

По завершении сеанса данные будут потеряны, если не предпринять специальных шагов, например, не записать данные в таблицу базы данных с помощью процесса Снимок в вашей интерактивной потоковой диаграмме. Если вы планируете использовать процесс Снимок, учтите, что для имен должны соблюдаться ограничения вашей базы данных. Например, если разрешено использовать в именах столбцов не более 256 символов, имя для пары имя - значение не может содержать более 256 символов.

Работа с параметром callback

Функция обратного вызова - это дополнительный параметр в каждом методе API JavaScript Interact.

Основной процесс браузера - это однопотоковый цикл событий. Выполнение длительных операций в однопотоковом цикле событий блокирует процесс. В результате этот процесс не будет обрабатывать следующие события, ожидая завершения вашей операции. Для предотвращения блокировки в длительных процессах в XMLHttpRequest предусмотрен асинхронный интерфейс. Этому интерфейсу вы передаете обратный вызов для запуска после завершения операции, и в ходе обработки он возвращает управление главному циклу событий, не блокируя процесс.

Если метод завершился удачно, функция обратного вызова вызывает`onSuccess`. Если метод завершился неудачно, функция обратного вызова вызывает `onError`.

Например, если требуется показать предложения на вашей веб-странице, воспользуйтесь методом `getOffers` и обратным вызовом для вывода предложений на странице. Веб-страница отреагирует обычным образом и не будет ждать, пока Interact возвратит предложения. Вместо этого, когда Interact возвратит предложения, ответ будет отправлен обратно параметру callback. Вы можете проанализировать данные обратного вызова и показать предложения на странице.

Можно использовать один универсальный обратный вызов для всех функций или же отдельные обратные вызовы для конкретных функций.

Для создания универсальной функции обратного вызова введите `var callback = InteractAPI.Callback.create(onSuccess, onError);`.

Для создания конкретных функций обратного вызова для метода getOffers воспользуйтесь следующей функцией.

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

О классе InteractAPI

Класс InteractAPI содержит методы, используемые вами для интеграции точки контакта с сервером среды выполнения. Все прочие классы и методы в API Interact поддерживают методы в этом классе.

Надо скомпилировать реализацию с помощью файла `interact_client.jar`, расположенного в каталоге `lib` вашей установки Interact среды выполнения.

startSession

Метод `startSession` создает и определяет сеанс среды выполнения.

```
function callStartSession(commandsToExecute, callback) {

        // читаем сконфигурированный запуск сеанса
        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` может инициировать до пяти действий:

- создать сеанс среды выполнения.
- загрузить в сеанс выполнения данные профиля посетителя для текущего уровня аудитории, в том числе любые таблицы измерений, помеченные для загрузки в отображении таблиц, которое определено для интерактивного канала.
- запустить сегментацию, выполняя все интерактивные потоковые диаграммы для текущего уровня аудитории.
- загрузить в сеанс данные подавления предложений, если для свойства `enableOfferSuppressionLookup` задано значение true.
- загрузить в сеанс данные переопределения оценок, если для свойства `enableScoreOverrideLookup` задано значение true.

Метод `startSession` требует следующих параметров:

- **sessionID** - строка, определяющая ID сеанса. Надо задать ID сеанса. Например, вы могли использовать сочетание ID покупателя и отметки времени.

  Чтобы определить, что составляет сеанс выполнения, надо указать ID сеанса. Этим значением управляет клиент. Все вызовы методов для одного и того же ID сеанса должны быть синхронизированы клиентом. Поведение для одновременных вызовов API с одним и тем ID сеанса не определено.
- **relyOnExistingSession** - логическое значение, определяющее, использует ли данный сеанс новый или же существующий сеанс. Допустимые значения: `true` и `false`. При значении `true` в методе `startSession` надо задать ID существующего сеанса. При значении `false` надо задать новый ID сеанса.

  Если вы задали для `relyOnExistingSession` значение `true`, и сеанс существует, среда выполнения использует данные существующего сеанса, не перезагружает никакие данные и не запускает сегментацию. Если этот сеанс не существует, среда выполнения создает новый сеанс, загружает данные и запускает сегментацию.

Задание для relyOnExistingSession значения true и его использование для всех вызовов startSession полезно, если длительность сеанса для вашей точки контакта больше, чем длительность сеанса среды выполнения. Например, сеанс сайта длится 2 часа, но сеанс среды выполнения длится только 20 минут.

Если вы вызываете startSession дважды с одним и тем же самым ID сеанса, при значении relyOnExistingSession false все данные сеанса первого вызова startSession теряются.

- **debug** - логическое значение, которое включает или выключает запись отладочной информации. Допустимые значения: true и false. Если задано true, Interact записывает отладочную информацию в журналы сервера среды выполнения. Флаг отладки задается для каждого сеанса отдельно. Таким образом, вы можете отслеживать данные отладки для отдельного сеанса.
- **interactiveChannel** - строка, определяющая имя интерактивного канала, к которому относится данный сеанс. Это имя должно точно соответствовать имени интерактивного канала, определенного в Campaign.
- **audienceID** - массив объектов NameValuePairImpl, в котором имена должны соответствовать именам физических столбцов любой таблицы, содержащей ID аудитории.
- **audienceLevel** - строка, определяющая уровень аудитории.
- **parameters** - объекты NameValuePairImpl, идентифицирующие параметры, которые требуется передать с startSession. Эти значения хранятся в данных сеанса и могут использоваться для сегментации.

  Если у вас есть несколько интерактивных потоковых диаграмм для одного и того же уровня аудитории, надо включить надмножество всех столбцов во всех таблицах. Если вы конфигурируете среду выполнения для загрузки таблицы профилей, и эта таблица профилей содержит все нужные вам столбцы, не требуется передавать никакие параметры, если только вы не хотите перезаписать данные в таблице профилей. Если ваша таблица профилей содержит только подмножество необходимых столбцов, надо включить недостающие столбцы как параметры.
- **callback** - Если метод выполнен успешно, функция обратного вызова вызывает onSuccess. Если метод завершился неудачно, функция обратного вызова вызывает onError.

Если audienceID или audienceLevel недопустимы и значение relyOnExistingSession - false, вызов startSession завершается неудачно. Если interactiveChannel недопустим, startSession завершается неудачно, независимо от того, какое значение задано для relyOnExistingSession - true или false.

Если для relyOnExistingSession задано значение true, и вы делаете второй вызов startSession с тем же sessionID, а срок ожидания для первого сеанса истек, Interact создает новый сеанс.

Если для relyOnExistingSession задано значение true, и вы делаете второй вызов startSession с тем же sessionID, но с другим audienceID или audienceLevel, сервер среды выполнения меняет аудиторию для существующего сеанса.

Если для relyOnExistingSession задано значение true, и вы делаете второй вызов startSession с тем же sessionID, но с другим interactiveChannel, сервер среды выполнения создает новый сеанс.

Возвращаемое значение

Сервер среды выполнения отвечает на startSession объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages (если StatusCode не равен 0)
- ApiVersion
- SessionID
- StatusCode

Дедупликация предложений по атрибутам предложений

При использовании интерфейса прикладного программирования (application programming interface, API) Interact предложения доставляют два вызова API: `getOffers` и `getOffersForMultipleInteractionPoints`. Вызов `getOffersForMultipleInteractionPoints` позволяет не допустить возврата дубликатов предложений на уровне *OfferID*, но не может дедуплицировать предложения в категории предложений. Поэтому, например, для Interact, чтобы возвратить только одно предложение из каждой категории предложений, ранее требовался обходной путь. За счет добавления двух параметров в вызов API `startSession` теперь стала возможна дедупликация предложений по другим атрибутам предложений, например, категории.

Этот список содержит сводку параметров, которые были добавлены к вызову API `startSession`. Дополнительную информацию об этих параметрах и о любых аспектах API Interact смотрите в *Руководстве администратора IBM Interact* или в файлах Javadoc, включенных в вашу установку Interact в `<домашний_каталог_Interact>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Чтобы создать вызов API `startSession` с дедупликацией предложений, так что последующие вызовы `getOffer` возвращали только по одному предложению из каждой категории, надо включить параметр `UACIOfferDedupeAttribute` как часть вызова API. Параметр можно задать в формате `имя,значение,тип`, как показано здесь:

  `UACIOfferDedupeAttribute,<attributeName>,string`

  В этом примере вам надо заменить *<имя_атрибута>* на имя атрибута предложения, который вы хотите использовать в качестве критерия дедупликации, например, Category.

  Примечание: Interact проверяет предложения с одинаковыми значениями заданного вами атрибута и удаляет все предложения, кроме предложения с максимальной оценкой. Если у предложений с атрибутом-дубликатом идентичные оценки, Interact возвращает случайный выбор из соответствующих предложений.

- `UACINoAttributeDedupeIfFewerOffer`. Когда вы включаете `UACIOfferDedupeAttribute` в вызов `startSession`, можно также задать параметр `UACINoAttributeDedupeIfFewerOffer`, чтобы определить поведение в случаях, когда список предложений после дедупликации больше не содержит достаточно предложений для удовлетворения исходного требования.

  Например, если вы задали `UACIOfferDedupeAttribute`, чтобы использовать категорию предложения для дедупликации предложений, и ваш последующий вызов getOffers требует возврата восьми предложений, в результате дедупликации пригодных предложений может стать меньше восьми. В этом случае задание для параметра `UACINoAttributeDedupeIfFewerOffer` значения true привело бы к добавлению некоторых дубликатов в список пригодных предложений, чтобы соблюсти требуемое число предложений. В этом примере, если вы задали для параметра значение false, число возвращенных предложений было бы меньше, чем требуемое.

По умолчанию в качестве значения UACINoAttributeDedupeIfFewerOffer задается true.

Например, предположим, что вы задали в параметрах startSession, что критерием дедупликации служит категория предложения, как показано здесь:

```
UACIOfferDedupeAttribute,Category,string;

UACINoAttributeDedupeIfFewerOffer,1,string
```

По умолчанию UACIOfferDedupeAttribute не будет дедуплицировать предложения, если возвращается меньше затребованного количества. Однако чтобы указать выполнение дедупликации при возврате меньшего, чем затребовано, количества предложений, надо задать параметр UACINoAttributeDedupeIfFewerOffer и указать для него значение 1.

Эти параметры вместе приводят к тому, что Interact дедуплицирует предложения на основе атрибута предложения "Category" и возвращает только предложения без дубликатов, даже если полученных предложений меньше, чем затребовано (для UACINoAttributeDedupeIfFewerOffer задано false).

Когда вы выполняете вызов API getOffers, исходный набор пригодных предложений может включать в себя следующие предложения:

- Category=Электроника: Предложение A1 с оценкой 100 и Предложение A2 с оценкой 50.
- Category=Смартфоны: Предложение B1 с оценкой 100, Предложение B2 с оценкой 80 и Предложение B3 с оценкой 50.
- Category=MP3-плееры: Предложение C1 с оценкой 100, Предложение C2 с оценкой 50.

В данном случае есть два предложения-дубликата в первой категории, три предложения-дубликата во второй категории и два предложения-дубликата в третьей категории. Возвращаются предложения с максимальной оценкой в каждой категории, то есть Предложение A1, Предложение B1 и Предложение C1.

Если в вызове API getOffers затребовано шесть предложений, в этом наборе примера для UACINoAttributeDedupeIfFewerOffer задано false, таким образом были бы возвращены только три предложения.

Если вызов API getOffers требует шесть предложений, и в этом примере параметр UACINoAttributeDedupeIfFewerOffer опущен или задан как true, некоторые предложения-дубликаты были бы включены в результат, чтобы обеспечить требуемое количество.

postEvent

Метод postEvent позволяет выполнять любое событие, определенное в интерактивном канале.

```
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**: строка, определяющая ID сеанса.
- **eventName**: строка, задающая имя события.

  Примечание: Имя события должно соответствовать имени события, как определено в интерактивном канале. Регистр символов в этом имени не учитывается.
- **eventParameters**. Объекты `NameValuePairImpl`, идентифицирующие параметры, которые требуется передать с этим событием. Эти значения хранятся в данных сеанса.

  Если это событие инициирует пересегментацию, надо убедиться, что все данные, требуемые интерактивными потоковыми диаграммами, доступны в данных сеанса. Если какое-либо из этих значений не было заполнено предшествующими действиями (например, `startSession` или `setAudience` или загрузкой таблицы профилей), надо включить eventParameter для каждого отсутствующего значения. Например, если вы сконфигурировали все таблицы профиля для загрузки в память, надо включить NameValuePair для временных данных, требуемых для интерактивных потоковых диаграмм.

  Если вы используете несколько уровней аудитории, скорее всего у вас есть различные наборы eventParameters для каждого уровня аудитории. Надо включать некоторую логику, чтобы обеспечить выбор правильного набора параметров для уровня аудитории.

  Важное замечание: Если это событие записывается в хронологию ответов, надо передать код процедуры для предложения. Надо задать имя для NameValuePair как `"UACIOfferTrackingCode"`.

  Для каждого события можно передать только один код процедуры. Если вы не передаете код процедуры для контакта предложения, Interact записывает в журнал контакт предложения для каждого предложения в последнем рекомендованном списке предложений. Если вы не передаете код процедуры для ответа, Interact возвращает ошибку.
- **callback** - Если метод выполнен успешно, функция обратного вызова вызывает `onSuccess`. Если метод завершился неудачно, функция обратного вызова вызывает `onError`.
- Есть несколько других зарезервированных параметров, используемых с `postEvent`, и другие методы; они обсуждаются далее в этом разделе.

Любое требование пересегментации или записи в хронологию контактов или ответов не ожидает ответа.

Пересегментация не очищает результаты предыдущей сегментации для текущего уровня аудитории. Можно использовать параметр `UACIExecuteFlowchartByName`, чтобы задать конкретные потоковые диаграммы для выполнения. Метод `getOffers` ожидает завершения пересегментации перед выполнением. Поэтому, если вы вызываете метод `postEvent`, который запускает пересегментацию, непосредственно перед вызовом `getOffers`, возможна задержка.

Возвращаемое значение

Сервер среды выполнения отвечает на `postEvent` объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- OfferList

- Profile
- SessionID
- StatusCode

getOffers

Метод `getOffers` позволяет затребовать предложения от сервера среды выполнения.

```
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** - строка, определяющая текущий сеанс.
- **Interaction point** - строка, задающая имя точки взаимодействия, на которую ссылается этот метод.

  Примечание: Это имя должно точно соответствовать имени точки взаимодействия, определенной в интерактивном канале.
- **nofRequested** - целое число, определяющее число затребованных предложений.
- **callback** - Если метод выполнен успешно, функция обратного вызова вызывает `onSuccess`. Если метод завершился неудачно, функция обратного вызова вызывает `onError`.

Метод `getOffers` ожидает срок (в миллисекундах), заданный свойством `segmentationMaxWaitTimeInMS`, чтобы вся пересегментация завершилась до запуска. Поэтому, если вы вызываете метод `postEvent`, который запускает пересегментацию, или метод `setAudience` непосредственно перед вызовом `getOffers`, возможна задержка.

Возвращаемое значение

Сервер среды выполнения отвечает на `getOffers` объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getOffersForMultipleInteractionPoints

Метод `getOffersForMultipleInteractionPoints` позволяет затребовать предложения от сервера среды выполнения для нескольких IP с устранением дублирования.

```
function callGetOffersForMultipleInteractionPoints(commandsToExecute, callback) {

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

        // усекаем строку
```

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

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

        //строим получение требований предложений
        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** - строка, определяющая текущий сеанс.
- **requestDetailsStr** - строка, задающая массив объектов `GetOfferRequest`.

  Каждый объект `GetOfferRequest` задает:

  – **ipName** - Имя точки взаимодействия (interaction point, IP), для которой объект запрашивает предложения

  – **numberRequested** - Число уникальных предложений, требуемых для указанной IP

  – **offerAttributes** - Требования атрибутов присылаемых предложений, используемых экземпляром `OfferAttributeRequirements`

  – **duplicationPolicy** - ID политики дубликатов для присылаемых предложений

    Политики дубликатов определяют, будут ли возвращаться предложения-дубликаты для различных точек взаимодействия в одном вызове метода. (*В пределах* одной точки взаимодействия предложения-дубликаты никогда не возвращаются.) В настоящее время поддерживаются две политики дубликатов.

    - NO_DUPLICATION (значение ID = 1). Никакое из предложений, включенных в предыдущие экземпляры `GetOfferRequest`, не будут включены в этот экземпляр `GetOfferRequest` (то есть Interact применяет дедупликацию).
    - ALLOW_DUPLICATION (значение ID = 2). Любое из предложений, удовлетворяющих заданным в этом экземпляре `GetOfferRequest` требованиям, будет включено. Предложения, включенные в предыдущие экземпляры `GetOfferRequest`, не будут согласовываться.

  – **callback** - Если метод отработал успешно, то функция обратного вызова вызывает `onSuccess`. Если метод завершился неудачно, то функция обратного вызова вызывает `onError`.

  Порядок требований в параметре массива - это порядок приоритетов при представлении предложений.

  Например, предположим, что точки взаимодействия в требовании - IP1, а затем IP2, что предложения-дубликаты запрещены (ID политики дубликатов = 1), и для каждой точки затребованы два предложения. Если Interact обнаруживает предложения A, B и C для IP1 и предложения A и D для IP2, ответ будет содержать предложения A и B для IP1 и только предложение D для IP2.

  Также заметим, что когда ID политики дубликатов - 1, предложения, представленные для точки с более высоким приоритетом, не будут представлены для данной точки.

Метод getOffersForMultipleInteractionPoints ожидает срок (в миллисекундах), заданный свойством segmentationMaxWaitTimeInMS, чтобы вся пересегментация завершилась до запуска. Поэтому, если вы вызываете метод postEvent, который запускает пересегментацию, или метод setAudience непосредственно перед вызовом getOffers, возможна задержка.

Возвращаемое значение

Сервер среды выполнения отвечает на getOffersForMultipleInteractionPoints объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- массив OfferList
- Profile
- SessionID
- StatusCode

setAudience

Метод setAudience позволяет задавать ID и уровень аудитории для посетителя.

```
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**-строка, определяющая ID сеанса.
- **audienceID** - массив объектов NameValuePairImpl, которые определяют уровень аудитории.
- **audienceLevel** - строка, определяющая уровень аудитории.
- **parameters** - объекты NameValuePairImpl, идентифицирующие параметры, которые требуется передать с setAudience. Эти значения хранятся в данных сеанса и могут использоваться для сегментации.

  У вас должно быть значение для каждого столбца в вашем профиле. Это надмножество всех столбцов во всех таблицах, определенных для интерактивного канала и любых данных реального времени. Если вы уже заполнили все данные сеанса с помощью startSession или postEvent, нет необходимости посылать новые параметры.
- **callback** - Если метод выполнен успешно, функция обратного вызова вызывает onSuccess. Если метод завершился неудачно, функция обратного вызова вызывает onError.

Метод setAudience запускает пересегментацию. Метод getOffers ожидает завершения пересегментации перед выполнением. Поэтому, если вы вызываете метод setAudience непосредственно перед вызовом getOffers, возможна задержка.

Метод setAudience также загружает данные профиля для ID аудитории. Можно использовать метод setAudience для принудительной перезагрузки тех же данных профиля, которые загружены методом startSession.

Метод setAudience перезагружает таблицы белого списка и черного списка в существующем сеансе. Можно использовать метод setAudience с параметрами UACIPurgePriorWhiteListOnLoad и UACIPurgePriorBlackListOnLoad для перезагрузки таблиц белого и черного списка в существующем сеансе.

По умолчанию, когда вызывается метод setAudience, все содержание черного списка удаляется. Можно задать параметры UACIPurgePriorWhiteListOnLoad и UACIPurgePriorBlackListOnLoad в вызове setAudience следующим образом:

- Если задать UACIPurgePriorBlackListOnLoad= 0, все содержимое таблицы белого списка сохраняется.
- Если задать UACIPurgePriorWhiteListOnLoad= 1, содержимое таблицы удаляется, и содержимое белого списка или черного списка для ID аудитории будет загружено из базы данных. После выполнения будет запущена пересегментация.

Возвращаемое значение

Сервер среды выполнения отвечает на setAudience объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getProfile

Метод getProfile позволяет получить профиль и временную информацию о посетителе, посещающем точку контакта.

```
function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        InteractAPI.getProfile(ssId, callback);

}
```

- **ID сеанса**-строка, определяющая ID сеанса.
- **callback** - Если метод отработал успешно, то функция обратного вызова вызывает onSuccess. Если метод завершился неудачно, то функция обратного вызова вызывает onError.

Возвращаемое значение

Сервер среды выполнения отвечает на getProfile объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- OfferList
- ProfileRecord
- SessionID
- StatusCode

endSession

Метод endSession отмечает конец сеанса во время выполнения. Когда сервер среды выполнения получает этот метод, сервер среды выполнения записывает информацию в журнал хронологии, очищает память и так далее.

```
function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        InteractAPI.endSession(ssId, callback);

}
```

- **ID сеанса** - Уникальная строка, определяющая сеанс.
- **callback** - Если метод отработал успешно, то функция обратного вызова вызывает onSuccess. Если метод завершился неудачно, то функция обратного вызова вызывает onError.

Если метод endSession не вызван, сеансы среды выполнения завершаются по истечению срока бездействия. Срок ожидания конфигурируется с помощью свойства sessionTimeout.

Возвращаемое значение

Сервер среды выполнения отвечает на метод endSession объектом Response со следующими заполненными атрибутами:

- SessionID
- ApiVersion
- OfferList
- Profile
- StatusCode
- AdvisoryMessages

setDebug

Метод setDebug позволяет задать уровень подробности записи в журнал для всех путей кода в сеансе.

```
function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        InteractAPI.setDebug(ssId, isDebug, callback);

}
```

- **sessionID**-строка, определяющая ID сеанса.
- **debug** - логическое значение, которое включает или выключает запись отладочной информации. Допустимые значения: true и false. Если задано true, Interact записывает отладочную информацию в журналы сервера среды выполнения.
- **callback** - Если метод отработал успешно, то функция обратного вызова вызывает onSuccess. Если метод завершился неудачно, то функция обратного вызова вызывает onError.

Возвращаемое значение

Сервер среды выполнения отвечает на setDebug объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getVersion

Метод `getVersion` возвращает версию текущей реализации сервера среды выполнения Interact.

```
function callGetVersion(commandsToExecute, callback) {

        InteractAPI.getVersion(callback);


}
```

Рекомендуется использовать этот метод при инициализации точки контакта с API Interact.

- **callback** - Если метод отработал успешно, то функция обратного вызова вызывает `onSuccess`. Если метод завершился неудачно, то функция обратного вызова вызывает `onError`.

Возвращаемое значение

Сервер среды выполнения отвечает на `getVersion` объектом Response со следующими заполненными атрибутами:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

executeBatch

Метод `executeBatch` позволяет выполнить несколько методов за одно требование к серверу среды выполнения.

```
function callExecuteBatch(commandsToExecute, callback) {

        if (!commandsToExecute)
            return ;

        InteractAPI.executeBatch(commandsToExecute.ssid,
        commandsToExecute.commands, callback);
}
```

- **session ID** - строка, определяющая ID сеанса. Этот ID сеанса используется для всех команд, выполняемых в этом вызове метода.
- **commands** - Массив объектов команды, по одному для каждой команды, которую вы хотите выполнить.
- **callback** - Если метод выполнен успешно, функция обратного вызова вызывает `onSuccess`. Если метод завершился неудачно, функция обратного вызова вызывает `onError`.

Результат вызова этого метода эквивалентен прямому вызову каждого метода в массиве Command. Этот метод минимизирует число фактических требований к серверу среды выполнения. Сервер среды выполнения выполняет каждый метод последовательно; для каждого вызова любая ошибка или предупреждения захватываются объектом Response, соответствующем этому вызову метода. Если возникает ошибкой, executeBatch продолжает выполнять оставшуюся часть вызовов в пакете. Если выполнение какого-либо метода приводит к ошибке, состояние высокого уровня для объекта BatchResponse отражает ту ошибку. Если ошибок не происходит, состояние высокого уровня отражает любые возможные предупреждения. Если предупреждений нет, состояние высокого уровня отражает успешное выполнение пакета.

Возвращаемое значение

Сервер среды выполнения отвечает на `executeBatch` объектом BatchResponse.

Пример API JavaScript

```
function isJavaScriptAPISelected() {
    var radios = document.getElementsByName('api');
    for (var i = 0, length = radios.length; i < length; i++) {
        if (radios[i].checked) {
            if (radios[i].value === 'JavaScript')
                return true;
            else  // только одна радиокнопка может быть логически нажата
                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);

    // Надо возвратить false, чтобы отключить поведение формы по умолчанию
    return false;
}

function callStartSession(commandsToExecute, callback) {

    // читаем сконфигурированный запуск сеанса
    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;

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

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

    //строим получение требований предложений
    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);

    // проверяем, есть ли в строке требования атрибуты
    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);
}

// усекаем строку
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]);
    }
}

// удаляем открывающие и закрывающие квадратные скобки в конце
function removeOpenCloseBrackets(strToUpdate) {
    if (strToUpdate)
        return strToUpdate.replace(/^\(+|\)+$/g, "");
    else
        return null;
}

function parseReqAttributes(ofReqAttrStr) {

    // исправляем строку
    ofReqAttrStr = trimString(ofReqAttrStr);
    ofReqAttrStr = removeOpenCloseBrackets(ofReqAttrStr);

    if (!ofReqAttrStr || ofReqAttrStr===="")
        return null;

    // получаем затребованное количество
    var pos = ofReqAttrStr.indexOf(",");
    var numRequested = ofReqAttrStr.substring(0,pos);
    ofReqAttrStr = ofReqAttrStr.substring(pos+1);

    // первая часть будет атрибутом, а остаток - дочерними атрибутами
    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) {
        // исправляем строку
        parts[i] = trimString(parts[i]);
        parts[i] = removeOpenCloseBrackets(parts[i]);
        parts[i] = trimString(parts[i]);
    }

    // строим список атрибутов
    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);
            }
        }

        // строим список дочерних атрибутов
        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]);


                // получаем затребованное количество
                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];  // не-число было задано как число, передаем его в API как есть
            }
            else {
                value = Number(nvp[1]);
            }
        }
        else {
            value = nvp[1];
        }
        // специальная обработка значения NULL
        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("********Ответ********<br> " + JSON.stringify(response)) ;
    }

    function onError(response) {
        showResponse("********Ошибка********<br> " + response) ;
    }

    function formatResoponse(response) {

    }

    function printBatchResponse(batResponse) {

    }

    function printResponse(response) {

    }
```

Пример объекта ответа JavaScript onSuccesss

В этом примере показаны три переменные для объекта ответа JavaScript, списки предложений, сообщения и профиль.

`offerList` возвращает непустой список, если вы вызвали `getOffer` или `getOffersForMultipleInteractionPoints` как API или как часть ваших пакетных команд. Перед выполнением каких-либо операций с этой переменной всегда надо выполнять проверку на null.

Необходимо всегда проверять состояние ответа JavaScript `messages`.

`Profile` возвращается с непустым значением, если `getProfile` используется как API или как часть ваших пакетных команд. Если `getProfile` не используется, можно проигнорировать эту переменную. Перед выполнением каких-либо операций с этой переменной всегда надо выполнять проверку на null.

```
function onSuccess(response)
InteractAPI.ResponseTransUtil._buildResponse = function(response) {
        'use strict';

        if (!response) return null;

        var offerList = null;
        // преобразуем списки предложений в объекты JS
        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;
        // преобразуем сообщения в объекты JS
        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;
        // преобразуем nvp профилей в объекты JS
        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) ;
    };
```

Глава 9. Об API ExternalCallout

Interact предлагает расширяемую макрокоманду EXTERNALCALLOUT для использования совместно с интерактивными потоковыми диаграммами. При помощи этой макрокоманды можно выполнять пользовательские алгоритмы обмена информацией с внешними системами во время выполнения потоковой диаграммы. Например, если нужно вычислить кредитную оценку покупателя во время выполненной потоковой диаграммы, можно создать класс Java (вызов), который выполняет эту задачу, и затем использовать макрокоманду EXTERNALCALLOUT в процессе Выбрать в интерактивной потоковой диаграмме, чтобы получить кредитную оценку при помощи этого вызова.

Конфигурирование EXTERNALCALLOUT содержит два главных шага. Во-первых, нужно создать класс Java, реализующий API ExternalCallout. Во-вторых, задать нужные свойства конфигурации Marketing Platform на сервере среды выполнения в категории Interact | flowchart | ExternalCallouts.

В дополнение к информации в этом разделе доступна документация JavaDoc для API ExternalCallout на любом сервере среды выполнения Interact в каталоге Interact/docs/externalCalloutJavaDoc.

Интерфейс IAffiniumExternalCallout

API ExternalCallout входит в интерфейс IAffiniumExternalCallout. Вам нужно реализовать интерфейс IAffiniumExternalCallout, чтобы использовать макрокоманду EXTERNALCALLOUT.

Класс, реализующий IAffiniumExternalCallout, должен иметь конструктор, при помощи которого его сможет инициализировать сервер среды выполнения.

- Если в классе нет конструкторов, компилятор Java создает конструктор по умолчанию, и этого достаточно.
- Если есть конструкторы с аргументами, нужно предусмотреть общедоступный конструктор без аргументов, который будет использоваться сервером среды выполнения.

При разработке внешнего вызова помните следующее:

- При каждой оценке выражения при помощи внешнего вызова создается новый экземпляр класса. Для статических элементов в классе нужно работать с проблемами безопасности.
- Если ваш внешний вызов использует системные ресурсы, такие как файлы или соединение с базой данных, надо управлять этими соединениями. У сервера среды выполнения нет утилиты для автоматической чистки соединений.

Надо скомпилировать реализацию с помощью файла interact_externalcallout.jar, расположенного в каталоге lib вашей установки IBM Interact среды выполнения.

При помощи IAffiniumExternalCallout сервер среды выполнения может запрашивать данные у класса Java. Интерфейс состоит из четырех методов:

- getNumberOfArguments
- getValue

- initialize
- shutdown

Добавление веб-службы для использования с макрокомандой EXTERNALCALLOUT

Используйте эту процедуру, чтобы добавить веб-службу для использования макрокоманды EXTERNALCALLOUT. Макрокоманда EXTERNALCALLOUT опознает ссылки, только если вы задали соответствующие свойства конфигурации.

Процедура

В Marketing Platform для среды выполнения добавьте или определите следующие свойства конфигурации в категории Interact > flowchart > externalCallouts.

Свойство конфигурации	Значение
Категория externalCallouts	Создайте категорию для своего внешнего вызова
class	Имена классов для внешнего вызова
classpath	параметр classpath для файлов класса внешнего вызова
Категория Parameter Data	Если для внешнего вызова нужны параметры, создайте для них новые свойства конфигурации параметра и назначьте каждому свойству значение

getNumberOfArguments

Метод getNumberOfArguments возвращает число аргументов, ожидаемых классом Java, с которым вы выполняете интеграцию.

```
getNumberOfArguments()
```

Возвращаемое значение

Объект getNumberOfArguments возвращает целое число.

Пример

В приведенном ниже примере показана печать число аргументов.

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

Метод getValue выполняет базовые функции вызова и возвращает результаты.

```
getValue(audienceID, configData, arguments)
```

Метод getValue требует следующих параметров:

- **audienceID** - значение, идентифицирующее ID аудитории.
- **configData** - карта с парами ключ-значение данных конфигурации, требуемыми для вызова.
- **arguments** - аргументы, требуемые для вызова. Каждый аргумент может быть строкой, числом с двойной точностью, датой или списком значений одного из этих типов. Аргумент-список может содержать пустые значения, однако список не может содержать одновременно, например, строку и число с двойной точностью.

Проверку типа аргумента должна выполнять ваша реализация.

Если метод `getValue` завершается неудачно по какой-либо причине, он возвращает `CalloutException`.

Возвращаемое значение

Метод getValue возвращает список строк.

Пример

```
public List<String> getValue(AudienceId audienceId, Map<String,
  String> configurationData, Object... arguments) throws CalloutException
{
 Long customerId = (Long) audienceId.getComponentValue("Покупатель");
 // теперь запросим scoreQueryUtility, чтобы получить кредитный рейтинг customerId
 Double score = scoreQueryUtility.query(customerId);
 String str = Double.toString(score);
 List<String> list = new LinkedList<String>();
 list.add(str);
 return list;
}
```

initialize

Метод `initialize` вызывается один раз при запуске сервера среды выполнения. Если существуют какие-либо операции, которые могут снизить производительность во время выполнения, такие как загрузка таблицы базы данных, их надо выполнить этим методом.

```
initialize(configData)
```

Метод `initialize` требует следующего параметра:

- **configData** - карта с парами ключ - значение данных конфигурации, требуемыми для вызова.

  Interact читает эти значения из параметров внешнего вызова, определенных в категории `Interact > Flowchart > External Callouts > [Внешний вызов] > Parameter Data`.

Если метод `initialize` завершается неудачно по какой-либо причине, он возвращает `CalloutException`.

Возвращаемое значение

Нет.

Пример

```
public void initialize(Map<String, String> configurationData) throws CalloutException
{
    // configurationData содержат пары ключ - значение, определенные для среды,
    // где работает сервер
    // инициализируйте scoreQueryUtility здесь
}
```

shutdown

Метод `shutdown` вызывается один раз при закрытии сервера среды выполнения. Если для вашего вызова требуются какие-либо задачи очистки, их надо запустить в этот момент.

```
shutdown(configData)
```

Метод shutdown требует следующего параметра:

- **configData** - карта с парами ключ - значение данных конфигурации, требуемыми для вызова.

Если метод shutdown завершается неудачно по какой-либо причине, он возвращает CalloutException.

Возвращаемое значение

Нет.

Пример

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
    // Закройте scoreQueryUtility здесь
}
```

Пример API ExternalCallout

В этом примере создается внешний вызов, получающая кредитную оценку.

Создайте внешний вызов, получающий кредитную оценку.

1. Создайте файл с именем GetCreditScore.java со следующим содержимым. В этом файле предполагается, что есть класс с именем ScoreQueryUtility, который получает оценку от программы моделирования.

```
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
{
 // класс, содержащий алгоритм для запроса внешней системы о кредитной оценке покупателя
 private static ScoreQueryUtility scoreQueryUtility;
 public void initialize(Map<String, String> configurationData) throws CalloutException
 {
  // configurationData содержит пары ключ - значение для конкретной среды, в которой работает сервер
  // инициализируем scoreQueryUtility здесь
 }

 public void shutdown(Map<String, String> configurationData) throws CalloutException
 {
 // закрываем scoreQueryUtility здесь
 }

 public int getNumberOfArguments()
 {
  // не ожидаем никаких дополнительных аргументов, кроме ID покупателя
  return 0;
 }

 public List<String> getValue(AudienceId audienceId, Map<String, String> configurationData,
  Object... arguments) throws CalloutException
 {
  Long customerId = (Long) audienceId.getComponentValue("Покупатель");
  // теперь запросим у scoreQueryUtility кредитную оценку для customerId
  Double score = scoreQueryUtility.query(customerId);
  String str = Double.toString(score);
  List<String> list = new LinkedList<String>();
```

```
  list.add(str);
  return list;
 }
}
```

2. Скомпилируйте из `GetCreditScore.java` файл `GetCreditScore.class`.
3. Создайте файл JAR с именем `creditscore.jar`, содержащий `GetCreditScore.class` и остальные файлы классов, которыми он пользуется.
4. Скопируйте файл JAR в какое-нибудь положение на сервере среды выполнения, например, в `/data/interact/creditscore.jar`.
5. Создайте внешнюю ссылку с именем `GetCreditScore` и параметром classpath `/data/interact/creditscore.jar` в категории `externalCallouts` на странице Управление конфигурациями.
6. На интерактивной потоковой диаграмме этот вызов можно использовать как `EXTERNALCALLOUT ('GetCreditScore')`.

Интерфейс IInteractProfileDataService

API Profile Data Services входит в интерфейс `iInteractProfileDataService`. При помощи этого интерфейса можно импортировать иерархические данные в сеанс Interact через один или несколько внешних источников данных (таких как плоский файл, веб-служба и так далее) во время запуска сеанса Interact или изменения ID аудитории сеанса Interact.

Чтобы разработать импорт иерархических данных с помощью API Profile Data Services, нужно написать класс Java, извлекающий информацию из любого источника данных и отображает ее в объект `ISessionDataRootNode`, а затем сослаться на отображенные данные при помощи макрокоманды EXTERNALCALLOUT в процессе Выбрать на интерактивной потоковой диаграммы.

Надо скомпилировать реализацию с помощью файла `interact_externalcallout.jar`, расположенного в каталоге `lib` вашей установки IBM Interact среды выполнения.

Полную документацию Javadoc по использованию этого интерфейса смотрите в файлах *домашний_каталог_Interact*`/docs/externalCalloutJavaDoc`, открыв их в любом браузере.

Пример реализации, показывающий, как использовать Profile Data Service, включая комментированные описания реализации примера, смотрите в файле *домашний_каталог_Interact*`/samples/externalcallout/XMLProfileDataService.java`.

Примечание: Пример реализации предназначен, только чтобы использоваться в качестве примера. Не надо использовать этот пример в своей реализации.

Добавление источника данных для использования совместно с Profile Data Services

При помощи этой процедуры можно добавить источник данных для использования совместно с Profile Data Services.

Об этой задаче

Макрокоманда EXTERNALCALLOUT опознает источник данных для импорта иерархических данных Profile Data Services, только если вы задали соответствующие свойства конфигурации.

Процедура

В Marketing Platform для среды выполнения добавьте или определите следующие свойства конфигурации в категории Interact > profile > Audience Levels > [AudienceLevelName] > Profile Data Services.

Свойство конфигурации	Значение
Категория Имя новой категории	Имя источника данных, который вы определяете. Нужно задать имя, уникальное среди источников данных для того же самого уровня аудитории.
enabled	Указывает, включен ли источник данных для уровня аудитории, на котором он определен.
className	Полное имя класса источника данных, реализующего IInteractProfileDataService
classPath	Параметр classpath для файлов класса Profile Data Services. Если его опустить, по умолчанию используется путь классов сервера прикладных программ.
Категория priority	Приоритет этого источника данных в пределах этого уровня аудитории. Требуется уникальное значение среди всех источников данных для каждого уровня аудитории. (Это значит, что если для некоторого источника данных задан приоритет 100, у остальных источников данных в пределах этого уровня аудитории не может быть приоритета 100.)

Интерфейс IParameterizableCallout

API Parameterizable входит в интерфейс IParameterizableCallout.

Этот базовый интерфейс представленных интерфейсов API, которые могут принимать параметры из конфигурации через Marketing Platform. Поскольку это базовый интерфейс, его не нужно внедрять непосредственно. Параметры извлекаются с дочерних узлов узла Parameter Data из категории, в которой есть ссылка на эту реализацию. В приведенном ниже примере ESB - это пользовательская реализация службы данных профиля, которая, в свою очередь, реализует интерфейс IParameterizableCallout. Параметры endPoint и login и их значения передаются в этот класс реализации, когда механизм Interact пытается инициализировать его или завершить его работу.

```
Profile Data Services
...ESB
   ...Parameter Data
      ...endPoint
      ...login
```

Интерфейс состоит из двух методов:

- initialize
- shutdown

initialize

Метод initialize инициализирует данный класс реализации.

```
void initialize(java.util.Map<java.lang.String,java.lang.String> configurationData)
                throws CalloutException
```

Метод initialize требует следующий параметр:

- **configurationData** - карта с парами значение - имя параметров, сконфигурированных пользователями

Генерирует

```
CalloutException
```

shutdown

Метод shutdown закрывает этот класс реализации.

```
void shutdown(java.util.Map<java.lang.String,java.lang.String> configurationData)
              throws CalloutException
```

Метод shutdown требует следующего параметра:

- **configurationData** - карта с парами значение - имя параметров, сконфигурированных пользователями

Генерирует

```
CalloutException
```

Интерфейс ITriggeredMessageAction

API Triggered Message Action входит в интерфейс ITriggeredMessageAction. При помощи этого интерфейса можно получить и задать имя данного экземпляра.

Интерфейс ITriggeredMessageAction служит базовым интерфейсом для других интерфейсов, и его не нужно реализовать непосредственно.

Интерфейс состоит из двух методов:

- getName
- setName

getName

Метод getName возвращает имя экземпляра ITriggeredMessageAction.

```
java.lang.String getName()
```

setName

Метод setName задает имя экземпляра ITriggeredMessageAction.

```
void setName(java.lang.String name)
```

При инициализации класса реализации этого интерфейса Interact задает имя интерфейса с именем, заданным в конфигурации пользовательского интерфейса.

В следующем примере имя шлюза - InteractLog.

```
triggeredMessage
           ...gateways
              ...InteractLog
```

Метод setName требует следующего параметра:

- name - имя, которое вы хотите задать для экземпляра ITriggeredMessageAction.

Интерфейс IChannelSelector

API выбора каналов входит в интерфейс IChannelSelector. При помощи этого интерфейса можно выбрать исходящие каналы с учетом свойств отправляемого предложения и атрибутов сеанса.

Пример реализации, показывающий, как использовать действия инициированных сообщений, включая комментированные описания реализации примера, смотрите в файле *домашний_каталог_Interact*/samples/triggeredmessage/SampleChannelSelector.java.

Примечание: Пример реализации предназначен, только чтобы использоваться в качестве примера. Не надо использовать этот пример в своей реализации.

Надо попытаться использовать эту реализацию вместо того, чтобы писать собственную.

Интерфейс состоит из одного метода:

- selectChannels

selectChannels

Метод selectChannels выбирает исходящие каналы, в которые должно быть направлено переданное предложение по интерфейсу IChannelSelector.

```
java.util.List<java.lang.String> selectChannels
                    (java.util.Map<java.lang.String,java.util.Map<java.lang.String,
                                   java.lang.Object>> availableChannels,
                                   com.unicacorp.interact.api.Offer offer,
                                   com.unicacorp.interact.treatment.
                                   optimization.IInteractSessionData sessionData)
```

Interact пытается отправить это предложение по всем возвращенным каналам.

Метод selectChannels требует следующих параметров:

- **availableChannels** - карта доступных исходящих каналов, которые сконфигурированы в пользовательском интерфейсе инициированных сообщений в параметрах времени разработки Interact. В каждой записи карты ключ - это имя канала, а значение - сконфигурированные параметры для этого канала во время разработки Interact. Порядок перечисления этой карты соответствует порядку, определенному в этом пользовательском интерфейсе. Если в пользовательском интерфейсе инициированных сообщений используется предпочтительный канал профиля, он заменяется фактическим каналом перед вызовом этого метода. Кроме того, если тот же самый канал несколько раз фигурирует в пользовательском интерфейсе, сохраняется только вхождение с самым высоким приоритетом, а все дубликаты удаляются.
- **offer** - доставляемое предложение
- **sessionData** - атрибуты, которые в данный момент хранятся в связанном сеансе Interact

Интерфейс IDispatcher

API диспетчера входит в интерфейс IDispatcher. Этот интерфейс отправляет предложения на шлюзы назначения.

Поскольку существует только один экземпляр этого класса для каждого сконфигурированного диспетчера, реализация этого интерфейса не может использовать информацию о состоянии с точки зрения Interact.

Пример реализации, показывающий, как использовать действия инициированных сообщений, включая комментированные описания реализации примера, смотрите в файле *домашний_каталог_Interact*/samples/triggeredmessage/SampleDispatcher.java.

Примечание: Пример реализации предназначен, только чтобы использоваться в качестве примера. Не надо использовать этот пример в своей реализации.

Надо попытаться использовать эту реализацию вместо того, чтобы писать собственную.

Интерфейс состоит из одного метода:

- dispatch

dispatch

Метод dispatch отправляет предложения целевым шлюзам через интерфейс IDispatcher.

```
boolean dispatch(java.lang.String channel,
            java.lang.String gatewayName,
            java.util.Collection<com.unicacorp.interact.api.Offer> offers,
            com.unicacorp.interact.api.NameValuePair[] profileData)
              throws com.unicacorp.interact.exceptions.InteractException
```

Как только исходящие каналы выбраны для предложения-кандидата, Interact пытается отправить предложения-кандидаты обработчикам, связанным с этим каналом. Обработчики, для которых предпринимается эта попытка, определяются на основе их приоритетов от высокого до низкого. Для каждого обработчика Interact вызывает этот метод сконфигурированного диспетчера. От реализации этого экземпляра диспетчера зависит, как направить предложение на целевой шлюз, сконфигурированный в этом же обработчике. Если несколько предложений направляются одному и тому же обработчику как результат одной и той же оценки инициированного сообщения, Interact пытается отправить все эти предложения в одном пакете.

Метод dispatch требует следующих параметров:

- **channel** - исходящий канал, на который отправляются эти предложения
- **gatewayName** - имя целевого шлюза
- **offers** - предложения, которые будут отправлены шлюзу в пакете
- **profileData** - атрибуты профиля, заполненные IGateway.validate и переданные IGateway.deliver

Возвращаемое значение

Метод dispatch возвращает значение, указывающее, была ли отправка выполнена успешно или неудачно

Генерирует

com.unicacorp.interact.exceptions.InteractException

Интерфейс IGateway

API Gateway входит в интерфейс `IGateway`. Этот интерфейс получает предложения от Interact и отправляет предложения по назначению.

Каждая реализация этого интерфейса передает информацию на свой объект назначения. Объект назначения должен выполнить необходимое преобразование данных, задание атрибутов и тому подобные задачи, связанные с конкретным объектом назначения.

Пример реализации, показывающий, как использовать действия инициированных сообщений, включая комментированные описания реализации примера, смотрите в файле *домашний_каталог_Interact*`/samples/triggeredmessage/SampleOutboundGateway.java`.

Примечание: Пример реализации предназначен, только чтобы использоваться в качестве примера. Не надо использовать этот пример в своей реализации.

Интерфейс состоит из двух методов:

- `deliver`
- `validate`

deliver

Метод `deliver` вызывается, чтобы отправить предложение или предложения месту назначения через интерфейс `IGateway`.

```
void deliver(java.util.Collection<com.unicacorp.interact.api.Offer> offers,
                    com.unicacorp.interact.api.NameValuePair[] profileData,
                    java.lang.String channel)
```

Метод `deliver` требует следующих параметров:

- **offers** - отправляемое предложение
- **profileData** - атрибуты профиля, которые метод validate заполняет в `parameterMap`
- **channel** - исходящий канал, на который будут отправлены эти предложения

validate

Метод `validate` проверяет предложения-кандидаты в интерфейсе `IGateway`.

```
java.util.Collection<com.unicacorp.interact.api.Offer> validate
    (com.unicacorp.interact.treatment.optimization.
     IInteractSessionData sessionData,
           java.util.Collection<com.unicacorp.interact.api.Offer> candidateOffers,
           java.util.Map<java.lang.String,java.lang.Object> parameterMap,
           java.lang.String channel)
```

Механизм Interact вызывает этот метод для проверки предложений-кандидатов. Реализация этого метода должна проверить предложения, атрибуты предложений и атрибуты сеансов на соответствие требованиям назначения, чтобы определить, какое предложение или предложения могут быть посланы через этот шлюз. Кроме того, можно добавить необходимые параметры в передаваемую карту, которая передается назад методу доставки.

Метод `validate` требует следующих параметров:

- **sessionData** - атрибуты, которые в данный момент хранятся в связанном сеансе Interact

- **candidateOffers** - предложения Interact, выбранные на основе метода выбора предложения, его параметров и других факторов. Эти предложения пригодны для доставки с точки зрения Interact, но должны быть еще рассмотрены шлюзом.
- **parameterMap** - карта, которую реализация этого метода должна использовать для передачи параметров своему методу доставки
- **channel** - исходящий канал, на который будут отправлены эти предложения

Глава 10. Утилиты IBM Interact

В этом разделе описываются утилиты администрирования, доступные с Interact.

Запуск утилиты внедрения (runDeployment.sh/.bat)

Инструмент командной строки runDeployment позволяет внедрить интерактивный канал для конкретной группы серверов из командной строки, используя параметры, предоставленные в файле deployment.properties, в котором перечислены все возможные параметры и который доступен в том же положении, что и сам инструмент runDeployment. Возможность запуска внедрения интерактивного канала из командной строки особенно полезна при использовании возможности OffersBySQL. Например, можно сконфигурировать пакетную потоковую диаграмму Campaign для периодического запуска. По завершении выполнения потоковой диаграммы можно вызвать триггер, чтобы инициализировать внедрение предложений в таблице OffersBySQL с помощью этого инструмента командной строки.

Описание

Инструмент командной строки runDeployment, автоматически установленный на сервере среды разработки Interact, можно найти в следующем положении:

домашний_каталог_Interact/interactDT/tools/deployment/runDeployment.sh (или runDeployment.bat для сервера Windows)

Единственный аргумент, передаваемый в эту команду - это положение файла deployment.properties, представляющего все возможные параметры, которые необходимы для внедрения сочетания групп серверов среды выполнения/разработки. Для справки представлен образец файла.

Примечание: Прежде чем использовать утилиту runDeployment, необходимо изменить ее в любом текстовом редакторе, чтобы задать положение среды выполнения Java на сервере. Например, можно указать как путь *домашний_каталог_Interact*/jre или *домашний_каталог_Platform*/jre, если какой-то из этих каталогов содержит среду выполнения Java для использования утилитой. Вместо этого можно предоставить путь к любой среде выполнения Java, поддерживаемой для использования с этим выпуском продуктов IBM .

Использование утилиты runDeployment в защищенной среде (SSL)

Чтобы использовать утилиту runDeployment при включенной защите для сервера Interact (то есть и при соединении через порт SSL), необходимо добавить свойство Java для склада доверенных сертификатов следующим образом:

1. При редактировании файла deployment.properties для внедрения вашего интерактивного канала измените свойство deploymentURL для использования защищенного URL SSL, как в следующем примере:

   deploymentURL=https://<*ХОСТ*>.<*ДОМЕН*>:<*ПОРТ*>/Campaign/interact/InvokeDeploymentServlet

2. Используя любой текстовый редактор, измените сценарий `runDeployment.sh` или `runDeployment.bat`, чтобы добавить следующий аргумент в строку, начинающуюся с ${JAVA_HOME}:

   `-Djavax.net.ssl.trustStore=<`*`TrustStorePath`*`>`

   Например, после добавления аргумента склада доверенных сертификатов эта строка могла бы выглядеть так:

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

   Замените <*TrustStorePath*> на путь к настоящему складу доверенных сертификатов SSL.

Выполнение утилиты

После изменения утилиты для определения среды выполнения Java и настройки копии файла `deployment.properties`, чтобы он соответствовал ваше среде, можно запустить эту утилиту следующей командой:

домашний_каталог_Interact`/interactDT/tools/deployment/runDeployment.sh` *deployment.properties*

Замените *домашний_каталог_Interact* на настоящее значение каталога установки среды разработки Interact, а *deployment.properties* - на настоящие путь и имя файла свойств, который вы настроили для этого внедрения.

Пример файла `deployment.properties`

Пример файла `deployment.properties` содержит закомментированный список всех параметров, которые нужно настроить для соответствия вашей среде. Образец файла содержит также комментарии с объяснениями, что означает каждый параметр и почему может потребоваться настроить его с конкретным значением.

```
##############################################################################
#
# Следующие свойства передаются в программу InvokeDeploymentClient.
# Программа будет искать параметр deploymentURL. Программа опубликует
# требование для этого URL; все другие параметры будут параметрами в
# этом требовании. Затем программа проверит состояние внедрения и
# вернется назад, если внедрение находится в окончательном состоянии (или если
# достигнуто заданное значение времени ожидания waitTime).
#
# вывод программы будет дан в следующем формате:
# <СОСТОЯНИЕ> : <Разные подробности>
#
# где состоянием может быть одно из следующих:
# ERROR
# RUNNING
# SUCCESS
#
# Разные подробности - это данные, которые обычно могли бы заполнять область сообщения о состоянии
# в графическом пользовательском интерфейсе внедрения сводной страницы ИК. ПРИМЕЧАНИЕ:
# в Разных подробностях могут встречаться теги HTML
#
##############################################################################

##############################################################################
# deploymentURL: URL для сервлета InvokeDeployment из среды разработки
# Interact.  Надо использовать следующий формат:
# http://хост_dt:порт/кампания/interact/InvokeDeploymentServlet
##############################################################################
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet
```

```
##############################################################################
# dtLogin: имя входа в систему, используемое для среды разработки, если
# вы хотели внедрить ИК через графический пользовательский интерфейс
# внедрения на сводной странице ИК.
##############################################################################
dtLogin=asm_admin

##############################################################################
# dtPW:  это пароль для dtLogin
##############################################################################
dtPW=

##############################################################################
# icName:  имя интерактивного канала, который вы хотите внедрить
##############################################################################
icName=ic1

##############################################################################
# partition:  имя раздела
##############################################################################
partition=partition1

##############################################################################
# request:  тип требования, которое в настоящее время должен выполнить этот
# инструмент: есть две возможности. Если значение - "deploy", будет
# выполняться внедрение. При всех остальных значениях инструмент просто вернет
# состояние последнего внедрения указанного интерактивного канала.
##############################################################################
request=deploy

##############################################################################
# serverGroup:  имя группы серверов, которая выбрана для внедрения интерактивного
# канала.
##############################################################################
serverGroup=defaultServerGroup

##############################################################################
# serverGroupType:  указывает, для какой группы серверов будет выполняться это
# внедрение - для группы производственных серверов или для группы серверов
# тестирования. 1 означает производство, 2 - тест.
##############################################################################
serverGroupType=1

##############################################################################
# rtLogin:  учетная запись, используемая при аутентификации для группы серверов,
# где выполняется внедрение.
##############################################################################
rtLogin=asm_admin

##############################################################################
# rtPW:  пароль для rtLogin
##############################################################################
rtPW=

##############################################################################
# waitTime:  после передачи инструментом требования внедрения он проверит
# состояние внедрения. Если внедрение не завершено (или завершено неудачно),
# инструмент продолжит опрашивать систему на это состояние, пока не будет
# достигнуто состояние Выполнено ИЛИ пока не будет достигнуто заданное время
# ожидания waitTime (в секундах)
##############################################################################
waitTime=5

##############################################################################
# pollTime: Если состояние внедрения все еще - Выполняется, инструмент продолжит
# проверять состояние. Между проверками состояния он переходит в спящий режим
```

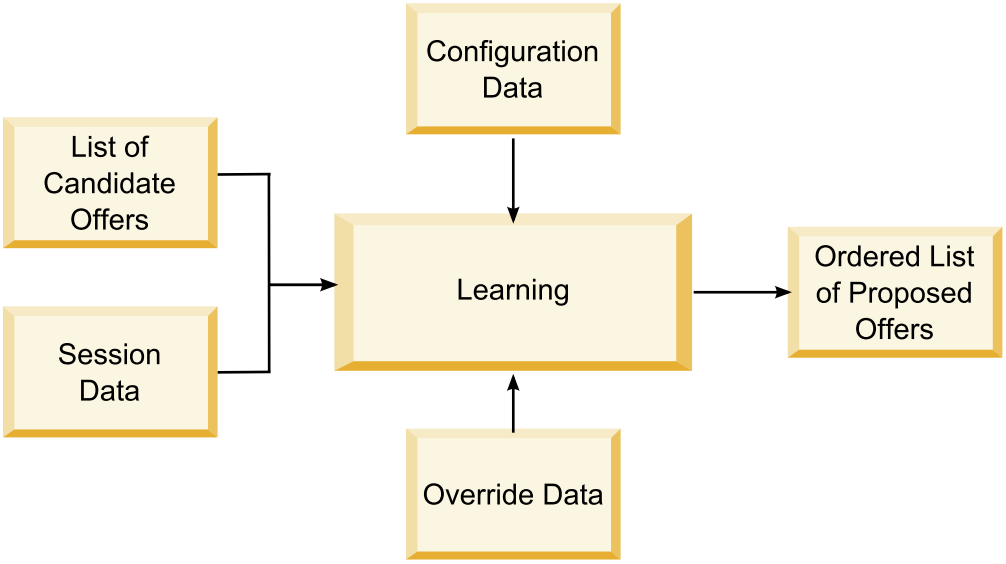
```
# на время, заданное в секундах параметром pollTime.
################################################################################
pollTime=3

################################################################################
# global: при значении false инструмент НЕ внедрит глобальные параметры.
# При недоступности этого свойства глобальные параметры все же внедряются.
################################################################################
global=true
```

Глава 11. Об API обучения

Interact содержит модуль обучения, который использует наивный байесовский алгоритм для отслеживания действий посетителей и представления оптимальных предложений (в терминах принятия). Вы можете реализовать этот же интерфейс Java со своими алгоритмами, используя API обучения для создания своего собственного модуля обучения.

Примечание: Если используется внешнее обучение, отчеты примера, относящиеся к обучению (Подробности обучения для интерактивных предложений и Анализ подъема для интерактивного сегмента) не возвращают допустимые данные.

На простейшем уровне API обучения предоставляет методы для сбора данных в среде выполнения, возвращающие упорядоченный список рекомендованных предложений.



Из Interact можно собрать следующие данные

- Данные контактов для предложения
- Данные восприятия для предложения
- Все данные сеанса
- Данные предложений специально для Campaign
- Свойства конфигурации, определенные в категории `learning` для среды разработки и в категории `offerserving` для среды выполнения

Эти данные можно использовать в ваших алгоритмах для создания списков представленных предложений. Затем можно возвратить список рекомендованных предложений в порядке убывания предпочтительности.

Хотя на диаграмме это не показано, при помощи API обучения можно также собирать данные для своей реализации обучения. Эти данные можно хранить в памяти или записать их в файл или базу данных для последующего анализа.

После создания классов Java можно преобразовать их в файлы JAR. Как только вы создали файлы JAR, необходимо также сконфигурировать среду выполнения для распознавания вашего внешнего модуля обучения, отредактировав свойства

конфигурации. Классы Java или файлы JAR надо скопировать на каждый сервер среды выполнения, который использует ваш внешний модуль обучения.

Кроме информации в этом руководстве, на каждом сервере среды выполнения в каталоге `Interact/docs/learningOptimizerJavaDoc` доступен JavaDoc для API оптимизатора обучения.

Надо скомпилировать реализацию с помощью файла `interact_learning.jar`, расположенного в каталоге `lib` вашей установки Interact среды выполнения.

При написании собственной реализации обучения необходимо иметь в виду следующие правила.

- Производительность критична.
- Реализация должна работать в многопотоковой среде и быть потокозащищенной.
- Реализация должна управлять внешними ресурсами с учетом режимов отказа и производительности.
- Используйте надлежащим образом исключительные ситуации, запись в журнал (log4j) и память.

Конфигурирование среды выполнения для распознавания внешних модулей обучения

API обучения Java можно использовать для написания вашего собственного модуля обучения. Среду выполнения надо сконфигурировать для распознавания вашей утилиты обучения в Marketing Platform.

Об этой задаче

Чтобы изменения вступили в силу, нужно перезапустить сервер среды выполнения Interact.

Процедура

1. В Marketing Platform для среды выполнения отредактируйте следующие свойства конфигурации в категории `Interact > offerserving`. Свойства конфигурации для API оптимизатора обучения содержатся в категории `Interact > offerserving > External Learning Config`.

Свойство конфигурации	Значение
`optimizationType`	**ExternalLearning**
`externalLearningClass`	имя класса для внешнего обучения
`externalLearningClassPath`	Путь к классу или файлам JAR на сервере среды выполнения для внешнего обучения. Если используется группа серверов и все серверы среды выполнения ссылаются на один и тот же экземпляр Marketing Platform, на каждом сервере должна быть копия этого класса или файлы JAR в том же каталоге.

2. Чтобы изменения вступили в силу, перезапустите сервер среды выполнения Interact.

Интерфейс ILearning

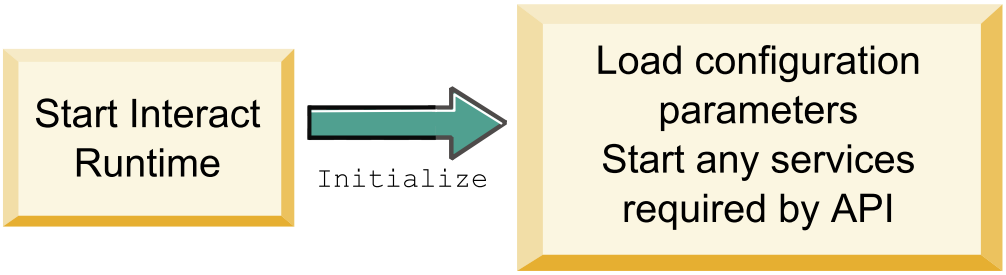
API обучения построен на основе интерфейса `ILearning`. Для поддержки настроенного алгоритма вашего модуля обучения необходимо реализовать интерфейс `ILearning`.

Интерфейс `ILearning` позволяет, в частности, собирать данные из среды выполнения для вашего класса Java и отправлять список рекомендованных предложений обратно на сервер среды выполнения.

initialize

Метод `initialize` вызывается один раз при запуске сервера среды выполнения. Если существуют какие-либо операции, которые не требуется повторять, и они могут снизить производительность во время выполнения, такие как загрузка статических данных из таблицы базы данных, их надо выполнить этим методом.

```
initialize(ILearningConfig config, boolean debug)
```



- **config** - объект `ILearningConfig`, определяющий все свойства конфигурации, которые относятся к обучению.
- **debug** - логическое значение. Значение `true` задает отладочный уровень записи в журнал для системы среды выполнения. Для оптимальных результатов выберите это значение до начала записи в журнал.

Если метод `initialize` завершается неудачно по какой-либо причине, он инициирует `LearningException`.

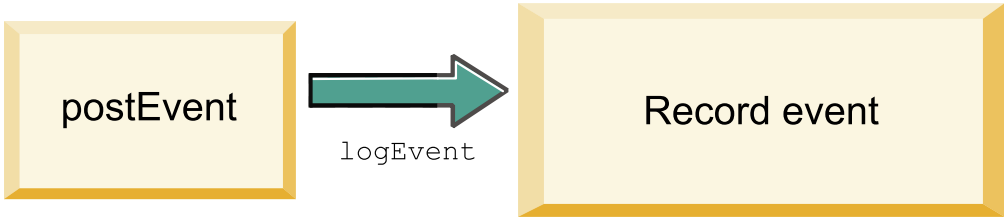
Возвращаемое значение

Нет.

logEvent

Метод `logEvent` вызывается сервером среды выполнения каждый раз, когда API Interact отправляет событие, которое сконфигурировано для записи в журнал как контакт или ответ. Этот метод служит для записи данных о контакте или ответе в базу данных или файл для создания отчетов или для обучения. Например, если вы хотите алгоритмически выяснять на основе критериев вероятность того, что клиент примет предложение, используйте этот метод для записи данных в журнал.

```
logEvent(ILearningContext context,
             IOffer offer,
             IClientArgs clientArgs,
             IInteractSession session,
             boolean debug)
```



- **context** - объект `ILearningContext`, в котором определяется контекст обучения события, например, контакт, принятие или отклонение.
- **offer** - объект `IOffer`, где определяется предложение, с которым связано записываемое в журнал событие.
- **clientArgs** - объект `IClientArgs`, в котором определяются любые параметры. В настоящее время для `logEvent` нет обязательных параметров `clientArgs`, так что этот объект может быть пуст.
- **session** - объект `IInteractSession`, в котором определяются все данные сеанса.
- **debug** - логическое значение. Значение `true` задает отладочный уровень записи в журнал для системы среды выполнения. Для оптимальных результатов выберите это значение до начала записи в журнал.

Если метод `logEvent` завершается неудачно, он инициирует `LearningException`.

Возвращаемое значение

Нет.

optimizeRecommendList

Метод `optimizeRecommendList` должен получать список рекомендованных предложений и данные сеанса и возвращать список, содержащий требуемое число предложений. Метод `optimizeRecommendList` должен тем или иным способом заказывать предложения с применением вашего алгоритма обучения. Список предложений должен заказываться так, чтобы предложения, которые нужно представить первыми, находились в начале списка. Например, если алгоритм обучения дает низшие оценки лучшим предложениям, предложения должны сортироваться в порядке 1, 2, 3. Если же алгоритм обучения дает высшие оценки лучшим предложениям, предложения должны сортироваться в порядке 100, 99, 98.

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

Метод `optimizeRecommendList` требует следующих параметров:

- **recList** - список объектов процедуры (предложений), рекомендованных средой выполнения.

- **clientArg** - объект IClientArgs, содержащий, как минимум число предложений, запрошенных средой выполнения.
- **session** - объект IInteractSession, содержащий все данные сеанса.
- **debug** - логическое значение. Значение `true` задает отладочный уровень записи в журнал для системы среды выполнения. Для оптимальных результатов выберите это значение до начала записи в журнал.

Если метод `optimizeRecommendList` завершается неудачно, он инициирует `LearningException`.

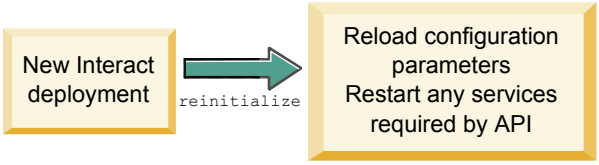
Возвращаемое значение

Метод `optimizeRecommendList` возвращает список объектов `ITreatment`.

reinitialize

Среда выполнения вызывает метод `reinitialize` при каждом новом внедрении. Этот метод передает все данные конфигурации обучения. Если у вас есть какие-либо службы, которые требуются для API обучения, читающего свойства конфигурации, этот интерфейс должен перезапустить эти службы.

```
reinitialize(ILearningConfig config,
                  boolean debug)
```



- **config** - объект `ILearningConfig`, содержащий все свойства конфигурации.
- **debug** - логическое значение. Значение `true` задает отладочный уровень записи в журнал для системы среды выполнения. Для оптимальных результатов выберите это значение до начала записи в журнал.

Если метод `logEvent` завершается неудачно, он инициирует `LearningException`.

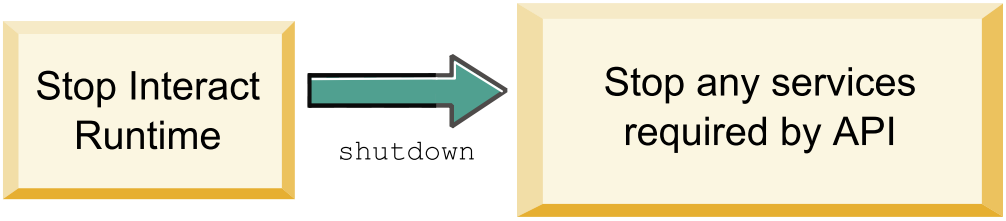
Возвращаемое значение

Нет.

shutdown

Среда выполнения вызывает метод `shutdown`, когда закрывается сервер среды выполнения. Если для вашего модуля обучения требуются какие-либо задачи очистки, их надо запустить в этот момент.

```
shutdown(ILearningConfig config, boolean debug)
```



Метод shutdown требует следующих параметров.

- **config** - объект ILearningConfig, определяющий все свойства конфигурации.
- **debug** - логическое значение. Значение true задает отладочный уровень записи в журнал для системы среды выполнения. Для оптимальных результатов выберите это значение до начала записи в журнал.

Если метод shutdown завершается неудачно по какой-либо причине, он инициирует LearningException.

Возвращаемое значение

Нет.

Интерфейс IAudienceID

Интерфейс IAudienceID поддерживает интерфейс IInteractSession. Это интерфейс для ID аудитории. Поскольку ваш ID аудитории может состоять из нескольких частей, этот интерфейс позволяет обращаться ко всем элементам ID аудитории, а также имени уровня аудитории.

getAudienceLevel

Метод getAudienceLevel возвращает уровень аудитории.

```
getAudienceLevel()
```

Возвращаемое значение

Метод getAudienceLevel возвращает строку, определяющую уровень аудитории.

getComponentNames

Метод getComponentNames получает набор имен компонентов, составляющих ID аудитории. Например, если ваш ID аудитории состоит из значений customerName и accountID, getComponentNames возвратит набор, содержащий строки customerName и accountID.

```
getComponentNames()
```

Возвращаемое значение

Набор строк, содержащих имена компонентов ID аудитории.

getComponentValue

Метод getComponentValue возвращает значение для указанного компонента.

```
getComponentValue(String componentName)
```

- **componentName** - строка, определяющая имя компонента, для которого вы хотите получить значение. Регистр символов в строке не учитывается.

Возвращаемое значение

Метод getComponentValue возвращает объект, определяющий значение компонента.

IClientArgs

Интерфейс IClientArgs поддерживает интерфейс ILearning. Этот интерфейс - абстракция, охватывающая все данные, который передаются серверу от точки контакта, еще не охваченной данными сеанса. Например, число предложений, затребованных методом getOffers API Interact. Эти данные хранятся в карте.

getValue

Метод getValue возвращает значение затребованного элемента карты.

```
getValue(int clientArgKey)
```

Следующие элементы требуются на карте.

- **1** - NUMBER_OF_OFFERS_REQUESTED. Число предложений, затребованных методом getOffers API Interact API. Эта константа возвращает целое число.

Возвращаемое значение

Метод getValue возвращает объект, определяющий значение требуемой постоянной карты.

IInteractSession

Интерфейс IInteractSession поддерживает интерфейс ILearning. Это интерфейс для текущего сеанса в среде выполнения.

getAudienceId

Метод getAudienceId возвращает объект AudienceID. Используйте интерфейс IAudienceID для извлечения значений.

```
getAudienceId()
```

Возвращаемое значение

Метод getAudienceId возвращает объект AudienceID.

getSessionData

Метод getSessionData возвращает немодифицируемую карту данных сеанса, где ключом является имя переменной сеанса. Имя переменной сеанса всегда задается в верхнем регистре. Для извлечения значений используйте интерфейс IInteractSessionData.

```
getSessionData()
```

Возвращаемое значение

Метод getSessionData возвращает объект IInteractSessionData.

Интерфейс IInteractSessionData

Интерфейс IInteractSessionData поддерживает интерфейс ILearning. Это интерфейс к данным сеанса среды выполнения для текущего посетителя. Данные сеанса хранятся в виде списка пар имя - значение. С помощью этого интерфейса можно также изменять значения данных в сеансе среды выполнения.

getDataType

Метод getDataType возвращает тип данных для указанного имени параметра.

```
getDataType(string parameterName)
```

Возвращаемое значение

Метод getDataType возвращает объект InteractDataType. IntearctDataType - это перечисляемый объект Java, который может принимать значения Unknown, String, Double, Date или List.

getParameterNames

Метод getParameterNames возвращает набор всех имен данных в текущем сеансе.

```
getParameterNames()
```

Возвращаемое значение

Метод getParameterNames возвращает набор имен, для которых были заданы значения. Каждое имя в этом наборе может быть передано в getValue(String) для возврата значения.

getValue

Метод getValue возвращает значение объекта, соответствующее указанному parameterName. Объектом может быть строка, число двойной точности или дата.

```
getValue(parameterName)
```

Метод getValue требует следующего параметра:

- **parameterName** - строка, определяющая имя пары значение-имя данных сеанса.

Возвращаемое значение

Метод getValue возвращает объект, содержащий значение названного параметра.

setValue

Метод setValue позволяет задать значение для указанного parameterName. Значение может имени тип String, Double или Date.

```
setValue(string имя_параметра, object значение)
```

Метод setValue требует следующих параметров:

- **parameterName** - строка, определяющая имя пары значение-имя данных сеанса.
- **value** - объект, определяющий значение указанного параметра.

Возвращаемое значение

Нет.

ILearningAttribute

Интерфейс ILearningAttribute поддерживает интерфейс ILearningConfig. Это интерфейс для атрибутов обучения, определенных в свойствах конфигурации в категории learningAttributes.

getName

Метод getName возвращает имя атрибута обучения.

```
getName()
```

Возвращаемое значение

Метод getName возвращает строку, определяющую имя атрибута обучения.

ILearningConfig

Интерфейс ILearningConfig поддерживает интерфейс ILearning. Это интерфейс к свойствам конфигурации для обучения. Все перечисленные ниже методы возвращают значение свойства.

Интерфейс состоит из 15 методов:

- **getAdditionalParameters** - возвращает карту дополнительных свойств, определенных в категории External Learning Config
- **getAggregateStatsIntervalInMinutes** - возвращает целое число
- **getConfidenceLevel** - возвращает целое число
- **getDataSourceName** - возвращает строку
- **getDataSourceType** - возвращает строку
- **getInsertRawStatsIntervalInMinutes** - возвращает целое число
- **getLearningAttributes** - возвращает список объектов ILearningAttribute
- **getMaxAttributeNames** - возвращает целое число
- **getMaxAttributeValues** - возвращает целое число
- **getMinPresentCountThreshold** - возвращает целое число
- **getOtherAttributeValue** - возвращает строку
- **getPercentRandomSelection** - возвращает целое число
- **getRecencyWeightingFactor** - возвращает число с плавающей запятой
- **getRecencyWeightingPeriod** - возвращает целое число
- **isPruningEnabled** - возвращает логическое значение

ILearningContext

Интерфейс ILearningContext поддерживает интерфейс ILearning.

getLearningContext

Метод getLearningContext возвращает константу, указывающую является ли объект сценарием контакта, принятия или отклонения.

```
getLearningContext()
```

- **1**-LOG_AS_CONTACT
- **2**-LOG_AS_ACCEPT
- **3**-LOG_AS_REJECT

4 и 5 зарезервированы для будущего использования.

Возвращаемое значение

Метод getLearningContext возвращает целое число.

getResponseCode

Метод getResponseCode возвращает код ответа, назначенный этому предложению. Это значение должно существовать в таблице UA_UsrResponseType в системных таблицах Campaign.

```
getResponseCode()
```

Возвращаемое значение

Метод getResponseCode возвращает строку, определяющую код ответа.

IOffer

Интерфейс IOffer поддерживает интерфейс ITreatment. Это интерфейс для объекта предложения, определенного в среде разработки. Используйте интерфейс IOffer, чтобы собрать подробности предложения из среды выполнения.

getCreateDate

Метод getCreateDate возвращает дату создания предложения.

```
getCreateDate()
```

Возвращаемое значение

Метод getCreateDate возвращает дату, когда было создано предложение.

getEffectiveDateFlag

Метод getEffectiveDateFlag возвращает число, определяющее дату вступления в силу предложения.

```
getEffectiveDateFlag()
```

- **0** - дата вступления в силу - это абсолютная дата, такая как 15 марта 2010 года.
- **1** - дата вступления в силу - это дата рекомендации.

Возвращаемое значение

Метод getEffectiveDateFlag возвращает целое число, определяющее дату вступления в силу предложения.

getExpirationDateFlag

Метод getExpirationDateFlag возвращает целое значение, описывающее дату истечения срока предложения.

```
getExpirationDateFlag()
```

- **0** - абсолютная дата, например 15 марта 2010 года.
- **1** - некоторое число дней после рекомендации, например, 14.
- **2** - конец месяца после рекомендации. Если предложение представлено 31 марта, это предложение истекает в тот же день.

Возвращаемое значение

Метод getExpirationDateFlag возвращает целое, описывающее дату истечения срока предложения.

getOfferAttributes

Метод `getOfferAttributes` возвращает атрибуты предложения, определенные для этого предложения, как объект `IOfferAttributes`.

```
getOfferAttributes()
```

Возвращаемое значение

Метод `getOfferAttributes` возвращает объект `IOfferAttributes`.

getOfferCode

Метод `getOfferCode` возвращает код предложения, как он определен в Campaign.

```
getOfferCode()
```

Возвращаемое значение

Метод `getOfferCode` возвращает объект `IOfferCode`.

getOfferDescription

Метод `getOfferDescription` возвращает описание предложения, определенного в Campaign.

```
getOfferDescription()
```

Возвращаемое значение

Метод `getOfferDescription` возвращает строку.

getOfferID

Метод `getOfferID` возвращает ID предложения, как он определен в Campaign.

```
getOfferID()
```

Возвращаемое значение

Метод `getOfferID` возвращает значение типа long, определяющее ID предложения.

getOfferName

Метод `getOfferName` возвращает имя предложения, как оно определено в Campaign.

```
getOfferName()
```

Возвращаемое значение

Метод `getOfferName` возвращает строку.

getUpdateDate

Метод `getUpdateDate` возвращает последнего изменения предложения.

```
getUpdateDate()
```

Возвращаемое значение

Метод `getUpdateDate` возвращает дату, определяющую, когда последний раз было изменено это предложение.

IOfferAttributes

Интерфейс `IOfferAttributes` поддерживает интерфейс `IOffer`. Это интерфейс для атрибутов предложения, определенных для предложения в среде разработки. Используйте интерфейс `IOfferAttributes`, чтобы собрать атрибуты предложения из среды выполнения.

getParameterNames

Метод `getParameterNames` возвращает список имен параметров предложения.

```
getParameterNames()
```

Возвращаемое значение

Метод `getParameterNames` возвращает набор, определяющий список имен параметров предложения.

getValue

Метод `getValue` возвращает объект, определяющий значение атрибута предложения.

```
getValue(String parameterName)
```

Метод `getValue` возвращает значение данного атрибута предложения.

Возвращаемое значение

Интерфейс IOfferCode

Интерфейс `IOfferCode` поддерживает интерфейс `ILearning`. Это интерфейс для кода предложения, определенного для предложения в среде разработки. Код предложения может состоять из одной или нескольких строк. Используйте интерфейс `IOfferCode`, чтобы собрать код предложения из среды выполнения.

getPartCount

Метод `getPartCount` метод возвращает число частей, составляющих код предложения.

```
getPartCount()
```

Возвращаемое значение

Метод `getPartCount` возвращает целое число, определяющее число частей кода предложения.

getParts

Метод `getParts` получает немодифицируемый список частей кода предложения. method gets an unmodifiable list of the offer code parts.

```
getParts()
```

Возвращаемое значение

Метод `getParts` возвращает немодифицируемый список частей кода предложения.

LearningException

Интерфейс LearningException поддерживает интерфейс ILearning. Некоторые методы в этом интерфейсе требуют реализаций для создания LearningException, представляющей собой простой подкласс java.lang.Exception. Настоятельно рекомендуется для нужд отладки строить LearningException с корневой исключительной ситуацией, если такая исключительная ситуация существует.

IScoreOverride

Интерфейс IScoreOverride поддерживает интерфейс ITreatment. Этот интерфейс позволяет читать данные, определенные в переопределении оценки или в таблице предложений по умолчанию.

getOfferCode

Метод getOfferCode возвращает значение столбцов кода предложения в таблице переопределения оценок для этого участника аудитории.

```
getOfferCode()
```

Возвращаемое значение

Метод getOfferCode возвращает объект IOfferCode, определяющий значение столбцов кода предложения в таблице переопределения оценок.

getParameterNames

Метод getParameterNames возвращает список параметров.

```
getParameterNames()
```

Возвращаемое значение

Метод getParameterNames возвращает набор, определяющий список параметров.

Метод IScoreOverride содержит следующие параметры. Если не указано иное, эти параметры - те же, что и для таблицы переопределения оценок.

- ADJ_EXPLORE_SCORE_COLUMN
- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT - Для переопределения оценки количества (при вычислении веса предложения)
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN - Для создания логического выражения, определяющего приемлемость предложения
- PREDICATE_SCORE - Для создания выражения, дающего числовую оценку
- SCORE_COLUMN
- ZONE_COLUMN

Можно также указать любой столбец, который вы добавляете к таблице переопределению оценок или к таблице предложений по умолчанию, используя то же самое имя для столбца.

getValue

Метод `getValue` возвращает значение столбца зоны в таблице переопределения оценок для этого участника аудитории.

`getValue(String parameterName)`

- **parameterName** - строка, определяющая имя параметра, для которого вы хотите получить значение.

Возвращаемое значение

Метод `getValue` возвращает объект, определяющий значение требуемого параметра.

ISelectionMethod

Интерфейс `ISelection` указывает метод, используемый для представления рекомендованного списка. Значение по умолчанию для объекта Treatment - EXTERNAL_LEARNING, поэтому задавать это значение не требуется. Это значение постоянно хранится в таблице подробной хронологии контактов для создания отчетов.

Если требуется сохранить данные для анализа позднее, этот интерфейс можно расширить за пределы существующих констант. Например, можно создать два разных модуля обучения и реализовать их на разных группах серверов. Интерфейс `ISelection` можно расширить, добавив группы серверов SERVER_GROUP_1 и SERVER_GROUP_2. Затем можно сравнить результаты этих двух модулей обучения.

Интерфейс ITreatment

Интерфейс `ITreatment` поддерживает интерфейс `ILearning` как интерфейс для информации о процедуре. Процедура представляет предложение, назначенное определенной ячейке, как задано в среде разработки. В этом интерфейсе можно получить информацию о ячейках и предложениях, а также назначенную им маркетинговую оценку.

getCellCode

Метод `getCellCode` возвращает код ячейки, как определено в Campaign. Это ячейка, назначенная интеллектуальному сегменту, который связан с этим предложением.

`getCellCode()`

Возвращаемое значение

Метод `getCellCode` возвращает строку, определяющую код ячейки.

getCellId

Метод `getCellId` возвращает внутренний ID ячейки, как он определен в Campaign. Это ячейка, назначенная интеллектуальному сегменту, который связан с этим предложением.

`getOfferName()`

Возвращаемое значение

Метод getCellId возвращает значение типа long, определяющее ID ячейки.

getCellName

Метод getCellName возвращает имя ячейки, как оно определено в Campaign. Это ячейка, назначенная интеллектуальному сегменту, который связан с этим предложением.

```
getCellName()
```

Возвращаемое значение

Метод getCellName возвращает строку, определяющую имя ячейки.

getLearningScore

Метод getLearningScore возвращает оценку для данной процедуры.

```
getLearningScore()
```

Очередность следующая.

1. Возвратить значение переопределения, если оно найдено в карте значений переопределения по ключу IScoreoveride.PREDICATE_SCORE_COLUMN
2. Возвратить оценку предиката, если это значение не пусто
3. Возвратить оценку маркетолога, если она найдена в карте значений переопределения по ключу IScoreoveride.SCORE
4. Возвратить оценку маркетолога

Возвращаемое значение

Метод getLearningScore возвращает целое число, определяющее оценку, которая получена алгоритмом обучения.

getMarketerScore

Метод getMarketerScore возвращает оценку маркетолога, определенную ползунком на вкладке стратегии взаимодействия для предложения.

```
getMarketerScore()
```

Для получения оценки маркетолога, определенной дополнительными опциями на вкладке стратегии взаимодействия, используйте getPredicateScore.

Для получения оценки маркетолога, фактически используемой процедурой, используйте getLearningScore.

Возвращаемое значение

Метод getMarketerScore возвращает целое число, определяющее оценку маркетолога.

getOffer

Метод getOffer возвращает предложение для процедуры.

```
getOffer()
```

Возвращаемое значение

Метод getOffer возвращает объект IOffer, определяющий предложение для данной процедуры.

getOverrideValues

Метод getOverrideValues возвращает переопределения, определенные в предложениях по умолчанию или в таблице переопределения оценок.

```
getOverrideValues()
```

Возвращаемое значение

Метод getOverrideValues возвращает объект IScoreOverride.

getPredicate

Метод getPredicate возвращает предикат, заданный столбцом предиката таблицы предложений по умолчанию, таблицы переопределения оценок или дополнительными опциями правил процедур.

```
getPredicate()
```

Возвращаемое значение

Метод getPredicate возвращает строку, которая определяет предикат, заданный столбцом предиката таблицы предложений по умолчанию, таблицы переопределения оценок или дополнительными опциями правил процедур.

getPredicateScore

Метод getPredicateScore возвращает оценку, заданную столбцом предиката таблицы предложений по умолчанию, таблицы переопределения оценок или дополнительными опциями правил процедур.

```
getPredicateScore()
```

Возвращаемое значение

Метод getPredicateScore возвращает число двойной точности, которое определяет оценку, заданную столбцом предиката таблицы предложений по умолчанию, таблицы переопределения оценок или дополнительными опциями правил процедур.

getScore

Метод getScore возвращает маркетинговую оценку, определенную либо стратегией взаимодействия в Campaign, либо таблицей переопределения оценок.

```
getScore()
```

Метод getScore возвращает одно из следующих значений:

- Маркетинговую оценку предложения, как определено на вкладке стратегии взаимодействия в Campaign, если для свойства enableScoreOverrideLookup задано значение false.
- Оценку предложения, как определено в таблице scoreOverrideTable, если для свойства enableScoreOverrideLookup задано значение true.

Возвращаемое значение

Метод getScore возвращает целое число, представляющее оценку предложения.

getTreatmentCode

Метод `getTreatmentCode` возвращает код процедуры.

```
getTreatmentCode()
```

Возвращаемое значение

Метод `getTreatmentCode` возвращает строку, определяющую код процедуры.

setActualValueUsed

Метод `setActualValueUsed` служит для определения, какие значения используются на различных этапах выполнения алгоритма обучения.

```
setActualValueUsed(string имя_параметра, object значение)
```

Например, если вы используете этот метод для записи в таблицы хронологии контактов и ответов и изменения существующих отчетов примеров, можно включать в отчеты данные от вашего алгоритма обучения.

- **parmName** - строка, определяющая имя параметра, который вы задаете.
- **value** - объект, определяющий значение параметра, которое вы задаете.

Возвращаемое значение

Нет.

Пример API обучения

В этом разделе содержится пример реализации ILearningInterface. Обратите внимание на то, что данная реализация - всего лишь пример и не предназначена для работы в производственной среде.

В этом примере отслеживается число принятий и контактов, а отношение числа принятий к числу контактов для конкретного предложения используется в качестве меры вероятности принятия этого предложения. Не представленные предложения имеют более высокий приоритет для рекомендации. Предложения с хотя бы одним контактом располагаются в порядке убывания вероятности их принятия.

В этом примере все численные значения хранятся в памяти. Это не реалистичный сценарий, поскольку сервер среды выполнения столкнется с нехваткой памяти. В настоящем производственном сценарии счетчики следует хранить в базе данных.

```
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;

/**
 * Это пример реализации оптимизатора обучения.
 * Интерфейс ILearning находится в библиотеке interact.jar.
```

```
 *
 * Чтобы фактически использовать эту реализацию, выберите ExternalLearning в качестве optimizationType на узле offerServing
 * прикладной программы Interact в конфигурации платформы. На узле offerserving есть также
 * категория External Learning config, в которой необходимо задать имя класса:
 * com.unicacorp.interact.samples.learning.v2.SampleLearning. Учтите однако, что эта реализация - только пример
 * и не предназначена для работы в производственной среде.
 *
 *
 * В этом примере отслеживается число принятий и контактов и рассматривается отношение числа принятий к числу контактов
 * для конкретного предложения в качестве меры вероятности принятия этого предложения.
 *
 *
 * Не представленные предложения получают более высокий приоритет для рекомендации.
 * Предложения с хотя бы одним контактом будут располагаться в порядке убывания вероятности их принятия.
 *
 * Примечание: все счетчики хранятся в памяти. Это не очень реалистичный сценарий, так как он рано или поздно приведет к нехватке
 * памяти. В настоящем производственном сценарии счетчики следует хранить в базе данных.
 *
 */

public class SampleLearning implements ILearning
{

    // Отображение ID предложений на число контактов для ID предложения
    private Map<Long,Integer> _offerToContactCount = new HashMap<Long, Integer>();

    // Отображение ID предложений на число контактов для ID предложения
    private Map<Long,Integer> _offerToAcceptCount = new HashMap<Long, Integer>();


    /* (не-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
    {
        // Если требуются удаленные соединения, это удобное место для их инициализации, поскольку данный
        // метод вызывается один раз при запуске интерактивной веб-программы среды выполнения.
        // В этом примере нет удаленных соединений и печати сообщений для нужд отладки, для которой
        // вызывается этот метод
        System.out.println("Вызов инициализации для SampleLearning");
    }

    /* (не-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
    {
        // Если IC внедрен, будет вызван этот метод повторной инициализации, чтобы обеспечить реализацию
        // для обновления измененных параметров конфигурации
        System.out.println("Вызов повторной инициализации для SampleLearning");
    }


    /* (не-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("Вызов logEvent для SampleLearning");


        if(context.getLearningContext()==ILearningContext.LOG_AS_CONTACT)
        {
            System.out.println("добавление контакта");

            // Отслеживаем все контакты в памяти
            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("добавляем принятие");
        // Отслеживаем все счетчики принятия в памяти, добавляя на карту
        synchronized(_offerToAcceptCount)
        {
           Integer count = _offerToAcceptCount.get(offer.getOfferId());
           if(count == null)
               count = new Integer(1);
           else
               count++;
           _offerToAcceptCount.put(offer.getOfferId(), ++count);
        }
        }

}

/* (не-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("Вызов optimizeRecommendList для SampleLearning");

    // Сортировать процедуры-кандидаты, вызывая сортировщик, определенный в этом классе, и возвращать отсортированный список
    Collections.sort(recList,new MyOfferSorter());

    // теперь просто возвращаем то, что было запрошено через переменную "numberRequested"
    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;
}

/* (не-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
{
    // Если существуют удаленные соединения, сейчас - удачный момент корректно
    // выполнить отсоединение, так как этот метод вызывается при закрытии веб-программы среды выполнения Interact.
    // Для этого примера не требуется ничего делать,
    // кроме вывода отладочного сообщения.
    System.out.println("Вызов закрытия для SampleLearning");

}
// Сортировка по:
// 1. предложения с нулевым числом контактов - для связей порядок основан на исходных данных ввода
// 2. вероятность принятия предложений в убывающем порядке - для сохранения связей порядок основан на исходных данных ввода

public class MyOfferSorter implements Comparator<ITreatment>
{
    private static final long serialVersionUID = 1L;

    /* (не-Javadoc)
     * @see java.lang.Comparable#compareTo(java.lang.Object)
     */
    public int compare(ITreatment treatment1, ITreatment treatment2)
    {

        // получаем число контактов для обеих процедур
        Integer contactCount1 = _offerToContactCount.get(treatment1.getOffer().getOfferId());
        Integer contactCount2 = _offerToContactCount.get(treatment2.getOffer().getOfferId());

        // если для процедуры не было контактов, он имеет приоритет
        if(contactCount1 == null || contactCount1 == 0)
```

```
                return -1;

            if(contactCount2 == null || contactCount2 == 0)
                return 1;

            // получить число принятий
            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;

            // в порядке убывания
            return (int) (acceptProbability2 - acceptProbability1);

        }
    }

}
```

Глава 12. IBM Interact WSDL

Установка Interact включает два файла XML WDSL (Web Services Description Language), содержащие описание доступных веб-служб и способов доступа к ним. Эти файлы можно просмотреть в вашем домашнем каталоге Interact, ниже приведен пример.

После установки Interact файлы WSDL Interact можно найти в следующем каталоге:

- *<домашний_каталог_Interact>*/conf/InteractService.wsdl
- *<домашний_каталог_Interact>*/conf/InteractAdminService.wsdl

В каждом новом выпуске или пакете исправлений возможны изменения в WDSL Interact. Подробную информацию смотрите в *Замечания по выпуску Interact* или файлах readme к выпуску.

Для информации здесь показана копия файла InteractService.wsdl. Чтобы убедиться, что используется самая свежая информация, посмотрите файлы WDSL, установленные вместе с Interact.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://soap.api.interact.unicacorp.com" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" 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>
```

Глава 13. Свойства конфигурации среды выполнения Interact

В этом разделе описываются все свойства конфигурации для среды выполнения Interact.

Interact | general

Эти свойства конфигурации задают общие параметры для среды выполнения, включая уровень записи в журнал по умолчанию и параметры локали.

log4jConfig

Описание

Расположение файла, содержащего свойства log4j. Это должен быть путь относительно переменной среды `INTERACT_HOME`. `INTERACT_HOME` - это расположение каталога установки Interact.

Значение по умолчанию

`./conf/interact_log4j.properties`

asmUserForDefaultLocale

Описание

Свойство `asmUserForDefaultLocale` задает пользователя IBM Marketing Software, от имени которого Interact получает параметры локали.

Параметры локали определяют то, какой язык будет появляться во время разработки и на каком языке будут представлены сообщения рекомендации от API Interact. Если параметр локали не соответствует параметрам операционной системы на вашем компьютере, Interact все равно будет функционировать, однако интерфейс во время разработки и сообщения рекомендации могут оказаться на другом языке.

Значение по умолчанию

`asm_admin`

Interact | general | learningTablesDataSource

Эти свойства конфигурации задают параметры источника данных для встроенных таблиц обучения. Вы должны задать этот источник данных, если используете встроенное обучение Interact.

Если вы создадите свою реализацию обучения с использованием API обучения, вы сможете сконфигурировать пользовательскую реализацию обучения, чтобы прочитать эти значения с использованием интерфейса `ILearningConfig`.

jndiName

Описание

Используйте свойство jndiName, чтобы задать источник данных Java Naming and Directory Interface (JNDI), заданный на сервере прикладных программ (Websphere или WebLogic) для таблиц обучения, доступ к которым получают серверы среды выполнения Interact.

Таблицы обучения создаются файлом ddl aci_lrntab, к в их число входят следующие таблицы (помимо прочих): UACI_AttributeValue и UACI_OfferStats.

Значение по умолчанию

Значения по умолчанию нет.

type

Описание

Тип базы данных для источника данных, используемого таблицами обучения, связанными с серверами среды выполнения Interact.

Таблицы обучения создаются файлом ddl aci_lrntab, к в их число входят следующие таблицы (помимо прочих): UACI_AttributeValue и UACI_OfferStats.

Значение по умолчанию

SQLServer

Допустимые значения

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Описание

Свойство ConnectionRetryPeriod задает время (в секундах), в течение которого Interact автоматически повторяет требование установления соединения с базой данных при ошибке для таблиц обучения. Interact автоматически попытается восстановить соединение с базой данных в течение этого времени, прежде чем сообщить об ошибке или сбое базы данных. Если задано значение, равное 0, Interact будет повторять попытки до бесконечности; если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Таблицы обучения создаются файлом ddl aci_lrntab, к в их число входят следующие таблицы (помимо прочих): UACI_AttributeValue и UACI_OfferStats.

Значение по умолчанию

-1

connectionRetryDelay

Описание

Свойство ConnectionRetryDelay задает время (в секундах), в течение которого Interact попытается повторно соединиться с базой данных после сбоя для таблиц обучения. Если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Таблицы обучения создаются файлом ddl aci_lrntab, к в их число входят следующие таблицы (помимо прочих): UACI_AttributeValue и UACI_OfferStats.

Значение по умолчанию

-1

ID пользователя

Описание

Имя схемы, содержащей таблицы для встроенного модуля обучения. Interact добавит значение этого свойства перед именем каждой таблицы, например, `UACI_IntChannel` превратится в `schema.UACI_IntChannel`.

Задавать схему вы не должны. Если вы не зададите схему, Interact сочтет, что владелец таблиц совпадает со схемой. Вы должны задать это значение, чтобы избежать неоднозначности.

Значение по умолчанию

Значения по умолчанию нет.

Interact | general | prodUserDataSource

Эти свойства конфигурации задают параметры источника данных для производственных таблиц профиля. Вы должны задать этот источник данных. Эта источник данных, на который ссылается среда выполнения при обработке интерактивных потоковых диаграмм после внедрения.

jndiName

Описание

Используйте свойство `jndiName`, чтобы задать источник данных Java Naming and Directory Interface (JNDI), заданный на сервере прикладных программ (Websphere или WebLogic) для таблиц обучения, доступ к которым получают серверы среды выполнения Interact.

Значение по умолчанию

Значения по умолчанию нет.

type

Описание

Тип базы данных для таблиц покупателей, доступ к которым осуществляется серверами среды выполнения Interact.

Значение по умолчанию

SQLServer

Допустимые значения

SQLServer | DB2 | ORACLE

aliasPrefix

Описание

Свойство `AliasPrefix` указывает, каким образом Interact формирует алиас, который компонент Interact создает автоматически при использовании таблицы измерения и записи в новую таблицу в таблицах покупателей, доступ к которым осуществляется серверами выполнения Interact.

Учтите, что у каждой базы данных есть максимальная длина идентификатора; чтобы быть уверенным в том, что заданное вами значение

не превысит максимальной длины идентификатора для базы данных, смотрите документацию по вашей базе данных.

Значение по умолчанию

A

connectionRetryPeriod

Описание

Свойство `ConnectionRetryPeriod` задает время (в секундах), в течение которого Interact автоматически повторяет требование установления соединения с базой данных при ошибке для таблиц среды выполнения покупателей. Interact автоматически попытается восстановить соединение с базой данных в течение этого времени, прежде чем сообщить об ошибке или сбое базы данных. Если задано значение, равное 0, Interact будет повторять попытки до бесконечности; если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Значение по умолчанию

-1

connectionRetryDelay

Описание

Свойство `ConnectionRetryDelay` задает время (в секундах), в течение которого Interact попытается повторно соединиться с базой данных после сбоя для таблиц среды выполнения покупателей в Interact. Если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Значение по умолчанию

-1

ID пользователя

Описание

Имя схемы, содержащей таблицы данных профиля. Interact добавит значение этого свойства перед именем каждой таблицы, например, `UACI_IntChannel` превратится в `schema.UACI_IntChannel`.

Задавать схему вы не должны. Если вы не зададите схему, Interact сочтет, что владелец таблиц совпадает со схемой. Вы должны задать это значение, чтобы избежать неоднозначности.

При использовании базы данных DB2 имя схемы должно быть в верхнем регистре.

Значение по умолчанию

Значения по умолчанию нет.

Interact | general | systemTablesDataSource

Эти свойства конфигурации задают параметры источника данных для системных таблиц для среды выполнения. Вы должны задать этот источник данных.

jndiName

Описание

Используйте свойство jndiName, чтобы задать источник данных Java Naming and Directory Interface (JNDI), заданный на сервере прикладных программ (Websphere или WebLogic) для таблиц среды выполнения.

База данных среды выполнения - это база данных, заполненная сценариями dll aci_runtime и aci_populate_runtime, и, например, она соде ржит следующие таблицы (помимо прочих): UACI_CHOfferAttrib и UACI_DefaultedStat.

Значение по умолчанию

Значения по умолчанию нет.

type

Описание

Тип базы данных для системных таблиц среды выполнения.

База данных среды выполнения - это база данных, заполненная сценариями dll aci_runtime и aci_populate_runtime, и, например, она соде ржит следующие таблицы (помимо прочих): UACI_CHOfferAttrib и UACI_DefaultedStat.

Значение по умолчанию

SQLServer

Допустимые значения

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Описание

Свойство ConnectionRetryPeriod задает время (в секундах), в течение которого Interact автоматически повторяет требование установления соединения с базой данных при ошибке для системных таблиц среды выполнения. Interact автоматически попытается восстановить соединение с базой данных в течение этого времени, прежде чем сообщить об ошибке или сбое базы данных. Если задано значение, равное 0, Interact будет повторять попытки до бесконечности; если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

База данных среды выполнения - это база данных, заполненная сценариями dll aci_runtime и aci_populate_runtime, и, например, она соде ржит следующие таблицы (помимо прочих): UACI_CHOfferAttrib и UACI_DefaultedStat.

Значение по умолчанию

-1

connectionRetryDelay

Описание

Свойство ConnectionRetryDelay задает время (в секундах), в течение которого Interact попытается повторно соединиться с базой данных после сбоя для системных таблиц среды выполнения Interact. Если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

База данных среды выполнения - это база данных, заполненная сценариями dll `aci_runtime` и `aci_populate_runtime`, и, например, она соде ржит следующие таблицы (помимо прочих): `UACI_CHOfferAttrib` и `UACI_DefaultedStat`.

Значение по умолчанию

-1

ID пользователя

Описание

Имя схемы, содержащей таблицы для среды выполнения. Interact добавит значение этого свойства перед именем каждой таблицы, например, `UACI_IntChannel` превратится в `schema.UACI_IntChannel`.

Задавать схему вы не должны. Если вы не зададите схему, Interact сочтет, что владелец таблиц совпадает со схемой. Вы должны задать это значение, чтобы избежать неоднозначности.

Значение по умолчанию

Значения по умолчанию нет.

Interact | general | systemTablesDataSource | loaderProperties

Эти свойства конфигурации задают параметры утилиты загрузки базы данных для системных таблиц для среды выполнения. Вы обязательно должны задать эти свойства, если вы используете только утилиту загрузки базы данных.

databaseName

Описание

Имя базы данных, с которой соединяется загрузчик базы данных.

Значение по умолчанию

Значения по умолчанию нет.

LoaderCommandForAppend

Описание

Параметр `LoaderCommandForAppend` задает команду, генерируемую для вызова утилиты загрузки базы данных для присоединения записей к промежуточным таблицам базы данных для хронологии контактов и ответов в Interact. Вы должны задать этот параметр, чтобы включить утилиту загрузки базы данных для данных хронологии контактов и ответов.

Этот параметр задает любой полный путь либо выполняемого файла утилиты загрузки базы данных, либо сценарий, запускающий утилиту загрузки базы данных. Использование сценария позволяет выполнить дополнительную настройку перед вызовом утилиты загрузки.

Для успешного запуска большинства утилит загрузки баз данных требуется несколько аргументов. Они могут включать в себя файл данных и контрольный файл для загрузки из них данных, а также базу данных и таблицу, в которые нужно загрузить данные. Маркеры при выполнении команды заменяются конкретными элементами.

Правильный синтаксис, который нужно использовать при вызове утилиты загрузки вашей базы данных, смотрите в документации по утилите загрузки вашей базы данных.

По умолчанию, этот параметр не задан.

Маркеры, доступные для `LoaderCommandForAppend`, описаны в следующей таблице.

Маркер	**Описание**
<CONTROLFILE>	Этот маркер будет заменен на полный путь и имя файла временного контрольного файла, который генерирует Interact в соответствии с шаблоном, заданным в параметре `LoaderControlFileTemplate`.
<DATABASE>	Этот маркер заменяется именем источника данных, в который Interact загружает данные. Это то же самое имя источника данных, которое используется в имени категории для этого источника данных.
<DATAFILE>	Этот маркер будет заменен на полный путь и имя файла временного файла данных, созданного компонентом Interact во время процесса загрузки. Этот файл расположен во временном каталоге Interact, `UNICA_ACTMPDIR`.
<DBCOLUMNNUMBER>	Этот маркер заменяется на ординал столбца в базе данных.
<FIELDLENGTH>	Этот маркер заменяется длиной поля, загружаемого в базу данных.
<FIELDNAME>	Этот маркер заменяется именем поля, загружаемого в базу данных.
<FIELDNUMBER>	Этот маркер заменяется числом полей, загружаемых в базу данных.
<FIELDTYPE>	Этот маркер заменяется литералом "CHAR( )". Длина поля задана в скобках: (). Если окажется, что ваша база данных не понимает тип поля CHAR, вы можете вручную задать соответствующий текст для типа поля и использовать маркер <FIELDLENGTH>. Например, для SQLSVR и SQL2000 вы бы использовали "SQLCHAR(<FIELDLENGTH>)"
<NATIVETYPE>	Этот маркер заменяется типом базы данных, в которую загружается это поле.
<NUMFIELDS>	Этот маркер заменяется числом полей в таблице.
<PASSWORD>	Этот маркер заменяется на пароль базы данных из текущего соединения потоковой диаграммы с источником данных.

Маркер	Описание
<TABLENAME>	Этот маркер заменяется именем таблицы базы данных, в которую Interact загружает данные.
<USER>	Этот маркер заменяется на пользователя базы данных из текущего соединения потоковой диаграммы с источником данных.

Значение по умолчанию

Значения по умолчанию нет.

LoaderControlFileTemplateForAppend

Описание

Свойство `LoaderControlFileTemplateForAppend` задает полный путь и имя файла шаблона контрольного файла, ранее сконфигурированного в Interact. Если этот параметр задан, Interact динамически построит временный контрольный файл на основе шаблона, который вы здесь укажете. Путь и имя этого временного контрольного файла доступны для маркера `<CONTROLFILE>`, который используется для свойства `LoaderCommandForAppend`.

Прежде чем использовать Interact в режиме утилиты загрузки базы данных, нужно сконфигурировать шаблон контрольного файла, заданный этим параметром. Шаблон контрольного файла поддерживает перечисленные ниже маркеры, которые при создании временного контрольного файла компонентом Interact динамически заменяются конкретными элементами.

Правильный синтаксис, необходимый для вашего контрольного файла, смотрите в документации по утилите загрузки для вашей базы данных. Маркеры, доступные для шаблона контрольного файла - это те же самые маркеры, которые используются для свойства `LoaderControlFileTemplate`.

По умолчанию, этот параметр не задан.

Значение по умолчанию

Значения по умолчанию нет.

LoaderDelimiterForAppend

Описание

Свойство `LoaderDelimiterForAppend` задает, будет ли временный файл данных Interact плоским файлом с фиксированной шириной полей или с разделителями и, если это будет файл с разделителями, какие символы или наборы символов будут использоваться в качестве разделителей.

Если значение не задано, Interact создаст временный файл данных как плоский файл с фиксированной шириной полей.

Если вы укажете значение, оно будет использоваться при вызове загрузчика для заполнения таблицы, про которую известно, что она не является пустой. Interact создает временный файл данных как плоский файл с разделителями, используя значение данного свойства в качестве разделителя.

По умолчанию, это свойство не задано.

Значение по умолчанию

Допустимые значения

Символы, которые можно заключить в двойные кавычки (если это нужно).

LoaderDelimiterAtEndForAppend

Описание

Некоторым внешним утилитам загрузки требуется, чтобы файл данных был файлом с разделителями и чтобы каждая строка заканчивалась разделителем. Чтобы учесть это требование, задайте для LoaderDelimiterAtEndForAppend значение TRUE, чтобы при вызове загрузчика для заполнения таблицы, которая, как известно, не является пустой, компонент Interact использовал разделители в конце каждой строки.

Значение по умолчанию

FALSE

Допустимые значения

TRUE | FALSE

LoaderUseLocaleDP

Описание

Свойство LoaderUseLocaleDP указывает, будет ли в качестве десятичного разделителя использоваться символ, связанный с локалью, когда Interact записывает числовые значения в файлы, которые должна загрузить утилита загрузки базы данных.

Задайте значение FALSE, чтобы указать, что в качестве десятичного разделителя используется точка (.).

Задайте значение TRUE, чтобы указать, что в качестве десятичного разделителя используется символ, соответствующий локали.

Значение по умолчанию

FALSE

Допустимые значения

TRUE | FALSE

Interact | general | testRunDataSource

Эти свойства конфигурации задают параметры источника данных для таблиц тест-запуска для среды разработки Interact. Вы должны задать этот источник данных хотя бы для одной из сред выполнения. Это таблицы, используемые при выполнении тест-запуска интерактивной потоковой диаграммы.

jndiName

Описание

Используйте свойство jndiName, чтобы задать источник данных Java Naming and Directory Interface (JNDI), заданный на сервере прикладных программ (Websphere или WebLogic) для таблиц покупателей, доступ к которым осуществляется средой разработки при выполнении тест-запусков интерактивных потоковых диаграмм.

Значение по умолчанию

Значения по умолчанию нет.

type

Описание

Тип базы данных для таблиц покупателей, доступ к которым осуществляется средой разработки при выполнении тест-запусков интерактивных потоковых диаграмм.

Значение по умолчанию

SQLServer

Допустимые значения

SQLServer | DB2 | ORACLE

aliasPrefix

Описание

Свойство `AliasPrefix` указывает, каким образом Interact формирует алиас, который компонент Interact создает автоматически при использовании таблицы измерения и записи в новую таблицу для таблиц покупателей, доступ к которым осуществляется средой разработки при выполнении тест-запусков интерактивных потоковых диаграмм.

Учтите, что у каждой базы данных есть максимальная длина идентификатора; чтобы быть уверенным в том, что заданное вами значение не превысит максимальной длины идентификатора для базы данных, смотрите документацию по вашей базе данных.

Значение по умолчанию

A

connectionRetryPeriod

Описание

Свойство `ConnectionRetryPeriod` задает время (в секундах), в течение которого Interact автоматически повторяет требование установления соединения с базой данных при ошибке для таблиц тест-запуска. Interact автоматически попытается восстановить соединение с базой данных в течение этого времени, прежде чем сообщить об ошибке или сбое базы данных. Если задано значение, равное 0, Interact будет повторять попытки до бесконечности; если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Значение по умолчанию

-1

connectionRetryDelay

Описание

Свойство `ConnectionRetryDelay` задает время (в секундах), в течение которого Interact попытается повторно соединиться с базой данных после сбоя для таблиц тест-запуска. Если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Значение по умолчанию

-1

ID пользователя

Описание

Имя схемы, содержащей таблицы для тест-запусков интерактивной потоковой диаграммы. Interact добавит значение этого свойства перед именем каждой таблицы, например, `UACI_IntChannel` превратится в `schema.UACI_IntChannel`.

Задавать схему вы не должны. Если вы не зададите схему, Interact сочтет, что владелец таблиц совпадает со схемой. Вы должны задать это значение, чтобы избежать неоднозначности.

Значение по умолчанию

Значения по умолчанию нет.

Interact | general | contactAndResponseHistoryDataSource

Эти свойства конфигурации задают параметры соединения для источника данных хронологии контактов и ответов, необходимого для отслеживания ответов между разными сеансами Interact. Эти параметры не связаны с модулем хронологии контактов и ответов.

jndiName

Описание

Используйте свойство `jndiName`, чтобы задать источник данных Java Naming and Directory Interface (JNDI), заданный на сервере прикладных программ (WebSphere или WebLogic) для источника данных хронологии контактов и ответов, необходимого для отслеживания ответов между разными сеансами Interact.

Значение по умолчанию

type

Описание

Тип базы данных для источника данных, используемого источником данных хронологии контактов и ответов, необходимого для отслеживания ответов между разными сеансами Interact.

Значение по умолчанию

SQLServer

Допустимые значения

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Описание

Свойство `ConnectionRetryPeriod` задает время (в секундах), в течение которого Interact автоматически повторяет требование установления соединения с базой данных при ошибке для отслежитвания ответов между сеансами Interact. Interact автоматически попытается восстановить соединение с базой данных в течение этого времени, прежде чем сообщить об ошибке или сбое базы данных. Если задано значение, равное 0, Interact будет повторять попытки до бесконечности; если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Значение по умолчанию

-1

connectionRetryDelay

Описание

Свойство `ConnectionRetryDelay` задает время (в секундах), в течение которого Interact попытается повторно соединиться с базой данных после сбоя для отслеживания ответов между разными сеансами Interact. Если задано значение, равное -1, никакие повторные попытки предприниматься не будут.

Значение по умолчанию

-1

ID пользователя

Описание

Имя схемы, содержащей таблицы для отслеживания ответов между разными сеансами Interact. Interact добавит значение этого свойства перед именем каждой таблицы, например, `UACI_IntChannel` превратится в `schema.UACI_IntChannel`.

Задавать схему вы не должны. Если вы не зададите схему, Interact сочтет, что владелец таблиц совпадает со схемой. Вы должны задать это значение, чтобы избежать неоднозначности.

Значение по умолчанию

Значения по умолчанию нет.

Interact | general | idsByType

Эти свойства конфигурации задают параметры для числовых ID, используемых модулем хронологии контактов и ответов.

initialValue

Описание

Первоначальное значение ID, используемое при генерировании ID с использованием таблицы UACI_IDsByType.

Значение по умолчанию

1

Допустимые значения

Любое значение больше 0.

retries

Описание

Число попыток перед генерированием исключения при генерировании ID с использованием таблицы UACI_IDsByType.

Значение по умолчанию

20

Допустимые значения

Любое целое число, больше 0.

Interact | flowchart

В этом разделе заданы параметры конфигурации для интерактивных потоковых диаграмм.

defaultDateFormat

Описание

Формат дат по умолчанию, используемый компонентом Interact для преобразования даты в строку и строки в дату.

Значение по умолчанию

MM/dd/yy

idleFlowchartThreadTimeoutInMinutes

Описание

Время в минутах, в течение которого Interact позволит интерактивной потоковой диаграмме оставаться в бездействии, прежде чем высвободить поток.

Значение по умолчанию

5

idleProcessBoxThreadTimeoutInMinutes

Описание

Время в минутах, в течение которого Interact позволит процессу интерактивной потоковой диаграммы оставаться в бездействии, прежде чем высвободить поток.

Значение по умолчанию

5

maxSizeOfFlowchartEngineInboundQueue

Описание

Максимальное число требований запуска потоковой диаграммы, которые Interact хранит в очереди. При достижении этого числа требований Interact перестанет принимать требования.

Значение по умолчанию

1000

maxNumberOfFlowchartThreads

Описание

Максимальное число потоков, выделенных для требований интерактивных потоковых диаграмм.

Значение по умолчанию

25

maxNumberOfProcessBoxThreads

Описание

Максимальное число потоков, выделенных для процессов интерактивных потоковых диаграмм.

Значение по умолчанию

50

maxNumberOfProcessBoxThreadsPerFlowchart

Описание

Максимальное число потоков, выделенных для процессов интерактивных потоковых диаграмм для каждого экземпляра потоковой диаграммы.

Значение по умолчанию

3

minNumberOfFlowchartThreads

Описание

Минимальное число потоков, выделенных для требований интерактивных потоковых диаграмм.

Значение по умолчанию

10

minNumberOfProcessBoxThreads

Описание

Минимальное число потоков, выделенных для процессов интерактивных потоковых диаграмм.

Значение по умолчанию

20

sessionVarPrefix

Описание

Префикс для переменных сеанса.

Значение по умолчанию

SessionVar

Interact | flowchart | ExternalCallouts | [ExternalCalloutName]

В этом разделе заданы параметры класса для пользовательских внешних вызовов, записанных с использованием API внешних вызовов.

class

Описание

Имя Java-класса, представленного данным внешним вызовом.

Это Java-класс, доступ к которому можно получить при помощи макроса IBM `EXTERNALCALLOUT`.

Значение по умолчанию

Значения по умолчанию нет.

classpath

Описание

Путь классов для Java-класса, представленного данным внешним вызовом. Путь классов должен ссылаться на файлы jar на сервере среды выполнения. Если вы используете группу серверов и все серверы выполнения используют один и тот же экземпляр Marketing Platform, на каждом сервере должна быть копия файла jar в одном и том же месте. Путь классов должен состоять из абсолютных расположений файлов jar, разделенных разделителем путей в соответствии с операционной системой на сервере среды выполнения, например, точка с запятой (;) в системах Windows и двоеточие (:) - в системах UNIX. Каталоги, содержащие файлы классов, не принимаются. Например, в системе Unix: `/path1/file1.jar:/path2/file2.jar`.

Этот путь классов должен содержать менее 1024 символов. Вы можете использовать файл манифеста в файле .jar, чтобы указать другие файлы .jar, так что в пути классов должен появляться только один файл .jar.

Это Java-класс, доступ к которому можно получить при помощи макроса IBM `EXTERNALCALLOUT`.

Значение по умолчанию

Значения по умолчанию нет.

Interact | flowchart | ExternalCallouts | [ExternalCalloutName] | Parameter Data | [parameterName]

В этом разделе задано значение параметра для пользовательского внешнего вызова, записанного с использованием API внешних вызовов.

value

Описание

Значение любого параметра, который требуется для класса внешнего вызова.

Значение по умолчанию

Значения по умолчанию нет.

Пример

Если для внешнего вызова требуется имя хоста внешнего сервера, создайте категорию параметра `host` и задайте свойство `value` в качестве имени сервера.

Interact | monitoring

Этот набор свойств конфигурации позволяет задать параметры мониторинга JMX. Вы должны сконфигурировать эти свойства, только если вы используете мониторинг JMX. Существуют отдельные свойства мониторинга JMX, которые нужно задать для модуля хронологии контактов и ответов в свойствах конфигурации для среды разработки Interact.

protocol

Описание

Задайте протокол для службы сообщений Interact.

Если вы выберете JMXMP, вы должны включить в путь классов следующие файлы JAR:

`Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar`

Значение по умолчанию

JMXMP

Допустимые значения

JMXMP | RMI

port

Описание

Номер порта для службы сообщений.

Значение по умолчанию

9998

enableSecurity

Описание

Логическое значение, которое включает или выключает безопасность службы сообщений JMXMP для сервера среды выполнения Interact. Если задано значение `true`, нужно будет вводить имя пользователя и пароль для получения доступа к службе JMX среды выполнения Interact. Эти учетные данные пользователя аутентифицируются компонентом Marketing Platform для среды выполнения. Jconsole не разрешает вход в систему с пустым паролем.

Это свойство не оказывает никакого влияния, если протоколом является RMI. Это свойство никак не влияет на JMX для Campaign (во время разработки Interact).

Значение по умолчанию

True

Допустимые значения

True | False

Interact | мониторинг | activitySubscribers

Этот набор свойств конфигурации позволяет корневому узлу иметь параметры, связанные с удаленными подписчиками, которые могут получать периодическое обновление основных данных о производительности в среде выполнения Interact.

heartbeatPeriodInSecs

Описание

Интервал в секундах, с которым каждый экземпляр времени выполнения посылает обновления подписчикам.

Значение по умолчанию

60

Interact | monitoring | activitySubscribers | (целевой объект)

(целевой объект)

Описание

Корневой узел для задания подписчика.

URL

Описание

URL этого подписчика. Эта конечная точка должна быть способна принять сообщения JSON, передаваемые через HTTP.

continuousErrorsForAbort

Описание

Количество последовательных неудачных изменений, при достижении которого экземпляр среды выполнения прекращает посылать изменения этому подписчику.

Значение по умолчанию

5

timeoutInMillis

Описание

Время ожидания в миллисекундах для процесса отправки изменения этому подписчику.

Значение по умолчанию

1000

Допустимые значения

Включен

Описание

Включен подписчик или отключен.

Значение по умолчанию

True

Допустимые значения

True или False

type

Описание

Тип этого склада данных. При выборе этой опции нужно добавить параметр **className** со значением, представляющим собой полное имя этого класса реализации. Если по пути классов среды выполнения Interact нет URI файла JAR, его нужно добавить в **classPath**.

Значение по умолчанию

InteractLog

Допустимые значения

InteractLog, RelationalDB и Custom

jmxInclusionCycles

Описание

Интервал в множителе **heartbeatPeriodInSecs**, с которым подробная статистика JMX отправляется этому подписчику.

Значение по умолчанию

5

Допустимые значения

Interact | profile

Этот набор свойств конфигурации управляет несколькими дополнительными возможностями представления предложений, в частности подавлением предложений и переопределением оценок.

enableScoreOverrideLookup

Описание

Если задано значение True, Interact загружает данные переопределения оценок из scoreOverrideTable при создании сеанса. Если задано значение False, Interact не загружает данные переопределения оценок для маркетинга при создании сеанса.

При значении true надо также сконфигурировать свойство Interact | profile | Audience Levels | (уровень аудитории) | scoreOverrideTable. Свойство scoreOverrideTable необходимо определить только для нужных вам уровней аудитории. Если оставить значение scoreOverrideTable для некоторого уровня аудитории пустым, это отключит переопределение оценок для этого уровня.

Значение по умолчанию

False

Допустимые значения

True | False

enableOfferSuppressionLookup

Описание

Если задано значение True, при создании сеанса Interact загружает данные подавления предложений из offerSuppressionTable. Если задано значение False, при создании сеанса Interact не загружает данные подавления предложений .

При значении true надо также сконфигурировать свойство Interact | profile | Audience Levels | (уровень аудитории) | offerSuppressionTable. Свойство enableOfferSuppressionLookup необходимо определить только для нужных вам уровней аудитории.

Значение по умолчанию

False

Допустимые значения

True | False

enableProfileLookup

Описание

В новой установке Interact это свойство признано устаревшим. В обновленной установке Interact это свойство будет действительным до первого внедрения.

Поведение загрузки для таблицы, используемой в интерактивной потоковой диаграмме, но не отображенной в интерактивный канал. Если задано значение `True`, при создании сеанса Interact загружает данные профиля из `profileTable`.

При значении true надо также сконфигурировать свойство `Interact | profile | Audience Levels | (уровень аудитории) | profileTable`.

Параметр **Загрузить эти данные в память при начале сеанса посетителя** в мастере по отображению таблиц в интерактивном канале переопределяет это свойство конфигурации.

Значение по умолчанию

`False`

Допустимые значения

True | False

defaultOfferUpdatePollPeriod

Описание

Время выжидания системой, прежде чем изменить предложения по умолчанию в кэше из таблицы предложений по умолчанию. Если задано значение -1, при запуске сервера среды выполнения система не изменяет предложения по умолчанию в кэше после загрузки первоначального списка в кэш.

Значение по умолчанию

-1

Interact | profile | Уровни аудитории | [AudienceLevelName]

Этот набор свойств конфигурации позволяет задать имена таблиц, необходимых для дополнительных функций Interact. Вы должны задать имя таблицы, только если вы используете связанную функцию.

Имя новой категории

Описание

Имя вашего уровня аудитории.

scoreOverrideTable

Описание

Имя таблицы, содержащей информацию о переопределении оценок для данного уровня аудитории. Это свойство применимо, если вы задали для свойства `enableScoreOverrideLookup` значение true. Вы должны задать это свойство для уровней аудитории, для которых вы хотите включить таблицу переопределения оценок. Если у вас нет таблицы переопределения оценок для

этого уровня аудитории, вы можете оставить это свойство не заданным, даже если для свойства enableScoreOverrideLookup задано значение true.

Interact ищет эту таблицу в таблицах покупателей, доступ к которым осуществляется серверами среды выполнения Interact, заданными в свойствах prodUserDataSource.

Если вы задали свойство schema для этого источника данных, Interact присоединит это имя таблицы в начало имени схемы, например, schema.UACI_ScoreOverride. Если вы введете полный путь, например, mySchema.UACI_ScoreOverride, Interact не будет присоединять имя схемы.

Значение по умолчанию

UACI_ScoreOverride

offerSuppressionTable

Описание

Имя таблицы, содержащей информацию о подавлении предложений для данного уровня аудитории. Вы должны задать это свойство для уровней аудитории, для которых вы хотите включить таблицу подавления предложений. Если у вас нет таблицы подавления предложений для этого уровня аудитории, вы можете оставить это свойство не заданным. Если для enableOfferSuppressionLookup задано значение true, для этого свойства должна быть задана допустимая таблица.

Interact ищет эту таблицу в таблицах покупателей, доступ к которым осуществляется серверами среды выполнения, заданными в свойствах prodUserDataSource.

Значение по умолчанию

UACI_BlackList

contactHistoryTable

Описание

Имя промежуточной таблицы для данных хронологии контактов для этого уровня аудитории.

Эта таблица хранится в таблицах среды выполнения (systemTablesDataSource).

Если вы задали свойство schema для этого источника данных, Interact присоединит это имя таблицы в начало имени схемы, например, schema.UACI_CHStaging. Если вы введете полный путь, например, mySchema.UACI_CHStaging, Interact не будет присоединять имя схемы.

Если сохранение хронологии контакта отключено, это свойство можно не задавать.

Значение по умолчанию

UACI_CHStaging

chOfferAttribTable

Описание

Имя таблицы атрибутов предложений хронологии контактов для этого уровня аудитории.

Эта таблица хранится в таблицах среды выполнения (systemTablesDataSource).

Если вы задали свойство schema для этого источника данных, Interact присоединит это имя таблицы в начало имени схемы, например, schema.UACI_CHOfferAttrib. Если вы введете полный путь, например, mySchema.UACI_CHOfferAttrib, Interact не будет присоединять имя схемы.

Если сохранение хронологии контакта отключено, это свойство можно не задавать.

Значение по умолчанию

UACI_CHOfferAttrib

responseHistoryTable

Описание

Имя промежуточной таблицы хронологии ответов для этого уровня аудитории.

Эта таблица хранится в таблицах среды выполнения (systemTablesDataSource).

Если вы задали свойство schema для этого источника данных, Interact присоединит это имя таблицы в начало имени схемы, например, schema.UACI_RHStaging. Если вы введете полный путь, например, mySchema.UACI_RHStaging, Interact не будет присоединять имя схемы.

Если сохранение хронологии ответов отключено, это свойство можно не задавать.

Значение по умолчанию

UACI_RHStaging

crossSessionResponseTable

Описание

Имя таблицы для данного уровня аудитории, необходимое для отслеживания ответов разных сеансов в таблицах хронологии контактов и ответов, доступных для функции отслеживания ответов.

Если вы задали свойство schema для этого источника данных, Interact присоединит это имя таблицы в начало имени схемы, например, schema.UACI_XSessResponse. Если вы введете полный путь, например, mySchema.UACI_XSessResponse, Interact не будет присоединять имя схемы.

Если запись в журнал межсеансовых ответов отключена, задавать это свойство не нужно.

Значение по умолчанию

UACI_XSessResponse

userEventLoggingTable

Описание

Это имя таблицы базы данных, используемой для записи в журнал пользовательских операций событий. Пользовательские события на вкладке События сводных страниц Интерактивного канала в интерфейсе Interact.

Заданная здесь таблица базы данных хранит такую информацию, как ID события, сколько раз произошло это событие с момента последней очистки кэша операций событий и так далее.

Если вы задали свойство schema для этого источника данных, Interact присоединит это имя таблицы в начало имени схемы, например, schema.UACI_UserEventActivity. Если вы введете полный путь, например, mySchema.UACI_UserEventActivity, Interact не будет присоединять имя схемы.

Значение по умолчанию

UACI_UserEventActivity

patternStateTable

Описание

Это имя таблицы базы данных, используемой для записи в журнал состояний паттернов событий, например, выполнено или нет условие паттерна, отключен ли паттерн, истек ли срок его действия и т.п.

Если вы задали свойство schema для этого источника данных, Interact присоединит это имя таблицы в начало имени схемы, например, schema.UACI_EventPatternState. Если вы введете полный путь, например, mySchema.UACI_EventPatternState, Interact не будет присоединять имя схемы.

patternStateTable требуется для каждого уровня аудитории, даже если вы не используете паттерны событий. Значение patternStateTable основано на DDL включенного UACI_EventPatternState. Ниже приведен пример, когда у ID аудитории есть два компонента, ComponentNum и 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)
)
```

Значение по умолчанию

UACI_EventPatternState

Interact | profile | Audience Levels | [имя_уровня_аудитории] | Offers by Raw SQL

Этот набор свойств конфигурации позволяет задать имена таблиц, необходимых для дополнительных функций Interact. Вы должны задать имя таблицы, только если вы используете связанную функцию.

enableOffersByRawSQL

Описание

Если задано значение True, Interact включает возможность offersBySQL для этого уровня аудитории, что позволяет сконфигурировать код SQL для выполнения при создании нужного набора предложений-кандидатов в среде выполнения. Если задано значение False, Interact не использует возможность offersBySQL.

Если задать значение true, вы можете сконфигурировать также свойство Interact | profile | Audience Levels | (уровень аудитории) | Offers by Raw SQL | SQL Template для определения одного или нескольких шаблонов SQL.

Значение по умолчанию

False

Допустимые значения

True | False

cacheSize

Описание

Размер кэша, используемого для сохранения результатов запросов OfferBySQL. Заметим, что использование кэша может оказать отрицательное действие, если результаты запросов для большинства сеансов уникальны.

Значение по умолчанию

-1 (кэш не используется)

Допустимые значения

-1 | значение

cacheLifeInMinutes

Описание

Если кэш включен, задает время в минутах, по истечении которого система очистит кэш, чтобы избежать устаревания.

Значение по умолчанию

-1 (не используется)

Допустимые значения

-1 | значение

defaultSQLTemplate

Описание

Имя шаблона SQL, который нужно использовать, если никакое имя не задано в вызовах API.

Значение по умолчанию

Отсутствует

Допустимые значения

Имя шаблона SQL

Имя

Категория конфигурации

Interact | profile | Audience Levels | [имя_уровня_аудитории] | Offers by Raw SQL | (Шаблоны SQL)

Описание

Имя, которое вы хотите присвоить этому шаблону запросов SQL. Введите описательное имя, которое будет значащим, когда вы будете использовать этот шаблон SQL в вызовах API. Обратите внимание на то, что при использовании здесь имени, *идентичного* имени, определенному в поле процессов Interact List для процедуры offerBySQL, будет использоваться SQL из поля процессов, а не введенный здесь SQL.

Значение по умолчанию

Нет

SQL

Категория конфигурации

`Interact | profile | Audience Levels | [имя_уровня_аудитории] | Offers by Raw SQL | (шаблоны SQL)`

Описание

Содержит запрос SQL, который будет вызываться этим шаблоном. Запрос SQL может содержать ссылки на имена переменных, которые являются частью данных сеанса посетителя (профиля). Например, `select * from MyOffers where category = ${preferredCategory}` ссылается на сеанс, содержащий переменную с именем `preferredCategory`.

Необходимо сконфигурировать SQL, чтобы запрашивать конкретные таблицы предложений, созданные при разработке для использования этой возможности. Обратите внимание на то, что хранимые процедуры здесь не поддерживаются.

Значение по умолчанию

Нет

Interact | profile | Audience Levels | [имя_уровня_аудитории] | SQL Template

Эти свойства конфигурации позволяют определить один или несколько шаблонов запросов SQL, используемых возможностью offersBySQL для Interact.

Имя

Описание

Имя, которое вы хотите присвоить этому шаблону запросов SQL. Введите описательное имя, которое будет значащим, когда вы будете использовать этот шаблон SQL в вызовах API. Обратите внимание на то, что при использовании здесь имени, *идентичного* имени, определенному в поле процессов Interact List для процедуры offerBySQL, будет использоваться SQL из поля процессов, а не введенный здесь SQL.

Значение по умолчанию

Нет

SQL

Описание

Содержит запрос SQL, который будет вызываться этим шаблоном. Запрос SQL может содержать ссылки на имена переменных, которые являются частью данных сеанса посетителя (профиля). Например, `select * from MyOffers where category = ${preferredCategory}` ссылается на сеанс, содержащий переменную с именем `preferredCategory`.

Необходимо сконфигурировать SQL, чтобы запрашивать конкретные таблицы предложений, созданные при разработке для использования этой возможности. Обратите внимание на то, что хранимые процедуры здесь не поддерживаются.

Значение по умолчанию

Нет

Interact | profile | Audience Levels | [AudienceLevelName | Profile Data Services | [источник_данных]

Этот набор свойств конфигурации позволяет задать имена таблиц, необходимых для дополнительных функций Interact. Вы должны задать имя таблицы, только если вы используете связанную функцию. Категория Службы данных профиля предоставляет информацию о встроенном источнике данных (называется База данных), создаваемом для всех уровней аудитории и предварительно конфигурируемом с приоритетом 100. Однако это можно изменить или отключить. Эта категория содержит также шаблон для дополнительных внешних источников данных. Если щелкнуть по этому шаблону с названием **Внешние службы данных**, можно определить описанные здесь параметры конфигурации.

New category name

Описание

(Недоступно для записи базы данных по умолчанию.) Имя источника данных, который вы определяете. Вводимое здесь имя должно быть уникальным среди источников данных для одного уровня аудитории.

Значение по умолчанию

Нет

Допустимые значения

Допускается любая текстовая строка.

enabled

Описание

Если задано значение `True`, этот источник данных включен для уровня аудитории, которому он назначен. Если задано значение `False`, Interact не использует этот источник данных для этого уровня аудитории.

Значение по умолчанию

`True`

Допустимые значения

True | False

className

Описание

(Недоступно для записи базы данных по умолчанию.) Полное имя класса источника данных, реализующего `IInteractProfileDataService`.

Значение по умолчанию

Нет.

Допустимые значения

Строка, представляющая полное имя класса.

classPath

Описание

(Недоступно для записи базы данных по умолчанию.) Дополнительный параметр конфигурации, представляющий путь для загрузки этого класса реализации источника данных. Если пропустить этот параметр, по умолчанию используется путь класса содержащего прикладную программу сервера.

Значение по умолчанию

Не показано, но если никакое значение здесь не представлено, по умолчанию используется путь классов содержащего прикладную программу сервера.

Допустимые значения

Строка, представляющая путь классов.

priority

Описание

Приоритет этого источника данных по этому уровню аудитории. Это должно быть уникальное значение по всем источникам данных для каждого уровня аудитории. (То есть если для источника данных задан приоритет 100, ни у какого другого источника данных на этом уровне аудитории не может быть приоритета 100.)

Значение по умолчанию

`100` для базы данных по умолчанию, `200` для пользовательского источника данных

Допустимые значения

Допускается любое неотрицательное целое число.

Interact | offerserving

Эти свойства конфигурации задают общие свойства конфигурации обучения. Если вы используете встроенное обучение для настройки реализации обучения, используйте свойства конфигурации для среды разработки.

offerTieBreakMethod

Описание

Свойство `offerTieBreakMethod` определяет поведение представления предложения, когда у двух предложений эквивалентные (связанные) оценки. Если задать для этого свойства значение по умолчанию Random, Interact представляет случайный выбор из всех предложений с одинаковыми оценками. Если задать для этой конфигурации значение Newer Offer, Interact представляет более новое предложение (на основании его большего значения ID) до более старого (с меньшим ID предложения), если оценки всех предложений одинаковые.

Примечание:

У Interact есть дополнительная возможность, позволяющая администратору сконфигурировать систему для возврата предложений в случайном порядке

независимо от оценки, задав опцию percentRandomSelection (Campaign | partitions | [partition_number] | Interact | learning | percentRandomSelection). Описанное здесь свойство offerTieBreakMethod используется только в том случае, если для percentRandomSelection задан ноль (отключено).

Значение по умолчанию

Случайные

Допустимые значения

Random | Newer Offer

optimizationType

Описание

Свойство optimizationType указывает, будет ли Interact использовать механизм обучения, чтобы помочь в назначении предложений. Значение NoLearning указывает, что Interact не использует обучение. Если задано значение BuiltInLearning, Interact использует Байесовский механизм обучения, встроенный в Interact. Если задано значение ExternalLearning, Interact будет использовать указанный вами механизм обучения. Выбрав значение ExternalLearning, вы должны задать свойства externalLearningClass и externalLearningClassPath.

Значение по умолчанию

NoLearning

Допустимые значения

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

Описание

Максимальное время (в миллисекундах), в течение которого сервер среды выполнения ожидает, когда интерактивная потоковая диаграмма завершится, прежде чем получать предложения.

Значение по умолчанию

5000

treatmentCodePrefix

Описание

Префикс, присоединяемый к кодам процедур.

Значение по умолчанию

Значения по умолчанию нет.

effectiveDateBehavior

Описание

Определяет, должен ли Interact использовать дату вступления в силу предложения при фильтрации представляемых посетителю предложений. Возможные значения:

- -1 указывает, что Interact должен игнорировать дату вступления в силу для предложения.

0 указывает, что Interact должен использовать дату вступления в силу для фильтрации, то есть при дате вступления в силу предложения, предшествующей или равной текущей дате, предложение представляется посетителям.

Если задано значение **effectiveDateGracePeriod**, для определения, представлять ли предложение, используется также льготный период.

- Любое положительное целое число указывает, что Interact должен использовать текущую дату плюс значение этого свойства, чтобы определить, представлять ли предложение посетителям, так что при дате вступления в силу предложения, предшествующей текущей дате плюс значение этого свойства, предложение представляется посетителям.

  Если задано значение **effectiveDateGracePeriod**, для определения, представлять ли предложение, используется также льготный период.

Значение по умолчанию

-1

effectiveDateGracePeriodOfferAttr

Описание

Задает имя пользовательского атрибута в определении предложения, указывающего льготный период даты вступления в силу. Например, можно сконфигурировать это свойство со значением `AltGracePeriod`. Затем можно определить предложения с пользовательским атрибутом `AltGracePeriod`, используемым для указания количества дней, которые будут использоваться как льготный период со свойством **effectiveDateBehavior**.

Допустим, что вы создали новый шаблон предложения с датой вступления в силу в 10 дней от текущей даты и включили пользовательский атрибут `AltGracePeriod`. При создании предложения по этому шаблону, если вы задали для `AltGracePeriod` значение 14 дней, предложение будет представляться посетителям, так как текущая дата находится в льготном периоде даты вступления в силу предложения.

Значение по умолчанию

Пустое поле

alwaysLogLearningAttributes

Описание

Указывает, должен ли Interact записывать в файлы журнала информацию об атрибутах посетителей, используемых модулем обучения. Обратите внимание на то, что задание для этого параметра значения true может повлиять на производительность обучения и размеры файлов журнала.

Значение по умолчанию

False

Interact | offerserving | Конфигурация встроенного обучения

Эти свойства конфигурации задают параметры записи в базу данных для встроенного обучения. Чтобы настроить реализацию обучения, используйте свойства конфигурации для среды разработки.

версия

Описание

Можно выбрать 1 или 2. Версия 1 - это базовая версия конфигурации, не использующая параметры для определения пределов потоков и записей. Версия 2 - это расширенная версия конфигурации, позволяющая задавать параметры потоков и записей для повышения производительности. Эти параметры выполняют агрегацию и удаление, когда достигаются предельные значения этих параметров.

Значение по умолчанию

1

insertRawStatsIntervalInMinutes

Описание

Время в минутах, в течение которого модуль обучения Interact ждет перед вставкой дополнительных строк в промежуточные таблицы обучения. Вам может потребоваться изменить это время в соответствии с объемом данных, которые модуль обучения обрабатывает в вашей среде.

Значение по умолчанию

5

Допустимые значения

Положительное целое число

aggregateStatsIntervalInMinutes

Описание

Время в минутах, в течение которого модуль обучения Interact выжидает между агрегированиями данных в таблицах статистики обучения. Вам может потребоваться изменить это время в соответствии с объемом данных, которые модуль обучения обрабатывает в вашей среде.

Значение по умолчанию

15

Допустимые значения

Целочисленное значение, больше нуля.

autoAdjustPercentage

Описание

Значение, определяющее процентную долю данных, для которых при запуске агрегации будет сделана попытка их обработки на основании показателей предыдущего запуска. По умолчанию для этого значения задан ноль, то есть агрегатор обрабатывает все промежуточные записи и эта функция автоматической корректировки отключена.

Значение по умолчанию

0

Допустимые значения

Число от 0 до 100.

enableObservationModeOnly

Описание

Если задано значение `True`, включает режим обучения, при котором Interact собирает данные для обучения, не используя их для рекомендаций или оценки предложений. Это позволяет выполнить самообучение в режиме запуска, пока не будет определено, что собрано достаточно данных для рекомендаций.

Значение по умолчанию

False

Допустимые значения

True | False

excludeAbnormalAttribute

Описание

Это параметр, определяющий, отмечать ли атрибуты как недопустимые. Если задано значение `IncludeAttribute`, аномальные атрибуты будут включены и не будут помечены как недействительные. Если задать значение `ExcludeAttribute`, аномальные атрибуты будут исключены и помечены как недействительные.

Значение по умолчанию

IncludeAttribute

Допустимые значения

IncludeAttribute | ExcludeAttribute

Interact | offerserving | Built-in Learning Config | Parameter Data | [имя_параметра]

Эти свойства конфигурации задают все параметры для внешнего модуля обучения.

numberOfThreads

Описание

Максимальное число потоков, используемых агрегатором обучения для обработки данных. Допустимое значение - это положительное целое число не больше максимального количества соединений, сконфигурированных в источнике данных обучения. Этот параметр используется только агрегатором версии 2.

Значение по умолчанию

10

maxLogTimeSpanInMin

Описание

Если выбрана версия 1 агрегатора, промежуточные записи можно обрабатывать итерационно, чтобы исключить наложение больших пакетов базы данных. В этом случае промежуточные записи обрабатываются блоками (чанками) итерационно в одном цикле агрегирования. Значение этого параметра определяет максимальный интервал времени для промежуточных записей, которые агрегатор пытается обработать в каждой итерации. Этот интервал времени основан на значении поля LogTime, связанного с каждой промежуточной записью, и будут обрабатываться только те записи, для которых LogTime попадает в самое раннее окно

времени. Допустимое значение - это неотрицательное целое число. Если выбрано значение 0, предел не устанавливается, то есть в одной итерации будут обрабатываться все промежуточные записи.

Значение по умолчанию

0

maxRecords

Описание

Если выбрана версия 2 агрегатора, промежуточные записи можно обрабатывать итерационно, чтобы исключить наложение больших пакетов базы данных. В этом случае промежуточные записи обрабатываются блоками (чанками) итерационно в одном цикле агрегирования. Значение этого параметра определяет максимальное число промежуточных записей, которые агрегатор пытается обработать в каждой итерации. Допустимое значение - это неотрицательное целое число. Если выбрано значение 0, предел не устанавливается, то есть в одной итерации будут обрабатываться все промежуточные записи.

Значение по умолчанию

0

value

Описание

Значение для любого параметра, который требуется классом для встроенного модуля обучения.

Значение по умолчанию

Значения по умолчанию нет.

Interact | offerserving | External Learning Config

Эти свойства конфигурации задают параметры класса для внешнего модуля обучения, написанного вами с использованием API обучения.

class

Описание

Если для `optimizationType` задано значение `ExternalLearning`, задайте для `externalLearningClass` имя класса внешнего механизма обучения.

Значение по умолчанию

Значения по умолчанию нет.

Доступность

Это свойство применимо, только если для параметра `optimizationType` задано значение `ExternalLearning`.

classPath

Описание

Если для `optimizationType` задано значение `ExternalLearning`, задайте для `externalLearningClass` путь класса внешнего механизма обучения.

Путь классов должен ссылаться на файлы jar на сервере среды выполнения. Если вы используете группу серверов и все серверы выполнения используют один и тот же экземпляр Marketing Platform, на каждом сервере должна быть копия файла jar в одном и том же месте. Путь классов должен состоять из абсолютных расположений файлов jar, разделенных разделителем путей в соответствии с операционной системой на сервере среды выполнения, например, точка с запятой (;) в системах Windows и двоеточие (:) - в системах UNIX. Каталоги, содержащие файлы классов, не принимаются. Например, в системе Unix: `/path1/file1.jar:/path2/file2.jar`.

Этот путь классов должен содержать менее 1024 символов. Вы можете использовать файл манифеста в файле .jar, чтобы указать другие файлы .jar, так что в пути классов должен появляться только один файл .jar.

Значение по умолчанию

Значения по умолчанию нет.

Доступность

Это свойство применимо, только если для параметра `optimizationType` задано значение `ExternalLearning`.

Interact | offerserving | Внешняя конфигурация обучения | Данные параметра | [parameterName]

Эти свойства конфигурации задают все параметры для внешнего модуля обучения.

значение

Описание

Значение любого параметра, который требуется для класса внешнего модуля обучения.

Значение по умолчанию

Значения по умолчанию нет.

Пример

Если внешнему модулю обучения требуется путь программы алгоритма разрешения, нужно создать категорию параметров `solverPath` и задать свойство `value` как путь программы.

Interact | offerserving | Constraints

Эти свойства конфигурации определяют ограничения, существующие для процесса представления предложения.

maxOfferAllocationInMemoryPerInstance

Описание

Размер блока предложений. Interact хранит пул предложений в памяти, так что система не должна запрашивать базу данных при всяком возвращении предложения. При всяком возвращении предложения этот пул корректируется. Когда пул исчерпывается, Interact получает другой блок предложений, чтобы заполнить пул.

Значение по умолчанию

1000

Допустимые значения

Целочисленное значение, больше 0.

maxDistributionPerIntervalPerInstanceFactor

Описание

Процентная доля ограничений данного выделения предложений, чтобы сервер среды выполнения поддерживал распределение по серверам среды выполнения.

Значение по умолчанию

100

Допустимые значения

Целое число от 0 до 100.

constraintCleanupIntervalInDays

Описание

Как часто отключенные значения вычищаются из таблицы UACI_OfferCount. Значение меньше 1 отключает эту опцию.

Значение по умолчанию

7

Допустимые значения

Целочисленное значение, больше 0.

Interact | services

Свойства конфигурации в этой категории задают параметры для всех служб, которые управляют сбором данных хронологии контактов и ответов, а также статистики для отчетов и записи в системные таблицы среды выполнения.

externalLoaderStagingDirectory

Описание

Это свойство задает расположение промежуточного каталога для утилиты загрузки базы данных.

Значение по умолчанию

Значения по умолчанию нет.

Допустимые значения

Путь относительно каталога установки Interact или абсолютный путь промежуточного каталога.

Если вы включите утилиту загрузки базы данных, вы должны будете задать свойство `cacheType` в категориях `contactHist` и `responstHist` как `Внешний файл загрузчика`.

Interact | services | contactHist

Свойства конфигурации в этой категории задают параметры для службы, собирающей данные для промежуточных таблиц хронологии контактов.

enableLog

Описание

Если задано значение `true`, служба, собирающая данные для записи хронологии контактов, будет включена. Если задано `false`, сбор данных не производится.

Значение по умолчанию

True

Допустимые значения

True | False

cacheType

Описание

Указывает, будут ли данные, собранные для хронологии контактов, храниться в памяти (`Кэш памяти`) или в файле (`Внешний файл загрузчика`). `Внешний файл загрузчика` можно использовать, только если вы сконфигурировали Interact для использования утилиты загрузки базы данных.

Если вы выберете `Кэш памяти`, используйте параметры категории `cache`. Если вы выберете `Внешний файл загрузчика`, используйте параметры категории `fileCache`.

Значение по умолчанию

Кэш памяти

Допустимые значения

Кэш памяти | Внешний файл загрузчика

Interact | services | contactHist | cache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей данные для промежуточной таблицы хронологии контактов.

threshold

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранные данные хронологии контактов в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | contactHist | fileCache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей данные хронологии контактов, если вы используете утилиту загрузки базы данных.

threshold

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранные данные хронологии контактов в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | defaultedStats

Свойства конфигурации в этой категории задают параметры для службы, собирающей статистику числа использований строки по умолчанию для точки взаимодействия.

enableLog

Описание

Если задано значение `true`, будет включена служба, собирающая статистику числа использований строк по умолчанию для точки взаимодействия, используемое для таблицы `UACI_DefaultedStat`. Если задано значение `false`, сбор статистики строк по умолчанию не производится.

Если вы не используете отчеты IBM, вы можете задать для этого свойства значение `false`, так как сбор данных не требуется.

Значение по умолчанию

True

Допустимые значения

True | False

Interact | services | defaultedStats | cache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей статистику числа использований строки по умолчанию для точки взаимодействия.

threshold

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранную статистику строк по умолчанию в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | eligOpsStats

Свойства конфигурации в этой категории задают параметры для службы, записывающей статистику предложений, соответствующих требованиям.

enableLog

Описание

Если задано значение `true`, служба, собирающая статистику предложений, соответствующих требованиям. Если задано `false`, сбор статистики предложений, соответствующих требованиям, не производится.

Если вы не используете отчеты IBM, вы можете задать для этого свойства значение `false`, так как сбор данных не требуется.

Значение по умолчанию

True

Допустимые значения

True | False

Interact | services | eligOpsStats | cache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей статистику предложений, отвечающих требованиям.

threshold

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранную статистику предложений, отвечающих требованиям, в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | eventActivity

Свойства конфигурации в этой категории задают параметры для службы, собирающей статистику действий событий.

enableLog

Описание

Если задано значение `true`, служба, собирающая статистику действий событий, будет включена. Если задано `false`, сбор статистики событий не производится.

Если вы не используете отчеты IBM, вы можете задать для этого свойства значение `false`, так как сбор данных не требуется.

Значение по умолчанию

True

Допустимые значения

True | False

Interact | services | eventActivity | cache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей статистику действий событий.

threshold

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранную статистику действий событий в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | eventPattern

Свойства конфигурации в категории `eventPattern` определяют параметры для службы, собирающей статистику операций с паттернами событий.

persistUnknownUserStates

Описание

Определяет, сохраняются ли в базе данных состояния паттернов событий для неизвестного ID аудитории (посетителя). По умолчанию после завершения сеанса в базе данных сохраняются состояния всех измененных паттернов событий, связанных с ID аудитории посетителя, если этот ID аудитории известен (то есть профиль посетителя можно найти в источнике данных профилей).

Свойство persistUnknownUserStates определяет, что случится, если ID аудитории неизвестен. По умолчанию для этого свойства задано значение False, и для неизвестных ID аудитории состояния паттерна событий отбрасываются в конце сеанса.

Если задать для этого свойства значение True, состояния паттернов событий неизвестных пользователей (чей профиль невозможно найти в сконфигурированной службе данных профиля) будут сохраняться.

Значение по умолчанию

False

Допустимые значения

True | False

mergeUnknowUserInSessionStates

Описание

Определяет, как сохраняются состояния паттерна событий для неизвестных ID аудитории (посетителей). Если ID аудитории переключается в середине сеанса, Interact пытается загрузить сохраненные состояния паттернов событий для нового ID аудитории из таблицы базы данных. Если ранее ID аудитории не был известен и вы задали для свойства mergeUnknowUserInSessionStates значение True, принадлежащие предыдущему ID аудитории в том же сеансе операции событий пользователя будут объединены с новым ID аудитории.

Значение по умолчанию

False

Допустимые значения

True | False

enableUserEventLog

Описание

Определяет, будут ли операции событий пользователя вноситься в журнал в базе данных.

Значение по умолчанию

False

Допустимые значения

True | False

Interact | services | eventPattern | userEventCache

Свойства конфигурации в категории userEventCache задают параметры, которые определяют, когда операция события перемещается из кэша для сохранения в базе данных.

порог

Описание

Определяет максимальное количество состояний паттернов событий, которые можно хранить в кэше состояний паттернов событий. При достижении этого предела последние использованные состояния вычищаются из кэша.

Значение по умолчанию

100

Допустимые значения

Желательное количество состояний паттернов событий для хранения в кэше.

insertPeriodInSecs

Описание

Определяет максимальное время в секундах, на которое операции событий пользователя ставятся в очередь в памяти. При достижении указанного этим свойством предела эти операции сохраняются в базе данных.

Значение по умолчанию

3600 (60 минут)

Допустимые значения

Нужный срок в секундах.

Interact | services | eventPattern | advancedPatterns

Свойства конфигурации в этой категории управляют, будет ли включена интеграция с Interact Advanced Patterns, и определяют интервалы времени ожидания для соединений с Interact Advanced Patterns.

enableAdvancedPatterns

Описание

Если задано значение `true`, включает интеграцию с Interact Advanced Patterns. Если задано значение `false`, интеграция не включается. Если ранее интеграции была включена, Interact использует последние состояния паттернов, полученные от Interact Advanced Patterns.

Значение по умолчанию

True

Допустимые значения

True | False

connectionTimeoutInMilliseconds

Описание

Максимальное время на установление соединения HTTP среды реального времени Interact с Interact Advanced Patterns. Если время ожидания требования истекает, Interact использует сохраненные данные от паттернов.

Значение по умолчанию

30

readTimeoutInMilliseconds

Описание

Максимальное время, которое можно затратить на получение данных после установления соединения HTTP между средой реального времени Interact и Interact Advanced Patterns и отправки требования к Interact Advanced Patterns на получение состояния паттерна событий. Если время ожидания требования истекает, Interact использует последние сохраненные данные от паттернов.

Значение по умолчанию

100

connectionPoolSize

Описание

Размер пула соединений HTTP для связи между средой реального времени Interact и Interact Advanced Patterns.

Значение по умолчанию

10

Interact | services | eventPattern | advancedPatterns | autoReconnect

Свойства конфигурации в этой категории определяют параметры для возможности автоматического восстановления соединения в интеграции с Interact Advanced Patterns.

включить

Описание

Определяет, будет ли система автоматически восстанавливать соединение, если возникнут проблемы соединения между средой реального времени Interact и Interact Advanced Patterns. Значение по умолчанию **True** включает эту возможность.

Значение по умолчанию

True

Допустимые значения

True | False

durationInMinutes

Описание

Это свойство определяет интервал времени в минутах, в течение которого система проверяет повторяющиеся проблемы соединения, возникающие между средой реального времени Interact и Interact Advanced Patterns.

Значение по умолчанию

10

numberOfFailuresBeforeDisconnect

Описание

Это свойство задает допустимое число неудачных попыток соединения за указанный период времени, прежде чем система автоматически отсоединится от Interact Advanced Patterns.

Значение по умолчанию

3

consecutiveFailuresBeforeDisconnect

Описание

Определяет, будет ли возможность автоматического восстановления соединения проверять только последовательные неудачные попытки соединения между средой реального времени Interact и Interact Advanced Patterns. Если задать для этого свойства значение **False**, будут оцениваться все неудачные попытки за заданный интервал времени.

Значение по умолчанию

True

sleepBeforeReconnectDurationInMinutes

Описание

Система выжидает указанное в этом свойстве время в минутах, прежде чем выполнить повторное соединение после того, как система разорвала соединение из-за повторяющихся сбоев, как это задано в других свойствах этой категории.

Значение по умолчанию

5

sendNotificationAfterDisconnect

Описание

Это свойства определяет, будет ли система отправлять уведомления по электронной почте при возникновении сбоя соединения. Сообщение с уведомлением включает в себя имя экземпляра реального времени Interact, для которого произошел сбой, и интервал времени до восстановления соединения, как указано свойством **sleepBeforeReconnectDurationInMinutes**. Значение по умолчанию **True** предполагает, что уведомления отправляются.

Значение по умолчанию

True

Interact | services | customLogger

Свойства конфигурации в этой категории задают параметры для службы, собирающей пользовательские данные для записи в таблицу (событие, использующее параметр события `UACICustomLoggerTableName`).

enableLog

Описание

Если задано значение `true`, включается пользовательская функция записи в таблицу. Если задано значение `false`, параметр события `UACICustomLoggerTableName` не будет действовать.

Значение по умолчанию

True

Допустимые значения

True | False

Interact | services | customLogger | cache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей пользовательские данные для таблицы (событие, использующее параметр события UACICustomLoggerTableName).

threshold

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранные пользовательские данные в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | responseHist

Свойства конфигурации в этой категории задают параметры для службы, записывающей данные в промежуточные таблицы хронологии ответов.

enableLog

Описание

Если задано значение true, служба, записывающая данные в промежуточные таблицы хронологии ответов, будет включена. Если задано значение false, никакие данные не будут записываться в промежуточные таблицы хронологии ответов.

Промежуточная таблица хронологии ответов, заданная свойством responseHistoryTable для уровня аудитории. Значение по умолчанию: UACI_RHStaging.

Значение по умолчанию

True

Допустимые значения

True | False

cacheType

Описание

Указывает, хранится ли кэш в памяти, или в файле. Файл внешнего загрузчика можно использовать только в том случае, если вы сконфигурировали Interact для использования утилиты загрузчика базы данных.

Если вы выберете Кэш памяти, используйте параметры категории cache. Если вы выберете Внешний файл загрузчика, используйте параметры категории fileCache.

Значение по умолчанию

Кэш памяти

Допустимые значения

Кэш памяти | Внешний файл загрузчика

actionOnOrphan

Описание

Этот параметр определяет, что делать с событиями ответа, у которых нет соответствующих событий контактов. Если задать значение `NoAction`, событие ответа обрабатывается так же, как при размещении соответствующего события контакта. Если задать значение `Warning`, событие ответа обрабатывается так же, как если бы было размещено соответствующее событие контакта, но сообщение с предупреждением записывается в файл `interact.log`. Если задать значение `Skip`, ответ даже не обрабатывается, и в файл `interact.log` записывается сообщение об ошибке. Выбранный вами здесь параметр будет действовать независимо от того, включена ли запись хронологии ответов в журнал.

Значение по умолчанию

NoAction

Допустимые значения

NoAction | Warning | Skip

Interact | services | responseHist | cache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей данные хронологии ответов.

порог

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранные данные хронологии ответов в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | response Hist | responseTypeCodes

Свойства конфигурации в этой категории определяют настройки для службы хронологии ответов.

Имя новой категории

Описание

Имя кода вашего типа ответа.

код

Описание

Пользовательский код для вашего типа ответа.

Значение по умолчанию

Пользовательский код, добавленный в таблицу UA_UsrResponseType.

действие

Описание

Действие, соответствующее пользовательскому коду типа ответа.

Действие, определенное для опубликованного события, замещает определенное здесь действие. Поэтому, если событие logAccept публикуется без responseTypeCode, оно рассматривается как событие принятия. Если событие logAccept публикуется с существующим в этой конфигурации responseTypeCode, для определения, есть ли это действие принятия, используется сконфигурированное действие. Если событие logAccept публикуется с responseTypeCode, которого нет в этой конфигурации, данное событие не рассматривается как событие принятия. Когда событие рассматривается как событие принятия, статистика обучения соответствующим образом изменяется, если обучение включено. Правила выражения предложения оцениваются, если есть одно из них на основе принятия этого предложения.

Значение по умолчанию

Нет

Допустимые значения

LogAccept | LogReject | None

Interact | services | responseHist | fileCache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей данные хронологии ответов, если вы используете утилиту загрузки базы данных.

threshold

Описание

Число собранных записей, по достижении которого Interact запишет их в базу данных.

`responseHist` - Таблица, заданная свойством `responseHistoryTable` для уровня аудитории. Значение по умолчанию: `UACI_RHStaging`.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в базу данных.

Значение по умолчанию

3600

Interact | services | crossSessionResponse

Свойства конфигурации в этой категории задают общие параметры для службы crossSessionResponse и процесса xsession. Вы должны сконфигурировать эти параметры, только если вы используете отслеживание ответов для разных сеансов Interact.

enableLog

Описание

Если задано значение `true`, будет включена служба crossSessionResponse, и Interact будет записывать данные в промежуточные таблицы отслеживания ответов для разных сеансов. Если задано `false`, служба crossSessionResponse отключается.

Значение по умолчанию

False

xsessionProcessIntervalInSecs

Описание

Время в секундах между запусками процесса xsession. Этот процесс перемещает данные из промежуточных таблиц отслеживания ответов для разных сеансов в промежуточную таблицу хронологии ответов и встроенный модуль обучения.

Значение по умолчанию

180

Допустимые значения

Целочисленное значение, больше нуля.

purgeOrphanResponseThresholdInMinutes

Описание

Время в минутах, в течение которого служба crossSessionResponse ожидает, прежде чем отметить все ответы, не соответствующие контактам, в таблицах хронологии контактов и ответов.

Если для ответа нет соответствующих данных в таблицах хронологии контактов и ответов, то по прошествии `purgeOrphanResponseThresholdInMinutes` минут Interact пометит ответ значением -1 в столбце `Пометка` в промежуточной таблице xSessResponse. Вы можете затем вручную сопоставить или удалить эти ответы.

Значение по умолчанию

180

Interact | services | crossSessionResponse | cache

Свойства конфигурации в этой категории задают параметры кэша для службы, собирающей данные ответов между разными сеансами.

threshold

Описание

Число собранных записей, по достижении которого служба flushCacheToDB запишет собранные данные ответов для разных сеансов в базу данных.

Значение по умолчанию

100

insertPeriodInSecs

Описание

Время в секундах между принудительными операциями записи в таблицу `XSessResponse`.

Значение по умолчанию

3600

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

Свойства в этом разделе указывают, как отслеживание ответов для разных сеансов сопоставляет коды процедур с хронологией контактов и ответов.

SQL

Описание

Это свойство указывает, будет ли Interact использовать SQL, сгенерированный системой, или пользовательский SQL, заданный в свойстве `OverrideSQL`.

Значение по умолчанию

Использовать SQL, сгенерированный системой

Допустимые значения

Использовать SQL, сгенерированный системой | Переопределить SQL

OverrideSQL

Описание

Если вы не используете команду SQL по умолчанию для сопоставления кода процедуры с хронологией контактов и ответов, введите здесь SQL или хранимую процедуру.

Если для параметра `SQL` задана опция `Использовать SQL, сгенерированный системой`, это значение игнорируется.

Значение по умолчанию

useStoredProcedure

Описание

Если задано значение true, параметр OverrideSQL должен содержать ссылку на хранимую процедуру, которая сопоставляет код процедуры с хронологией контактов и ответов.

Если задано значение false, параметр OverrideSQL (если он используется) должен представлять собой запрос SQL.

Значение по умолчанию

нет

Допустимые значения

true | false

Тип

Описание

Связанный тип TrackingCodeType, заданный в таблице UACI_TrackingType в таблицах среды выполнения. Если вы не исправляли таблицу UACI_TrackingType, для свойства Type должно быть задано значение 1.

Значение по умолчанию

1

Допустимые значения

Целое число, заданное в таблице UACI_TrackingType.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

Свойства в этом разделе указывают, как отслеживание ответов для разных сеансов сопоставляет коды предложений с хронологией контактов и ответов.

SQL

Описание

Это свойство указывает, будет ли Interact использовать SQL, сгенерированный системой, или пользовательский SQL, заданный в свойстве OverrideSQL.

Значение по умолчанию

Использовать SQL, сгенерированный системой

Допустимые значения

Использовать SQL, сгенерированный системой | Переопределить SQL

OverrideSQL

Описание

Если вы не используете команду SQL по умолчанию для сопоставления кода предложения с хронологией контактов и ответов, введите здесь SQL или хранимую процедуру.

Если для параметра SQL задана опция Использовать SQL, сгенерированный системой, это значение игнорируется.

Значение по умолчанию

useStoredProcedure

Описание

Если задано значение true, параметр OverrideSQL должен содержать ссылку на хранимую процедуру, которая сопоставляет код предложения с хронологией контактов и ответов.

Если задано значение false, параметр OverrideSQL (если он используется) должен представлять собой запрос SQL.

Значение по умолчанию

нет

Допустимые значения

true | false

Тип

Описание

Связанный тип TrackingCodeType, заданный в таблице UACI_TrackingType в таблицах среды выполнения. Если вы не исправляли таблицу UACI_TrackingType, для свойства Type должно быть задано значение 2.

Значение по умолчанию

2

Допустимые значения

Целое число, заданное в таблице UACI_TrackingType.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

Свойства в этом разделе указывают, как отслеживание ответов для разных сеансов сопоставляет заданный пользователем альтернативный код с хронологией контактов и ответов.

Имя

Описание

Это свойство задает имя для альтернативного кода. Оно должно совпадать со значением Name в таблице UACI_TrackingType в таблицах среды выполнения.

Значение по умолчанию

OverrideSQL

Описание

Команда SQL или хранимая процедура для сопоставления альтернативного кода с хронологией контактов и ответов по коду предложения или коду процедуры.

Значение по умолчанию

useStoredProcedure

Описание

Если задано значение true, параметр OverrideSQL должен содержать ссылку на хранимую процедуру, которая сопоставляет альтернативный код с хронологией контактов и ответов.

Если задано значение false, параметр OverrideSQL (если он используется) должен представлять собой запрос SQL.

Значение по умолчанию

нет

Допустимые значения

true | false

Тип

Описание

Связанный тип TrackingCodeType, заданный в таблице UACI_TrackingType в таблицах среды выполнения.

Значение по умолчанию

3

Допустимые значения

Целое число, заданное в таблице UACI_TrackingType.

Interact | services | threadManagement | contactAndResponseHist

Свойства конфигурации в этой категории задают параметры управления потоками для служб, собирающих данные для промежуточных таблиц хронологии контактов и ответов.

corePoolSize

Описание

Число потоков, которые нужно сохранить в пуле, даже если они бездействуют, для сбора данных хронологии контактов и ответов.

Значение по умолчанию

5

maxPoolSize

Описание

Максимальное число потоков, которые нужно сохранить в пуле, для сбора данных хронологии контактов и ответов.

Значение по умолчанию

5

keepAliveTimeSecs

Описание

Если число потоков превышает основное, это максимальное время, превышающее время ожидания бездействующих потоков для новых задач, перед прекращением работы для сбора данных хронологии контактов и ответов.

Значение по умолчанию

5

queueCapacity

Описание

Размер очереди, используемой пулом потоков, для сбора данных хронологии контактов и ответов.

Значение по умолчанию

1000

termWaitSecs

Описание

При завершении работы сервера выполнения это время в секундах, которые следует выждать для завершения сбора данных хронологии контактов и ответов.

Значение по умолчанию

5

Interact | services | threadManagement | allOtherServices

Свойства конфигурации в этой категории задают параметры управления потоками для служб, собирающих статистику соответствия предложений требованиям, статистику активности событий, статистику использования строк по умолчанию и пользовательский журнал для табличных данных.

corePoolSize

Описание

Число потоков, которые нужно сохранить в пуле, даже если они бездействуют, для служб, собирающих статистику соответствия предложений требованиям, статистику активности событий, статистику использования строк по умолчанию и пользовательский журнал для табличных данных.

Значение по умолчанию

5

maxPoolSize

Описание

Максимальное число потоков, которые нужно сохранить в пуле, для служб, собирающих статистику соответствия предложений требованиям, статистику активности событий, статистику использования строк по умолчанию и пользовательский журнал для табличных данных.

Значение по умолчанию

5

keepAliveTimeSecs

Описание

Если число потоков превышает основное, существует максимальное время, превышающее время ожидания бездействующих потоков для новых задач,

перед прекращением работы служб, собирающих статистику соответствия предложений требованиям, статистику активности событий, статистику использования строк по умолчанию и пользовательский журнал для табличных данных.

Значение по умолчанию

5

queueCapacity

Описание

Размер очереди, используемой пулом потоков, для служб, собирающих статистику соответствия предложений требованиям, статистику активности событий, статистику использования строк по умолчанию и пользовательский журнал для табличных данных.

Значение по умолчанию

1000

termWaitSecs

Описание

При завершении работы сервера выполнения это время в секундах, которые следует выждать для завершения работы служб потоков, для служб, собирающих статистику соответствия предложений требованиям, статистику активности событий, статистику использования строк по умолчанию и пользовательский журнал для табличных данных.

Значение по умолчанию

5

Interact | services | threadManagement | flushCacheToDB

Свойства конфигурации в этой категории задают параметры управления потоками для потоков, которые записывают собранные данные в кэш в таблицы базы данных среды выполнения.

corePoolSize

Описание

Число потоков, которые нужно сохранить в пуле, для запланированных потоков, которые записывают кэшированные данные в хранилище данных.

Значение по умолчанию

5

maxPoolSize

Описание

Максимальное число потоков, которые нужно сохранить в пуле, для запланированных потоков, которые записывают кэшированные данные в хранилище данных.

Значение по умолчанию

5

keepAliveTimeSecs

Описание

Если число потоков превышает основное, это максимальное время, превышающее время ожидания бездействующих потоков для новых задач, перед прекращением работы для запланированных потоков, которые записывают кэшированные данные в хранилище данных.

Значение по умолчанию

5

queueCapacity

Описание

Размер очереди, используемой пулом потоков, для запланированных потоков, которые записывают кэшированные данные в хранилище данных.

Значение по умолчанию

1000

termWaitSecs

Описание

При завершении работы сервера выполнения это время в секундах, которые следует выждать для запланированных потоков, которые записывают кэшированные данные в хранилище данных.

Значение по умолчанию

5

Interact | services | threadManagement | eventHandling

Свойства конфигурации в этой категории определяют параметры управления потоками для служб, собирающих данные для обслуживания событий.

corePoolSize

Описание

Число потоков для хранения в пуле, даже если они не используются, для сбора данных для обработки событий.

Значение по умолчанию

1

maxPoolSize

Описание

Максимальное число потоков для хранения в пуле для служб, собирающих данные для обработки событий.

Значение по умолчанию

5

keepAliveTimeSecs

Описание

Когда количество потоков превышает базовое, это максимальное время ожидания избыточными неактивными потоками новых задач, прежде чем будет прерван сбор данных для обработки событий.

Значение по умолчанию

5

queueCapacity

Описание

Размер очереди, используемой пулом потоков для сбора данных обработки событий.

Значение по умолчанию

1000

termWaitSecs

Описание

При отключении сервера среды выполнения это время ожидания в секундах завершения потоков для служб, собирающих данные для обработки событий.

Значение по умолчанию

5

Interact | services | configurationMonitor

Свойства конфигурации в этой категории позволяют включать или отключать интеграцию с Interact Advanced Patterns без необходимости перезапуска среды реального времени Interact, а также определяют интервал для опроса значения свойства, включающего интеграцию.

enable

Описание

Если задано значение `true`, включает службу обновления значения свойства **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**. Если задано значение `false`, вы должны перезапустить среду реального времени Interact при изменении значения свойства **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**.

Значение по умолчанию

False

Допустимые значения

True | False

refreshIntervalInMinutes

Описание

Определяет временной интервал для опроса значения свойства **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**.

Значение по умолчанию

5

Interact | cacheManagement

Этот набор свойств конфигурации определяет параметры для выбора и конфигурирования каждого из поддерживаемых менеджеров кэшей, которые можно использовать для повышения производительности Interact, таких как EHCache, встроенный в вашу установку Interact, дополнительный модуль кэширование WebSphere eXtreme Scale или любая другая внешняя система кэширования.

Используйте свойства конфигурации **Interact | cacheManagement | Cache Managers** для конфигурирования менеджера кэшей, который вы хотите использовать. Используйте свойства конфигурации **Interact | cacheManagement | caches** для указания, какой менеджер кэширования должен использовать Interact для повышения производительности.

Interact | cacheManagement | Cache Managers

Категория Cache Managers задает параметры для решений управления кэшем, которые вы собираетесь использовать с Interact.

Interact | cacheManagement | Cache Managers | EHCache

Категория EHCache определяет параметры для решения управления кэшем EHCache, и вы можете настроить его для повышения производительности Interact.

Interact | Cache Managers | EHCache | Parameter Data

Свойства конфигурации в этой категории управляют тем, как работает система управления кэшами EHCache для повышения производительности Interact.

cacheType

Описание

Серверы среды выполнения Interact можно сконфигурировать в группе серверов, чтобы использовать адрес широковещания для совместного использования данных кэшей. Эту ситуацию называют *распределенным кэшем*. Параметр cacheType задает, используете ли вы встроенный механизм кэширования в EHCache в **локальном** (автономно) или **распределенном** (как в группе серверов среды выполнения) режиме.

Примечание:

Если для параметра cacheType выбрать значение **Distributed**, все совместно использующие кэш серверы должны быть частью одной группы серверов. Кроме этого, необходимо включить режим широковещания между всеми участниками группы серверов.

Значение по умолчанию

Локально

Допустимые значения

Local | Distributed

multicastIPAddress

Описание

Если указать для параметра **cacheType** значение "distributed", кэш конфигурируется для широковещания между всеми участниками группы

серверов среды выполнения Interact. Значение multicastIPAddress - это IP-адрес, используемый для приема всеми серверами Interact для группы серверов.

Этот IP-адрес должен быть уникальным по всем вашим группам серверов.

Значение по умолчанию

230.0.0.1

multicastPort

Описание

Если указать для параметра **cacheType** значение "distributed", параметр **multicastPort** будет указывать порт, используемый для приема всеми серверами Interact для группы серверов.

Значение по умолчанию

6363

overflowToDisk

Описание

Менеджер кэшей EHCache управляет информацией сеанса с использованием доступной памяти. Для сред, где размер сеанса велик из-за большого профиля, поддерживаемое в памяти количество сеансов может быть недостаточно велико для поддержки пользовательского сценария. В таких случаях у EHCache есть дополнительная возможность, разрешающая временно записывать информацию кэшей, превышающую объем доступной памяти, на жесткий диск.

Если для свойства **overflowToDisk** задать значение "yes", каждая виртуальная Java-машина (JVM) сможет обработать больше одновременных сеансов, чем допускала бы доступная память.

Значение по умолчанию

Нет

Допустимые значения

No | Yes

diskStore

Описание

Когда для свойства конфигурации **overflowToDisk** задано значение `Yes`, оно задает дисковый каталог для хранения записей кэшей, которым не хватило доступной памяти. Если этого свойства конфигурации не существует или его значение недопустимо, этот каталог диска создается автоматически во временном каталоге операционной системы по умолчанию.

Значение по умолчанию

Нет

Допустимые значения

Каталог, для которого у веб-программы, содержащей среду выполнения Interact, есть привилегии записи.

(Параметр)

Описание

Шаблон, который можно использовать, чтобы создать пользовательский параметр для использования с менеджером кэшей. Можно задать любое имя параметра и значение, которое он должен иметь.

Для создания пользовательского параметра щелкните по ***(Параметр)*** и введите имя и значение, которое вы хотите присвоить этому параметру. Когда вы щелкнете по **Сохранить изменения**, параметр, который вы создали, будет добавлен к списку в категории Данные параметра.

Значение по умолчанию

Нет

Interact | cacheManagement | Cache Managers | Extreme Scale

Категория Extreme Scale задает параметры для адаптера, чтобы использовать решение по управлению кэшами WebSphere eXtreme Scale, так что вы можете настроить его для повышения производительности Interact.

ClassName

Описание

Полное имя класса, соединяющего Interact с сервером WebSphere eXtreme Scale. Это должно быть `com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`.

Значение по умолчанию

`com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`

ClassPath

Описание

URI положения файла `interact_wxs_adapter.jar`, такой как `file:///IBM/IMS/Interact/lib/interact_wxs_adapter.jar` или `file:///C:/IBM/IMS/Interact/lib/interact_wxs_adapter.jar`. Однако если этот файл jar уже включен в путь классов сервера-хоста прикладной программы, это поле нужно оставить пустым.

Значение по умолчанию

Пустое поле

Interact | Cache Managers | Extreme Scale | Parameter Data

Свойства конфигурации в этой категории управляют адаптером WebSphere eXtreme Scale, который дополнительно включается в вашу установку Interact. Эти параметры нужно сконфигурировать для каждого сервера среды выполнения Interact, действующего как клиент для сетки серверов eXtreme Scale.

catalogPropertyFile

Описание

URI положения файла свойств, используемого для запуска сервера каталогов WebSphere eXtreme Scale. Если для запуска сервера каталогов используется адаптер Extreme Scale, это свойство необходимо задать. В противном случае оно не используется.

Значение по умолчанию

file:///C:/depot/Interact/dev/main/extremescale/config/catalogServer.props

containerPropertyFile

Описание

URI положения файла свойств, используемого для запуска экземпляров контейнера WebSphere eXtreme Scale. Необходимо задать это свойство, если для запуска серверов контейнера WebSphere eXtreme Scale используется включенный компонент сервера. В противном случае оно не используется.

Значение по умолчанию

file:///C:/depot/Interact/dev/main/extremescale/config/containerServer.props

deploymentPolicyFile

Описание

URI положения файла политики внедрения, используемого для запуска сервера каталогов WebSphere eXtreme Scale. Необходимо задать это свойство, если для запуска сервера каталогов WebSphere eXtreme Scale используется включенный компонент сервера. В противном случае оно не используется.

Значение по умолчанию

file:///C:/depot/Interact/dev/main/extremescale/config/deployment.xml

objectGridConfigFile

Описание

URI положения файла конфигурации сетки объектов, используемого для запуска сервера каталогов WebSphere eXtreme Scale, а также для связанного с кэшем компонента, выполняемого одновременно с сервером среды выполнения Interact в той же виртуальной Java-машине (JVM).

Значение по умолчанию

file:///C:/depot/Interact/dev/main/extremescale/config/objectgrid.xml

gridName

Описание

Имя сетки WebSphere eXtreme Scale, содержащей все кэши Interact.

Значение по умолчанию

InteractGrid

catalogURLs

Описание

Содержащий имя хоста URL или IP-адрес и порт, через которые сервер каталогов WebSphere eXtreme Scale принимает соединения.

Значение по умолчанию

Нет

(Параметр)

Описание

Шаблон, который можно использовать, чтобы создать пользовательский параметр для использования его с менеджером кэшей. Можно задать любое имя параметра и значение, которое он должен иметь.

Для создания пользовательского параметра щелкните по ***(Параметр)*** и введите имя и значение, которое вы хотите присвоить этому параметру. Когда вы щелкнете по **Сохранить изменения**, параметр, который вы создали, будет добавлен к списку в категории Данные параметра.

Значение по умолчанию

Нет

Interact | caches

Используйте этот набор свойств конфигурации, чтобы указать, какой поддерживаемый менеджер кэшей, например, Ehcache или кэширование WebSphere eXtreme Scale, нужно использовать для повышения производительности Interact, а также для конфигурирования конкретных свойств кэша конфигурируемого вами сервера среды выполнения.

Сюда входят кэши для хранения данных сеанса, состояний паттерна событий и результатов сегментации. Корректируя эти параметры, можно указать, какое именно решение для кэшей использовать для каждого типа кэширования, а также задать отдельные параметры для управления работой кэшей.

Interact | cacheManagement | caches | InteractCache

Категория InteractCache конфигурирует кэширование для всех объектов сеанса, в том числе данные профиля, результаты сегментации, последние предоставленные процедуры, переданные через методы API параметры и другие объекты, используемые средой выполнения Interact.

Категория InteractCache требуется для правильной работы Interact.

Категорию InteractCache можно сконфигурировать также через внешнюю конфигурацию EHCache для параметров, не поддерживаемых в **Interact | cacheManagement | Caches**. При использовании EHCache необходимо обеспечить правильное конфигурирование InteractCache.

CacheManagerName

Описание

Имя менеджера кэшей, обрабатывающего кэш Interact. Введенное здесь значение должно обозначать один из менеджеров кэшей, определенных свойствами конфигурации **Interact | cacheManagement | Cache Managers**, таких как `EHCache` или `Extreme Scale`.

Значение по умолчанию

EHCache

Допустимые значения

Любой из менеджеров кэшей, определенных свойством конфигурации **Interact | cacheManagement | Cache Managers**.

maxEntriesInCache

Описание

Максимальное число объектов данных сеанса для хранения в этом кэше. При достижении максимального числа объектов данных сеансов и при необходимости сохранения данных для дополнительного сеанса будет удаляться последний из использованных объектов.

Значение по умолчанию

100000

Допустимые значения

Целое число больше 0.

timeoutInSecs

Описание

Время в секундах, прошедшее с момента последнего использования или изменения объекта данных сеанса; используется для определения, когда объект удаляется из кэша.

Примечание: Если вы проводили обновление от версии, предшествующей 9.1, потребуется произвести повторное конфигурирование свойства `timeoutInSecs`, так как оно было перемещено.

Значение по умолчанию

300

Допустимые значения

Целое число больше 0.

Interact | Caches | Interact Cache | Parameter Data

Свойства конфигурации в этой категории управляют кэшем Interact, автоматически используемым в вашей установке Interact. Эти параметры нужно сконфигурировать по отдельности для каждого сервера среды выполнения Interact.

asyncIntervalMillis

Описание

Время в миллисекундах, которое должен выждать менеджер кэшей EHCache, прежде чем реплицировать любые изменения на другие экземпляры среды выполнения Interact. Если это значение не положительно, такие изменения будут реплицироваться синхронно.

По умолчанию это свойство конфигурации не создается. Если вы создаете это свойство, оно используется только в том случае, если EHCache - это менеджер кэшей, а для свойства ehCache **cacheType** задано значение `распределенно`.

Значение по умолчанию

Нет.

(Параметр)

Описание

Шаблон, который можно использовать, чтобы создать пользовательский параметр для использования с кэшами Interact. Можно задать любое имя параметра и значение, которое он должен иметь.

Для создания пользовательского параметра щелкните по ***(Параметр)*** и введите имя и значение, которое вы хотите присвоить этому параметру. Когда вы щелкнете по **Сохранить изменения**, параметр, который вы создали, будет добавлен к списку в категории Данные параметра.

Значение по умолчанию

Нет

Interact | cacheManagement | caches | PatternStateCache

Категория PatternStateCache используется для хостинга состояний паттерна событий и правил подавления предложений в реальном времени. По умолчанию этот кэш сконфигурирован как кэш сквозной записи и сквозного чтения, поэтому Interact пытается использовать первый паттерн событий кэша и данные подавления предложения. Если затребованной записи в кэше нет, реализация кэша загружает ее из источника данных, или через конфигурацию JNDI, или непосредственно с использованием соединения JDBC.

Для использования соединения JNDI Interact соединяется с существующим провайдером источника данных, определенным через заданный сервер с помощью имени JNDI, URL и т.п. Для соединения JDBC необходимо предоставить набор параметров JDBC, включающих в себя имя класса драйвера JDBC, URL базы данных и информацию об аутентификации.

Обратите внимание на то, что при определении нескольких источников JNDI и JDBC используется первый включенный источник JNDI, а если включенных источников JNDI нет, используется первый включенный источник JDBC.

Категория PatternStateCache требуется для правильной работы Interact.

Категорию PatternStateCache можно сконфигурировать также через внешнюю конфигурацию EHCache для параметров, не поддерживаемых в **Interact | cacheManagement | Caches**. При использовании EHCache необходимо убедиться в правильном конфигурировании PatternStateCache.

CacheManagerName

Описание

Имя менеджера кэшей, обрабатывающего кэш состояния паттерна Interact. Введенное здесь значение должно обозначать один из менеджеров кэшей, определенных свойствами конфигурации **Interact | cacheManagement | Cache Managers**, таких как `EHCache` или `Extreme Scale`.

Значение по умолчанию

EHCache

Допустимые значения

Любой из менеджеров кэшей, определенных свойством конфигурации **Interact | cacheManagement | Cache Managers**.

maxEntriesInCache

Описание

Максимальное число состояний паттерна событий для хранения в этом кэше. При достижении максимального числа состояний паттерна событий и при необходимости сохранения дополнительного состояния паттерна событий будет удаляться последний из использованных объектов.

Значение по умолчанию

100000

Допустимые значения

Целое число больше 0.

timeoutInSecs

Описание

Задает интервал времени в секундах для истечения срока хранения объекта состояния паттерна событий в кэше состояний паттерна событий. Когда такой объект состояния остается неиспользуемым в кэше в течение `timeoutInSecs` секунд, он может быть исключен из кэша на основании правила последнего использования. Обратите внимание на то, что значение этого свойства должно быть больше определенного значения свойства `sessionTimeoutInSecs`.

Примечание: Если вы проводили обновление от версии, предшествующей 9.1, потребуется произвести повторное конфигурирование свойства `timeoutInSecs`, так как оно было перемещено.

Значение по умолчанию

300

Допустимые значения

Целое число больше 0.

Interact | Caches | PatternStateCache | Parameter Data:

Свойства конфигурации в этой категории управляют кэшем состояний паттернов, используемым для хранения состояний паттернов событий и правил подавления предложений в реальном времени.

(Параметр)

Описание

Шаблон, который можно использовать для создания пользовательского параметра, чтобы использовать его с кэшем состояний паттернов. Можно задать любое имя параметра и значение, которое он должен иметь.

Для создания пользовательского параметра щелкните по ***(Параметр)*** и введите имя и значение, которое вы хотите присвоить этому параметру. Когда вы щелкнете по **Сохранить изменения**, параметр, который вы создали, будет добавлен к списку в категории Данные параметра.

Значение по умолчанию

Нет

Interact | cacheManagement | caches | PatternStateCache | loaderWriter:

Категория **loaderWriter** содержит конфигурацию загрузчика, взаимодействующего с внешними репозиториями для получения и сохранения паттернов событий.

className

Описание

Полное имя класса для этого загрузчика. Этот класс должен соответствовать требованиями выбранного менеджера кэшей.

Значение по умолчанию

`com.unicacorp.interact.cache.ehcache.loaderwriter.PatternStateEHCacheLoaderWriter`

Допустимые значения

Полное имя класса.

classPath

Описание

Путь к файлу класса загрузчика. Если вы оставили это значение пустым или запись недопустима, используется путь класса для запуска Interact.

Значение по умолчанию

Нет

Допустимые значения

Допустимый путь класса.

writeMode

Описание

Задает режим функции записи для сохранения новых или измененных состояний паттерна событий в кэше. Допустимые опции:

- WRITE_THROUGH. При всяком появлении новой записи или при изменении существующей записи она немедленно вносится в репозитории.
- WRITE_BEHIND. Менеджер кэшей некоторое время выжидает, чтобы собрать несколько записей, а затем пакетом сохраняет их в репозиториях.

Значение по умолчанию

WRITE_THROUGH

Допустимые значения

WRITE_THROUGH или WRITE_BEHIND.

batchSize

Описание

Максимальное число объектов состояния паттерна событий, пакетом сохраняемых функцией записи. Это свойство используется только в том случае, когда для **writeMode** задано значение `WRITE_BEHIND`.

Значение по умолчанию

100

Допустимые значения

Целое значение.

maxDelayInSecs

Описание

Максимальное время в секундах, в течение которого менеджер кэшей выжидает, прежде чем сохранить объект состояния паттерна событий. Это свойство используется только в том случае, когда для **writeMode** задано значение `WRITE_BEHIND`.

Значение по умолчанию

5

Допустимые значения

Целое значение.

Interact | Caches | PatternStateCache | loaderWriter | Parameter Data:

Свойства конфигурации в этой категории управляют загрузчиком кэша состояний паттернов.

(Параметр)

Описание

Шаблон, который можно использовать для создания пользовательского параметра, чтобы использовать его с загрузчиком кэша состояний паттернов. Можно задать любое имя параметра и значение, которое он должен иметь.

Для создания пользовательского параметра щелкните по ***(Параметр)*** и введите имя и значение, которое вы хотите присвоить этому параметру. Когда вы щелкнете по **Сохранить изменения**, параметр, который вы создали, будет добавлен к списку в категории Данные параметра.

Значение по умолчанию

Нет

Interact | cacheManagement | кэши | PatternStateCache | loaderWriter | jndiSettings:

Категория **jndiSettings** содержит конфигурацию для источников данных JNDI, которые загрузчик будет использовать для связи с поддерживающей базой данных. Чтобы создать новый набор параметров JNDI, раскройте категорию **jdniSettings** и щелкните по свойству ***(jndiSetting)***.

(jndiSettings)

Примечание: Когда используется WebSphere Application Server, loaderWriter не соединяется при помощи **jndiSettings**.

Описание

При щелчке по этой категории появляется форма. Чтобы задать источник данных JNDI, введите следующие значения:

- **Имя новой категории** - это имя, которое вы хотите использовать для идентификации данного соединения JNDI.
- Поле **включено** позволяет указать, хотите ли вы, чтобы это соединение JNDI было доступно для использования или нет. Задайте для новых соединений значение `True`.
- **jdniName** - имя JNDI, уже определенное на источнике данных при его конфигурировании.

- **providerUrl** - URL для поиска этого источника данных JNDI. Если вы оставите это поле пустым, используется URL веб-программы, которая содержит модуль времени выполнения Interact.
- **Фабрика начального контекста** - полное имя для класса фабрики начального контекста с целью соединения с провайдером JNDI. Если веб-программа, содержащая модуль времени выполнения Interact, используется для **providerUrl**, оставьте это поле пустым.

Значение по умолчанию

Нет.

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jdbcSettings:

Категория **jdbcSettings** содержит конфигурацию для соединений JDBC, которые загрузчик будет использовать для связи с поддерживающей базой данных. Чтобы создать новый набор параметров JDBC, раскройте категорию **jdbcSettings** и щелкните по свойству ***(jdbcSetting)***.

(jdbcSettings)

Описание

При щелчке по этой категории появляется форма. Чтобы задать источник данных, введите следующие значения:

- **Имя новой категории** - это имя, которое вы хотите использовать для идентификации данного соединения JDBC.
- Поле **включено** позволяет указать, хотите ли вы, чтобы это соединение JDBC было доступно для использования или нет. Задайте для новых соединений значение `True`.
- **driverClassName** - полное имя класса этого драйвера JDBC. Этот класс должен существовать по пути классов, сконфигурированном для запуска сервера кэша хоста.
- **databaseUrl** - URL для поиска этого источника данных JDBC.
- **asmUser** - имя пользования IBM Marketing Software, сконфигурированного с регистрационными данными для соединения с базой данных через это соединение JDBC.
- **asmDataSource** - имя источника данных IBM Marketing Software, сконфигурированного с регистрационными данными для соединения с базой данных через это соединение JDBC.
- **maxConnection** - максимальное число одновременных соединений, которые разрешено устанавливать с базой данных через это соединение JDBC.

Значение по умолчанию

Нет.

Interact | triggeredMessage

Свойства конфигурации в этой категории задают общие настройки для всех инициированных сообщений и поставки канала предложений.

backendProcessIntervalMin

Описание

Это свойство определяет время в минутах, в течение которого поток внутренней системы загружает и обрабатывает задержанные доставленные предложения. Это значение должно быть целым числом. Если это значение нулевое или отрицательное, процесс внутренней системы отключается.

Допустимые значения

Положительное целое число

autoLogContactAfterDelivery

Описание

Если для этого свойства задано значение true, событие контактов автоматически публикуется, как только это предложение отправляется или ставится в очередь для задержанной доставки. Если для этого свойства задано значение false, события контактов не публикуются для исходящих предложений. Это - поведение по умолчанию.

Примечание:

- Если вы хотите захватывать дополнительные атрибуты в хронологии контактов, когда инициируется исходящее сообщение, можно добавить дополнительные пользовательские атрибуты как столбцы в хронологии контактов. При публикации события, которое может инициировать исходящее сообщение, можно передать значения для этих атрибутов в методе `postEvent` как параметры значений имен
- Чтобы параметризовать предложение для исходящего канала, можно назначить предложения в связанной стратегии, внедрить канал, персонализовать предложение и в инициированном сообщении выбрать опцию **Автоматически выбрать следующее лучшее предложение**.

Допустимые значения

True | False

waitForFlowchart

Описание

Это свойство определяет, должна ли потоковая диаграмма дожидаться окончания текущей выполняемой сегментации, а также поведение в случае истечения времени этого ожидания.

DoNotWait: Обработка инициированного сообщения начинается независимо от того, выполняется в настоящее время сегментация или нет. Однако если сегменты используются в правиле соответствия или опция NextBestOffer используется как способ выбора предложения, выполнение TM будет задержано.

OptionalWait : Обработка инициированного сообщения ожидает, пока закончится текущая выполняемая сегментация или истечет ее время ожидания. При истечении времени ожидания в журнал вносится предупреждение, и обработка этого инициированного сообщения продолжается. Это значение по умолчанию.

MandatoryWait: Обработка инициированного сообщения ожидает, пока закончится текущая выполняемая сегментация или истечет ее время ожидания. При истечении времени ожидания в журнал вносится сообщение об ошибке, и обработка этого инициированного сообщения прерывается.

Допустимые значения

DoNotWait | OptionalWait | MandatoryWait

Interact | triggeredMessage | offerSelection

Свойства конфигурации в этой категории задают общие настройки для выбора предложений в инициированных сообщениях.

maxCandidateOffers

Описание

Это свойство определяет максимальное число предложений, соответствующим требованиям, которые механизм возвращает, чтобы получить лучшее предложение для доставки. Есть вероятность, что ни одно из этих возвращенных подходящих предложений нельзя будет отправить на основании выбранного канала. Чем больше есть предложений-кандидатов, тем меньше шанс, что такое случится. Однако увеличение числа предложений-кандидатов может увеличить время обработки.

Допустимые значения

Положительное целое число

defaultCellCode

Описание

Если доставленное сообщение - это результат оценки стратегического правила или записи из таблицы, существует связанная с ним ячейка назначения, и информация этой ячейки используется во всех связанных журналах. Однако если входные данные для выбора предложений - это список конкретных предложений, ячейка назначения недоступна. В этом случае используется значение этого параметра конфигурации. Необходимо убедиться, что во внедрение включены эта ячейка назначения и ее кампания. Простейший способ осуществить это - добавить ячейку во внедренную стратегию.

Interact | triggeredMessage | dispatchers

Свойства конфигурации в этой категории задают общие настройки для всех диспетчеров в инициированных сообщениях.

dispatchingThreads

Описание

Это свойство определяет количество потоков, используемых механизмом для асинхронного вызова диспетчеров. Если это значение - 0 или отрицательное число, вызов диспетчеров синхронный. Значение по умолчанию - 0.

Допустимые значения

Целое число

Interact | triggeredMessage | dispatchers | <имя_диспетчера>

Свойства конфигурации в этой категории задают общие настройки для конкретного диспетчера в инициированных сообщениях.

имя категории

Описание

Это свойство задает имя для этого диспетчера. Это имя должно быть уникальным среди всех диспетчеров.

type

Описание

Это свойство определяет тип диспетчера.

Допустимые значения

InMemoryQueue | JMSQueue | Custom

Примечание: Если используется тип JMSQueue или Custom, для интеграции Interact с IBM MQ средой выполнения Interact должен быть сервер прикладных программ с JDK 1.7. Для WebSphere и WebLogic рекомендуется использовать последнюю предоставленную версию пакета Fix Pack для JDK.

JMSQueue поддерживает только WebLogic. Нельзя использовать JMSQueue, если вы используете WebSphere Application Server.

className

Описание

Это свойство определяет полное имя класса этой реализации диспетчера. Если тип - InMemoryQueue, это значение должно быть пустым. Если тип - Custom, у этого параметра должно быть значение `com.unicacorp.interact.eventhandler.triggeredmessage.dispatchers.IBMMQDispatcher`.

classPath

Описание

Это свойство задает URL файла JAR, включающего реализацию этого диспетчера.

Если тип - Custom, у этого параметра должно быть значение `file://<домашний_каталог_Interact>/lib/interact_ibmmqdispatcher.jar;file://<домашний_каталог_Interact>/lib/com.ibm.mq.allclient.jar;file://<домашний_каталог_Interact>/lib/jms.jar`

Interact | triggeredMessage | dispatchers | <имя_диспетчера> | Parameter Data

Свойства конфигурации в этой категории задают общие параметры для конкретного диспетчера в инициированных сообщениях.

Можно выбирать из трех типов диспетчеров. InMemoryQueue - это внутренний диспетчер для Interact. Custom используется для IBM MQ. JMSQueue используется для соединения с провайдером JMS через JNDI.

имя категории

Описание

Это свойство задает имя для этого параметра. Имя должно быть уникальным среди всех параметров для этого диспетчера.

value

Описание

Это свойство определяет параметры в формате пар имя - значение, необходимые диспетчеру.

Примечание: Все параметры для инициирования сообщений регистрозависимы, и их надо вводить, как здесь показано.

Если тип - это InMemoryQueue, поддерживается следующий параметр.

- queueCapacity: Необязательное. Максимальное число предложений, которые могут ожидать в очереди своего распределения. Если это свойство задается, оно должно быть положительным целым числом. Если оно не задается или недопустимо, используется значение по умолчанию 1000.

Если тип - Custom, поддерживаются следующие параметры.

- providerUrl: <имя_хоста>:порт (зависит от регистра)
- queueManager: Имя менеджера очередей, созданного на сервере IBM MQ.
- messageQueueName: Имя очереди сообщений, созданной на сервере IBM MQ.
- enableConsumer: Для этого свойства нужно задать значение true.
- asmUserforMQAuth: Имя пользователя для ведения журнала на сервере. Оно требуется, когда сервер принудительно проводит аутентификацию. В противном случае его не надо указывать.
- authDS: Пароль, связанный с именем пользователя для ведения журнала на сервере. Он требуется, когда сервер принудительно проводит аутентификацию. В противном случае его не надо указывать.

Если тип - JMSQueue, поддерживаются следующие параметры.

- providerUrl: URL для провайдера JNDI (зависит от регистра).
- connectionFactoryJNDI: Имя JNDI фабрики соединений JMS.
- messageQueueJNDI: Имя JNDI очереди JMS, куда отправляются и откуда получаются инициированные сообщения.
- enableConsumer: Должен ли потребитель этих инициированных сообщений быть запущен в Interact. Для этого свойства надо задать значение true. Если оно не задано или недопустимо, используется значение по умолчанию false.
- initialContextFactory: Полное имя класса JNDI фабрики начального контекста. Если используется WebLogic, значением этого параметра должно быть `weblogic.jndi.WLInitialContextFactory`.

Interact | triggeredMessage | gateways | <имя_шлюза>

Свойства конфигурации в этой категории задают общие настройки для конкретного шлюза в инициированных сообщениях.

Interact не поддерживает нескольких экземпляров одного шлюза. Все файлы конфигурации шлюзов должны быть доступны со всех узлов среды выполнения Interact. В случае распределенной установки убедитесь, что файлы шлюзов расположены в положении совместного использования.

имя категории

Описание

Это свойство задает имя для этого шлюза. Оно должно быть уникально среди всех шлюзов.

className

Описание

Это свойство определяет полное имя класса этой реализации шлюза.

classPath

Описание

Это свойство задает URI файла JAR, включающего реализацию этого шлюза. Если оставить это свойство пустым, используется путь классов прикладной программы, содержащей Interact.

Например, в системе Windows, если файл JAR шлюза доступен в каталоге `C:\IBM\EMM\EmailGateway\IBM_Interact_OMO_OutboundGateway_Silverpop_1.0\lib\OMO_OutboundGateway_Silverpop.jar`, classPath должен быть таким: `file:///C:/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`. В системе Unix, если файл jar шлюза доступен в каталоге `/opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`, classPath должен быть таким: `file:///opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`.

Interact | triggeredMessage | gateways | <имя_шлюза> | Parameter Data

Свойства конфигурации в этой категории задают общие параметры для конкретного шлюза в инициированных сообщениях.

имя категории

Описание

Это свойство задает имя для этого параметра. Имя должно быть уникальным среди всех параметров для этого шлюза.

value

Описание

Это свойство в формате пар имя - значение определяет параметры, необходимые этому шлюзу. Для всех шлюзов поддерживаются следующие параметры.

Примечание: Все параметры для инициирования сообщений регистрозависимы, и их надо вводить, как здесь показано.

- validationTimeoutMillis: Срок ожидания в миллисекундах для проверки предложения через этот шлюз. Значение по умолчанию - 500.
- deliveryTimeoutMillis: Срок ожидания в миллисекундах для доставки предложения через этот шлюз. Значение по умолчанию - 1000.

Interact | triggeredMessage | channels

Свойства конфигурации в этой категории задают общие настройки для всех каналов в инициированных сообщениях.

type

Описание

Это свойство определяет корневой узел для параметров, относящихся к определенному шлюзу. По умолчанию используется встроенный селектор каналов, основанный на списке каналов, которые определены в пользовательском интерфейсе инициированных сообщений. Если выбрано поведение по умолчанию, значения className и classPath нужно оставить пустыми. В общем случае используется пользовательская реализация IChannelSelector.

Допустимые значения

Default | Custom

className

Описание

Это свойство определяет полное имя класса пользовательской реализации селектора каналов. Этот параметр обязателен, если тип - Custom.

classPath

Описание

Это свойство задает URL файла JAR, включающего пользовательскую реализацию выбора канала. Если оставить это свойство пустым, используется путь классов прикладной программы, содержащей Interact.

Interact | triggeredMessage | channels | Parameter Data

Свойства конфигурации в этой категории задают общие параметры для конкретного канала в инициированных сообщениях.

имя категории

Описание

Это свойство задает имя для этого параметра. Имя должно быть уникальным среди всех параметров для этого канала.

value

Описание

Это свойство определяет параметры в формате пар имя-значение, необходимые селектору каналов.

Если для вашего канала используются **Предпочтительные каналы покупателя**, необходимо создать

Interact | triggeredMessage | channels | <имя_канала>

Свойства конфигурации в этой категории задают общие настройки для конкретного канала в инициированных сообщениях.

имя категории

Описание

Это свойство задает имя канала, по которому отправляются предложения. Оно должно соответствовать имени, заданному во время разработки в **Campaign | partitions | <partition[N]> | Interact | outboundChannels**.

Interact | triggeredMessage | channels | <имя_канала> | <имя_обработчика>

Свойства конфигурации в этой категории задают общие настройки для конкретного обработчика в инициированных сообщениях, используемого для отправки предложений.

имя категории

Описание

Это свойство задает имя обработчика, который канал будет использовать для отправки предложений.

диспетчер

Описание

Это свойство задает имя диспетчера, по которому обработчик отправляет предложения в шлюз. Оно должно быть определено в категории **interact | triggeredMessage | dispatchers**.

шлюз

Описание

Это свойство задает имя шлюза, куда обработчик в конечном счете отправляет предложения. Оно должно быть определено в категории **interact | triggeredMessage | gateways**.

mode

Описание

Это свойство задает режим использования этого обработчика. Если выбрано Failover, этот обработчик будет использоваться только в том случае, если всем обработчикам с более высоким приоритетом в этом канале не удалось отправить предложения. Если выбрано Addon, этот обработчик используется независимо от того, успешно ли отправили предложения другие обработчики.

приоритет

Описание

Это свойство задает приоритет этого обработчика. Механизм сначала пытается использовать обработчик с самым высоким приоритетом для отправки предложений.

Допустимые значения

Любое целое число

По умолчанию

100

Interact | activityOrchestrator

Категория activityOrchestrator задает приемники и шлюзы для ваших операций входящих шлюзов Interact.

Используйте свойства конфигурации **Interact | activityOrchestrator | receivers** для конфигурирования ваших приемников Interact. Используйте свойства конфигурации **Interact | activityOrchestrator | gateways** при конфигурировании ваших шлюзов для использования в Interact.

Interact | activityOrchestrator | receivers

Категория приемников activityOrchestrator задает приемники событий для ваших операций входящих шлюзов Interact.

Category name

Описание

Имя вашего приемника.

Type

Описание

Тип приемника. Можно выбрать следующие варианты: `IBM MQ` и `Пользовательский`. `Пользовательский` требует, чтобы вы использовали реализацию `iReceiver`.

Enabled

Описание

Выберите `True` для включения приемника или `false` для его отключения.

className

Описание

Это свойство определяет полное имя класса этой реализации приемника. Это используется только, когда тип `Пользовательский`.

classPath

Описание

Это свойство задает URI файла JAR, включающего реализацию этого приемника. Если оставить это свойство пустым, используется путь классов прикладной программы, содержащей Interact. Это используется только, когда тип `Пользовательский`.

Interact | activityOrchestrator | приемники | Данные параметров

Можно добавить параметры приемника, такие как queueManager и messageQueueName для определения очереди приемника.

Interact | activityOrchestrator | gateways

Категория шлюзов activityOrchestrator задает шлюзы для ваших операций входящих шлюзов Interact.

Имя категории

Описание

Имя вашего шлюза.

className

Описание

Это свойство определяет полное имя класса этой реализации шлюза.

classPath

Описание

Это свойство задает URI файла JAR, включающего реализацию этого шлюза. Если оставить это свойство пустым, используется путь класса прикладной программы, содержащей Interact. Это используется только, когда тип Пользовательский.

Interact | activityOrchestrator | шлюзы | Данные параметров

Параметры шлюзов, такие как OMO-conf_inbound_UBX_interactEventNameMapping и OMO-conf_inbound_UBX_interactEventPayloadMapping, можно добавить в ваши файлы конфигурации шлюзов.

Interact | ETL | patternStateETL

Свойства конфигурации в этой категории определяют настройки для процесса ETL.

Имя новой категории

Описание

Предоставьте имя, уникально идентифицирующее эту конфигурацию. Обратите внимание на то, что при запуске автономного процесса ETL необходимо предоставить точно это имя. Для удобства при указании этого имени в командной строке вам может потребоваться исключить имя, содержащее пробелы или знаки препинания, то есть использовать такое имя как ETLProfile1.

runOnceADay

Описание

Определяет, должен ли автономный процесс ETL из этой конфигурации запускаться один раз каждый день. Допустимые ответы - **Yes** (Да) или **No** (Нет). Если ответить здесь **No**, расписание запусков для процесса будет определяться **processSleepIntervalInMinutes**.

preferredStartTime

Описание

Предпочтительное время, когда должен начинаться автономный процесс ETL. Укажите время в формате ЧЧ:ММ:СС AM/PM, например, 01:00:00 AM.

preferredEndTime

Описание

Предпочтительное время, когда должен завершаться автономный процесс ETL. Укажите время в формате ЧЧ:ММ:СС AM/PM, например, 08:00:00 AM.

processSleepIntervalInMinutes

Описание

Если вы не сконфигурировали автономный процесс ETL для однократного ежедневного запуска (задается свойством **runOnceADay**), это свойство задает интервал между запусками процесса ETL. Например, если здесь задать 15, автономный процесс ETL будет выжидать 15 минут после остановки выполнения, прежде чем начаться снова.

maxJDBCInsertBatchSize

Описание

Максимальное число записей в пакете JDBC перед принятием запроса. По умолчанию для этого свойства задается значение 5000. Обратите внимание на то, что это не максимальное число записей, обрабатываемых ETL за одну итерацию. При каждой итерации ETL обрабатывает все доступные записи из таблицы UACI_EVENTPATTERNSTATE. Однако все эти записи разбиваются на чанки размером **maxJDBCInsertSize**.

maxJDBCFetchBatchSize

Описание

Максимальное число записей в пакете JDBC перед его получением из промежуточной базы данных.

Вам может потребоваться увеличить это значение, чтобы настроить производительность ETL.

communicationPort

Описание

Сетевой порт, через который автономный процесс ETL принимает требование остановки. При обычных условиях не должно быть причин для изменения этого значения по умолчанию.

queueLength

Описание

Значение, используемое для настройки производительности. Собрания данных состояний паттернов, собранных и преобразованных в объекты, которые добавляются в очередь для обработки и записи в базу данных. Это свойство управляет размером данной очереди.

completionNotificationScript

Описание

Задает абсолютный путь сценария для запуска после завершения процесса ETL. Если указать сценарий, в сценарий уведомления о завершении передается три параметра: время начала, время завершения и полное число обработанных записей паттернов событий. Время начала и время завершения - это числовые значения, соответствующие числу миллисекунд, прошедшему с 1970 г.

Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS

Свойства конфигурации в этой категории определяют параметры для DS времени выполнения ETL.

type

Описание

Список поддерживаемых типов баз данных для источника данных, который вы определяете.

dsname

Описание

Имя JNDI источника данных. Это имя должно использоваться также в конфигурации источника данных пользователя, чтобы гарантировать, что у пользователя есть доступ к источникам данных назначения и времени выполнения.

драйвер

Описание

Имя используемого драйвера JDBC, такое как одно из следующих:

Oracle: `oracle.jdbc.OracleDriver`

Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2: `com.ibm.db2.jcc.DB2Driver`

serverURL

Описание

URL источника данных, такое как одно из следующих:

Oracle:
`jdbc:oracle:thin:@<хост_вашей_БД>:<порт_вашей_БД>:<имя_службы_вашей_БД>`

Microsoft SQL Server: `jdbc:sqlserver://`
`<хост_вашей_БД>:<порт_вашей_БД>;<имя_вашей_БД>`

IBM DB2: `jdbc:db2://<хост_вашей_БД>:<порт_вашей_БД>/<имя_вашей_БД>`

connectionpoolSize

Описание

Значение, указывающее размер пула соединений, заданное для настройки производительности. Данные состояния паттерна читаются и преобразуются одновременно в зависимости от доступных соединений с базами данных. Увеличение размера пула позволяет использовать больше соединений с базами данных одновременно в зависимости от ограничений памяти и возможностей чтения/записи базы данных. Например, если задано значение 4, одновременно будут выполняться четыре задания. Если у вас большой объем данных, может потребоваться увеличить это значение до 10 или до 20 при условии доступности достаточной памяти и производительности базы данных.

ID пользователя

Описание

Имя схемы базы данных, с которой соединяется эта конфигурация.

connectionRetryPeriod

Описание

Свойство `ConnectionRetryPeriod` задает время (в секундах), в течение которого Interact автоматически повторяет требование установления соединения с базой данных при ошибке. Interact автоматически попытается восстановить соединение с базой данных в течение этого времени, прежде чем сообщить об ошибке или сбое базы данных. Если задано значение 0, Interact повторяет попытки до бесконечности; если задано значение -1, никакие повторные попытки предприниматься не будут.

connectionRetryDelay

Описание

Свойство `ConnectionRetryDelay` задает время (в секундах), в течение которого Interact попытается повторно соединиться с базой данных после сбоя. Если значение равно -1, то повторная попытка не предпринимается.

Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS

Свойства конфигурации в этой категории определяют параметры для DS назначения ETL.

type

Описание

Список поддерживаемых типов баз данных для источника данных, который вы определяете.

dsname

Описание

Имя JNDI источника данных. Это имя должно использоваться также в конфигурации источника данных пользователя, чтобы гарантировать, что у пользователя есть доступ к источникам данных назначения и времени выполнения.

драйвер

Описание

Имя используемого драйвера JDBC, такое как одно из следующих:

Oracle: `oracle.jdbc.OracleDriver`

Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2: `com.ibm.db2.jcc.DB2Driver`

serverURL

Описание

URL источника данных, такое как одно из следующих:

Oracle:
`jdbc:oracle:thin:@<хост_вашей_БД>:<порт_вашей_БД>:<имя_службы_вашей_БД>`

Microsoft SQL Server: `jdbc:sqlserver://`
`<хост_вашей_БД>:<порт_вашей_БД>;<имя_вашей_БД>`

IBM DB2: jdbc:db2://<хост_вашей_БД>:<порт_вашей_БД>/<имя_вашей_БД>

connectionpoolSize

Описание

Значение, указывающее размер пула соединений, заданное для настройки производительности. Данные состояния паттерна читаются и преобразуются одновременно в зависимости от доступных соединений с базами данных. Увеличение размера пула позволяет использовать больше соединений с базами данных одновременно в зависимости от ограничений памяти и возможностей чтения/записи базы данных. Например, если задано значение 4, одновременно будут выполняться четыре задания. Если у вас большой объем данных, может потребоваться увеличить это значение до 10 или до 20 при условии доступности достаточной памяти и производительности базы данных.

ID пользователя

Описание

Имя схемы базы данных, с которой соединяется эта конфигурация.

connectionRetryPeriod

Описание

Свойство ConnectionRetryPeriod задает время (в секундах), в течение которого Interact автоматически повторяет требование установления соединения с базой данных при ошибке. Interact автоматически попытается восстановить соединение с базой данных в течение этого времени, прежде чем сообщить об ошибке или сбое базы данных. Если задано значение 0, Interact повторяет попытки до бесконечности; если задано значение -1, никакие повторные попытки предприниматься не будут.

connectionRetryDelay

Описание

Свойство ConnectionRetryDelay задает время (в секундах), в течение которого Interact попытается повторно соединиться с базой данных после сбоя. Если значение равно -1, то повторная попытка не предпринимается.

Interact | ETL | patternStateETL | <patternStateETLName> | Report

Свойства конфигурации в этой категории определяют настройки для процесса агрегации отчетов ETL.

включить

Описание

Включает или выключает интеграцию отчета с ETL. По умолчанию для этого свойства задано значение disable (отключено).

Если задано значение disable, это свойство отключает обновления в таблице UARI_DELTA_PATTERNS. Это не отключает возможность составления отчетов полностью.

Примечание: Чтобы отключить интеграцию отчетов с ETL, надо также изменить триггер TR_AGGREGATE_DELTA_PATTERNS для отключения промежуточной таблицы UACI_ETLPATTERNSTATERUN.

retryAttemptsIfAggregationRunning

Описание

Сколько раз ETL пытается проверить, выполнена ли агрегация отчета, если задан флаг блокировки. По умолчанию для этого свойства задано значение 3.

sleepBeforeRetryDurationInMinutes

Описание

Время сна (в минутах) между последовательными попытками. По умолчанию для этого свойства задано значение 5 (минут).

aggregationRunningCheckSql

Описание

Это свойство позволяет задать пользовательский SQL, который можно запустить, чтобы увидеть, задан ли флаг блокировки агрегации отчета. По умолчанию это свойство пусто.

Когда это свойство не задано, ETL выполняет следующий код SQL для получения флага блокировки.

```
select count(1) AS ACTIVERUNS from uari_pattern_lock where islock='Y'
=> Если ACTIVERUNS > 0, блокировка задана
```

aggregationRunningCheck

Описание

Включите или выключите переключатель, чтобы задать, выполняется ли агрегация отчета до запуска ETL. По умолчанию для этого свойства задано значение Включено.

Глава 14. Свойства конфигурации среды разработки Interact

В этом разделе описываются все свойства конфигурации для среды разработки Interact.

Campaign | partitions | partition[n] | reports

Свойство **Campaign | partitions | partition[n] | reports** определяет различные типы папок для отчетов.

offerAnalysisTabCachedFolder

Описание

Свойство `offerAnalysisTabCachedFolder` указывает расположение папки, содержащей спецификацию для пакетных (расширенных) отчетов о предложениях, перечисленных на вкладке Анализ в панели навигации. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

Описание

Свойство `segmentAnalysisTabOnDemandFolder` указывает расположение папки, содержащей отчеты о сегментах, перечисленные на вкладке Анализ для сегмента. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

Описание

Свойство `offerAnalysisTabOnDemandFolder` указывает расположение папки, содержащей отчеты о предложениях, перечисленные на вкладке Анализ для предложения. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

segmentAnalysisTabCachedFolder

Описание

Свойство `segmentAnalysisTabCachedFolder` указывает расположение папки, содержащей спецификацию для пакетных (расширенных) отчетов о сегментах, перечисленных на вкладке Анализ в панели навигации. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']
```

analysisSectionFolder

Описание

Свойство `analysisSectionFolder` задает расположение корневой папки, в которой хранятся спецификации отчетов. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign']
```

campaignAnalysisTabOnDemandFolder

Описание

Свойство `campaignAnalysisTabOnDemandFolder` указывает расположение папки, содержащей отчеты о кампаниях, перечисленные на вкладке Анализ для кампании. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']
```

campaignAnalysisTabCachedFolder

Описание

Свойство `campaignAnalysisTabCachedFolder` указывает расположение папки, содержащей спецификацию для пакетных (расширенных) отчетов о кампаниях, перечисленных на вкладке Анализ в панели навигации. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']/folder[@name='cached']
```

campaignAnalysisTabEmessageOnDemandFolder

Описание

Свойство `campaignAnalysisTabEmessageOnDemandFolder` указывает расположение папки, содержащей отчеты eMessage, перечисленные на вкладке Анализ для кампании. Путь указывается с использованием формы записи XPath.

Значение по умолчанию

```
/content/folder[@name='Affinium Campaign']/folder[@name='eMessage
Reports']
```

campaignAnalysisTabInteractOnDemandFolder

Описание

Строка папки сервера отчетов для отчетов Interact.

Значение по умолчанию

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

Доступность

Это свойство применимо, только если вы устанавливаете Interact.

interactiveChannelAnalysisTabOnDemandFolder

Описание

Строка папки сервера отчетов для отчетов на вкладке анализа Интерактивный канал.

Значение по умолчанию

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

Доступность

Это свойство применимо, только если вы устанавливаете Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking

Эти свойства конфигурации задают параметры для модуля хронологии контактов и ответов Interact.

isEnabled

Описание

Если задано значение `yes`, включает модуль хронологии контактов и ответов Interact, который копирует хронологию контактов и ответов Interact из промежуточных таблиц в среде выполнения Interact в таблицы хронологии контактов и ответов Campaign. Для свойства `interactInstalled` также должно быть задано значение `yes`.

Значение по умолчанию

нет

Допустимые значения

yes | no

Доступность

Это свойство применимо, только если вы установили Interact.

runOnceADay

Описание

Указывает, нужно ли запускать ETL хронологии контактов и ответов раз в день. Если задать для этого свойства значение `Yes`, ETL запустится в запланированный период, заданный параметрами `preferredStartTime` и `preferredEndTime`.

Если для выполнения ETL потребуется более 24 часов, в связи с чем будет пропущено начальное время на следующий день, этот день будет пропущен, и ETL запустится в запланированное время через сутки. Например, если сконфигурирован запуск ETL с 1 часа ночи до 3 часов ночи, а процесс запустится в 1 час ночи в понедельник и завершится в 2 часа ночи во вторник,

следующий запуск, который был первоначально запланирован на 1 час ночи во вторник, будет пропущен, а следующий запуск ETL начнется в 1 час ночи в среду.

При планировании ETL не учитывается переход на летнее/зимнее время. Например, если запуск ETL запланирован в период с 1 до 3 часов ночи, то при переходе на летнее/зимнее время он может выполняться с 12 часов ночи или с 2 часов ночи.

Значение по умолчанию

Нет

Доступность

Это свойство применимо, только если вы установили Interact.

processSleepIntervalInMinutes

Описание

Время в минутах, которое модуль хронологии контактов и ответов Interact выжидает между копированием данных из промежуточных таблиц среды выполненияInteract в таблицы хронологии контактов и ответов Campaign.

Значение по умолчанию

60

Допустимые значения

Любое целочисленное значение, больше нуля.

Доступность

Это свойство применимо, только если вы установили Interact.

preferredStartTime

Описание

Предпочтительное время для начала ежедневного процесса ETL. Это свойство, при использовании в сочетании со свойством preferredEndTime, задает предпочтительный интервал времени, в течение которого должен запускаться процесс ETL. ETL запустится в указанный интервал времени и обработает число записей, заданных параметром `maxJDBCFetchBatchSize`. Формат - ЧЧ:мм:сс AM или PM при использовании 12-часового времени.

Значение по умолчанию

12:00:00 AM

Доступность

Это свойство применимо, только если вы установили Interact.

preferredEndTime

Описание

Предпочтительное время для завершения ежедневного процесса ETL. Это свойство, при использовании в сочетании со свойством preferredStartTime, задает предпочтительный интервал времени, в течение которого должен запускаться процесс ETL. ETL запустится в указанный интервал времени и обработает число записей, заданных параметром `maxJDBCFetchBatchSize`. Формат - ЧЧ:мм:сс AM или PM при использовании 12-часового времени.

Значение по умолчанию

2:00:00 AM

Доступность

Это свойство применимо, только если вы установили Interact.

purgeOrphanResponseThresholdInMinutes

Описание

Время в минутах, которое модуль хронологии контактов и ответов Interact выжидает между удалением ответов без соответствующих контактов. Это позволяет предотвратить запись в журнал ответов без записи контактов.

Значение по умолчанию

180

Допустимые значения

Любое целочисленное значение, больше нуля.

Доступность

Это свойство применимо, только если вы установили Interact.

maxJDBCInsertBatchSize

Описание

Максимальное число записей в пакете JDBC перед принятием запроса. Это не максимальное число записей, которые обрабатывает модуль хронологии контактов и ответов Interact за один проход. При каждом проходе модуль хронологии контактов и ответов Interact обрабатывает все доступные записи из промежуточных таблиц. Однако все эти записи разбиваются на чанки размером `maxJDBCInsertSize`.

Значение по умолчанию

1000

Допустимые значения

Любое целочисленное значение, больше нуля.

Доступность

Это свойство применимо, только если вы установили Interact.

maxJDBCFetchBatchSize

Описание

Максимальное число записей в пакете JDBC перед его получением из промежуточной базы данных. Вам может потребоваться увеличить это значение, чтобы настроить эффективность модуля хронологии контактов и ответов.

Например, чтобы обработать 2,5 миллиона записей хронологии контактов в сутки, нужно задать для `maxJDBCFetchBatchSize` число, превышающее 2,5 миллиона, чтобы были обработаны записи за сутки.

Можно задать для `maxJDBCFetchChunkSize` и `maxJDBCInsertBatchSize` меньшие значения (в данном примере это может быть 50000 и 10000,

соответственно). Некоторые записи за следующий день также могут быть обработаны, он они потом будут сохранены до следующего дня.

Значение по умолчанию

1000

Допустимые значения

Любое целочисленное значение, больше нуля.

maxJDBCFetchChunkSize

Описание

Максимальный размер чанка JDBC при чтении данных во время ETL (extract, transform, load - извлечение, перенос, загрузка). В некоторых случаях размер чанка, превышающий размер вставки, позволяет повысить скорость процесса ETL.

Значение по умолчанию

1000

Допустимые значения

Любое целочисленное значение, больше нуля.

deleteProcessedRecords

Описание

Указывает, нужно ли сохранять записи хронологии контактов и ответов после их обработки.

Значение по умолчанию

Да

completionNotificationScript

Описание

Задает абсолютный путь сценария для запуска после завершения ETL. Если задать сценарий, в сценарий уведомления о выполнении передаются пять аргументов: время начала, время завершения, полное число обработанных записей CH, полное число обработанных записей RH и состояние. Время начала и время завершения - это числовые значения, соответствующие числу миллисекунд, прошедшему с 1970 г. Аргумент состояния указывает, было ли задание ETL завершено успешно, или произошел сбой. 0 означает успешное завершение задания ETL. 1 указывает на сбой и наличие некоторых ошибок в задании ETL.

Значение по умолчанию

Нет

fetchSize

Описание

Позволяет задать параметр JDBC fetchSize при получении записей из промежуточной таблицы.

В базах данных Oracle задайте для этого параметра число записей, которые JDBC должен получить при каждом цикле обхода сети. При больших пакетах

(100 КБ и больше) попробуйте использовать значение, равное 10000. Проследите за тем, чтобы не использовать здесь слишком большое значение, так как это влияет на использование памяти, и выигрыш может стать пренебрежимо малым (если не превратится в недостаток).

Значение по умолчанию

Нет

daysBackInHistoryToLookupContact

Описание

Ограничивает записи, среди которых выполняется поиск при запросах хронологии ответов, только записями за указанное число дней в прошлом. Для баз данных с большим числом записей хронологии ответов это может сократить время обработки для запросов, ограничив период поиска заданным числом дней.

Значение по умолчанию 0 обозначает, что поиск выполняется среди всех записей.

Значение по умолчанию

0 (ноль)

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

Эти свойства конфигурации задают источник данных для модуля хронологии контактов и ответов Interact.

jndiName

Описание

Используйте свойство `systemTablesDataSource`, чтобы задать источник данных Java Naming and Directory Interface (JNDI), заданный на сервере прикладных программ (Websphere или WebLogic) для таблиц среды выполнения Interact.

База данных среды выполнения Interact - это база данных, заполненная сценариями dll `aci_runtime` и `aci_populate_runtime`, и, например, она соде ржит следующие таблицы (помимо прочих): `UACI_CHOfferAttrib` и `UACI_DefaultedStat`.

Значение по умолчанию

Значения по умолчанию нет.

Доступность

Это свойство применимо, только если вы установили Interact.

databaseType

Описание

Тип базы данных для источника данных среды выполнения Interact.

Значение по умолчанию

SQLServer

Допустимые значения

SQLServer | Oracle | DB2

Доступность

Это свойство применимо, только если вы установили Interact.

schemaName

Описание

Имя схемы, содержащей промежуточные таблицы модуля хронологии контактов и ответов. Оно должно совпадать с таблицами среды выполнения.

Задавать схему вы не должны.

Значение по умолчанию

Значения по умолчанию нет.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

Эти свойства конфигурации задают тип контактов из кампании, которые отображаются в 'contact' для отчетов или обучения.

установлен контакт

Описание

Значение, заданное для столбца `ContactStatusID` в таблице `UA_DtlContactHist` в системных таблицах Campaign для контакта предложения. Значение должно быть действительной записью в таблице `UA_ContactStatus`. Подробную информацию о добавлении типов контактов смотрите в публикации *Campaign: Руководство администратора*.

Значение по умолчанию

2

Допустимые значения

Целочисленное значение, больше нуля.

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

Эти свойства конфигурации задают ответы для принятия или отклонения для отчетов и обучения.

принять

Описание

Значение, заданное для столбца `ResponseTypeID` в таблице `UA_ResponseHistory` в системных таблицах Campaign для принятого предложения. Значение должно быть действительной записью в таблице `UA_UsrResponseType`. Вы должны задать для столбца `CountsAsResponse` значение 1, соответствующее ответу.

Подробную информацию о добавлении типов ответов смотрите в публикации *Campaign: Руководство администратора*.

Значение по умолчанию

3

Допустимые значения

Целочисленное значение, больше нуля.

Доступность

Это свойство применимо, только если вы установили Interact.

отклонить

Описание

Значение, заданное для столбца `ResponseTypeID` в таблице `UA_ResponseHistory` в системных таблицах Campaign для отклоненного предложения. Значение должно быть действительной записью в таблице `UA_UsrResponseType`. Вы должны задать для столбца `CountsAsResponse` значение `2`, соответствующее отклонению. Подробную информацию о добавлении типов ответов смотрите в публикации *Campaign: Руководство администратора*.

Значение по умолчанию

8

Допустимые значения

Любое целочисленное значение, больше нуля.

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | report

Эти свойства конфигурации задают имена отчетов при интеграции с Cognos.

interactiveCellPerformanceByOfferReportName

Описание

Имя отчета Эффективность интерактивной ячейки по предложениям. Это имя должно соответствовать имен этого отчета на сервере Cognos.

Значение по умолчанию

Эффективность интерактивных ячеек по предложениям

treatmentRuleInventoryReportName

Описание

Имя отчета Перечень правил процедур. Это имя должно соответствовать имен этого отчета на сервере Cognos.

Значение по умолчанию

Перечень правил процедуры канала

deploymentHistoryReportName

Описание

Имя отчета о хронологии внедрения. Это имя должно соответствовать имен этого отчета на сервере Cognos.

Значение по умолчанию

Хронология внедрения канала

Campaign | partitions | partition[n] | Interact | learning

Эти свойства конфигурации позволяют настроить встроенный модуль обучения.

confidenceLevel

Описание

Процент, указывающий, насколько должна быть скрыта утилита обучения перед переключением из исследования в использование. Значение, равное 0, эффективно выключает исследование.

Это свойство применимо, если для свойства `Interact > offerserving > optimizationType` для среды выполнения Interact задано только значение `BuiltInLearning`.

Значение по умолчанию

95

Допустимые значения

Целое число от 0 до 95, кратное 5 или 99.

validateonDeployment

Описание

Если задать значение `No`, Interact не проверяет модуль обучения при внедрении. Если задать значение `yes`, Interact проверяет модуль обучения при внедрении.

Значение по умолчанию

`No`

Допустимые значения

Yes | No

maxAttributeNames

Описание

Максимальное число атрибутов обучения, которое отслеживает утилита обучения Interact.

Это свойство применимо, если для свойства `Interact > offerserving > optimizationType` для среды выполнения Interact задано только значение `BuiltInLearning`.

Значение по умолчанию

`10`

Допустимые значения

Любое целочисленное значение.

maxAttributeValues

Описание

Максимальное число значений, которые модуль обучения Interact отслеживает для каждого атрибута обучения.

Это свойство применимо, если для свойства `Interact > offerserving > optimizationType` для среды выполнения Interact задано только значение `BuiltInLearning`.

Значение по умолчанию

5

otherAttributeValue

Описание

Имя по умолчанию для атрибута, используемого для представления всех значений атрибутов, кроме `maxAttributeValues`.

Это свойство применимо, если для свойства `Interact > offerserving > optimizationType` для среды выполнения Interact задано только значение `BuiltInLearning`.

Значение по умолчанию

`Другой`

Допустимые значения

Строка или число.

Пример

Если для maxAttributeValues задано значение 3, а для otherAttributeValue задано другое значение, модуль обучения будет отслеживать первые три значения. Все остальные значения назначается другая категория. Например, если вы отслеживаете атрибут цвета волос посетителя, и у первых пяти посетителей были черные, коричневые, светлые, рыжие и седые волосы, утилита обучения отследит следующие цвета волос: черные, коричневые и светлые. Цвета рыжие и седые будут сгруппированы в другой категории `otherAttributeValue`.

percentRandomSelection

Описание

Процент времени, когда модуль обучения представляет случайное предложение. Например, задание для percentRandomSelection значения 5 означает, что 5% времени (5 из каждых 100 рекомендаций) модуль обучения будет представлять случайное предложение независимо от оценки. Включение `percentRandomSelection` переопределяет свойство конфигурации `offerTieBreakMethod`. При включении `percentRandomSelection` это свойство задается независимо от включения или отключения обучения и от использования встроенного или внешнего обучения.

Значение по умолчанию

5

Допустимые значения

Любое целое число от 0 (что отключает возможность `percentRandomSelection`) до 100.

recencyWeightingFactor

Описание

Десятичное представление процента набора данных, заданного параметром `recencyWeightingPeriod`. Например, значение по умолчанию, равное 0,15, означает, что 15% данных, используемых утилитой обучения, берутся из `recencyWeightingPeriod`.

Это свойство применимо, если для свойства `Interact > offerserving > optimizationType` для среды выполнения Interact задано только значение `BuiltInLearning`.

Значение по умолчанию

0,15

Допустимые значения

Десятичное значение меньше 1.

recencyWeightingPeriod

Описание

Размер данных в часах, обеспечивающий процент веса `recencyWeightingFactor` от модуля обучения. Например, значение по умолчанию, равное 120, означает, что количество данных `recencyWeightingFactor`, используемое модулем обучения, взято за последние 120 часов.

Это свойство применимо, только если для параметра `optimizationType` задано значение `builtInLearning`.

Значение по умолчанию

120

minPresentCountThreshold

Описание

Минимальное число раз, которое должно быть представлено предложение, прежде чем данные, используемые в вычислениях, и модуль обучения перейдут в режим исследования.

Значение по умолчанию

0

Допустимые значения

Целочисленное значение, больше или равное нулю.

enablePruning

Описание

Если задано значение `Yes`, модуль обучения Interact алгоритмически определить, является ли атрибут обучения (стандартный или динамический) не прогностическим. Если атрибут обучения не является прогностическим,

модуль обучения не будет рассматривать этот атрибут при определении веса предложения. Так будет продолжаться, пока модуль обучения собирает данные обучения.

Если задано значение `No`, модуль обучения всегда будет использовать все атрибуты обучения. Если не отбрасывать непрогностические атрибуты, модуль обучения может быть не таким точным, как это возможно.

Значение по умолчанию

Да

Допустимые значения

Yes | No

Campaign | partitions | partition[n] | Interact | learning | learningAttributes | [learningAttribute]

Эти свойства конфигурации задают атрибуты обучения.

attributeName

Описание

Каждое значение `attributeName` - это имя атрибута посетителя, которое должен отслеживать модуль обучения. Это значение должно совпадать с именем пары имя-значение в данных сеанса.

Это свойство применимо, если для свойства `Interact > offerserving > optimizationType` для среды выполнения Interact задано только значение `BuiltInLearning`.

Значение по умолчанию

Значения по умолчанию нет.

Campaign | partitions | partition[n] | Interact | deployment

Эти свойства конфигурации задают параметры внедрения.

chunkSize

Описание

максимальный размер фрагментации в КБ для каждого пакета внедрения Interact.

Значение по умолчанию

500

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup]

Эти свойства конфигурации задают параметры группы серверов.

serverGroupName

Описание

Имя группы серверов среды выполнения Interact. Это имя, которое появляется на вкладке сводки для интерактивного канала.

Значение по умолчанию

Значения по умолчанию нет.

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

Эти свойства конфигурации задают серверы среды выполнения Interact.

instanceURL

Описание

URL сервера среды выполнения Interact. В группе сервером может быть несколько серверов среды выполнения Interact, однако каждый сервер должен быть создан в новой категории.

Значение по умолчанию

Значения по умолчанию нет.

Пример

`http://`*`сервер`*`:`*`порт`*`/interact`

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | flowchart

Эти свойства конфигурации задают среду выполнения Interact, используемую для тест-запусков интерактивных потоковых диаграмм.

serverGroup

Описание

Имя группы серверов Interact, используемое компонентом Campaign для выполнения тест-запуска. Это имя должно совпадать с именем категории, созданным вами в `serverGroups`.

Значение по умолчанию

Значения по умолчанию нет.

Доступность

Это свойство применимо, только если вы установили Interact.

dataSource

Описание

Используйте свойство `dataSource`, чтобы указать для Campaign физический источник данных, который следует использовать при выполнении тест-запусков интерактивных потоковых диаграмм. Это свойство должно соответствовать источнику данных, заданному свойством `Campaign >`

partitions > partitionN > dataSources для источника данных тест-запусков, заданного во время разработки Interact.

Значение по умолчанию

Значения по умолчанию нет.

Доступность

Это свойство применимо, только если вы установили Interact.

eventPatternPrefix

Описание

Свойство eventPatternPrefix - это строчное значение, которое добавляется спереди к именам паттернов событий, чтобы их можно было использовать в выражениях процессов выбора или решения с интерактивными потоковыми диаграммами.

Заметим, что если вы изменяете это значение, надо внедрить глобальные изменения в интерактивном канале, чтобы измененная конфигурация вступила в силу.

Значение по умолчанию

EventPattern

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

Эти свойства конфигурации задают код ячейки по умолчанию для таблицы предложений по умолчанию. Вы должны сконфигурировать эти свойства, только если вы задаете глобальные назначения предложений.

DefaultCellCode

Описание

Код ячейки по умолчанию, который используется компонентом Interact, если вы не зададите код ячейки в таблице предложений по умолчанию.

Значение по умолчанию

Значения по умолчанию нет.

Допустимые значения

Строка, соответствующая формату кода ячейки, заданному в Campaign

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

Эти свойства конфигурации задают код ячейки по умолчанию для таблицы offersBySQL. Вы должны сконфигурировать эти свойства, только если вы используете запросы SQL для получения нужного набора предложений-кандидатов.

DefaultCellCode

Описание

Код ячейки по умолчанию, который Interact использует для любого предложения в таблицах OffersBySQL и у которого в столбце кода ячейки находится нулевое значение (или если столбец кодов ячеек вообще отсутствует). Значение должно быть действительным кодом ячейки.

Значение по умолчанию

Значения по умолчанию нет.

Допустимые значения

Строка, соответствующая формату кода ячейки, заданному в Campaign

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

Эти свойства конфигурации задают код ячейки по умолчанию для таблицы переопределения оценок. Вы должны сконфигурировать эти свойства, только если вы задаете индивидуальные назначения предложений.

DefaultCellCode

Описание

Код ячейки по умолчанию, который используется компонентом Interact, если вы не зададите код ячейки в таблице переопределения оценок.

Значение по умолчанию

Значения по умолчанию нет.

Допустимые значения

Строка, соответствующая формату кода ячейки, заданному в Campaign

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | server | internal

Свойства в этой категории задают параметры интеграции и пределы внутренних ID для выбранного раздела Campaign. Если в вашей установке Campaign есть несколько разделов, задайте эти свойства для каждого раздела, на который вы хотите повлиять.

internalIdLowerLimit

Категория конфигурации

`Campaign|partitions|partition[n]|server|internal`

Описание

Свойства `internalIdUpperLimit` и `internalIdLowerLimit` ограничивают внутренние ID Campaign заданным диапазоном. Учтите, что значения указываются включительно, то есть, Campaign может использовать и нижнюю, и верхнюю границу диапазона.

Значение по умолчанию

0 (ноль)

internalIdUpperLimit

Категория конфигурации

Campaign|partitions|partition[n]|server|internal

Описание

Свойства internalIdUpperLimit и internalIdLowerLimit ограничивают внутренние ID Campaign заданным диапазоном. Значения указываются включительно, то есть, Campaign может использовать и нижнюю, и верхнюю границу диапазона. Если установлено Distributed Marketing, то задайте значение 2147483647.

Значение по умолчанию

4294967295

eMessageInstalled

Категория конфигурации

Campaign|partitions|partition[n]|server|internal

Описание

Указывает, что компонент eMessage установлен. Если выбрано Yes, то функции eMessage будут доступны в интерфейсе Campaign.

Программа установки IBM задает для этого свойства значение Yes для раздела по умолчанию в вашей установке eMessage. Для дополнительных разделов, в которых вы установили eMessage, это свойство нужно сконфигурировать вручную.

Значение по умолчанию

No

Допустимые значения

Yes | No

interactInstalled

Категория конфигурации

Campaign|partitions|partition[n]|server|internal

Описание

После установки среды разработки Interact для этого свойства конфигурации нужно задать значение Yes, чтобы включить среду разработки Interact в Campaign.

Если компонент Interact не установлен, то задайте значение No. Если для этого свойства задано No, то меню и опции Interact не удаляются из интерфейса пользователя. Чтобы удалить меню и опции, нужно вручную отменить регистрацию Interact при помощи утилиты configTool.

Значение по умолчанию

Нет

Допустимые значения

Yes | No

Доступность

Это свойство применимо, только если вы установили Interact.

MO_UC_integration

Категория конфигурации

`Campaign|partitions|partition[n]|server|internal`

Описание

Разрешает интеграцию с Marketing Operations для этого раздела, если интеграция разрешена в параметрах конфигурации **Platform**. Дополнительную информацию смотрите в публикации *IBM Marketing Operations и Campaign: Руководство по интеграции*.

Значение по умолчанию

Нет

Допустимые значения

Yes | No

MO_UC_BottomUpTargetCells

Категория конфигурации

`Campaign|partitions|partition[n]|server|internal`

Описание

Разрешает ячейки снизу вверх для электронных таблиц ячеек назначения в этом разделе, если разрешено **MO_UC_integration**. Если задано `Yes`, то ячейки назначения, заданные сверху вниз и снизу вверх, будут видны, но ячейки назначения, заданные снизу вверх, будут доступны только для чтения. Дополнительную информацию смотрите в публикации *IBM Marketing Operations и Campaign: Руководство по интеграции*.

Значение по умолчанию

Нет

Допустимые значения

Yes | No

Legacy_campaigns

Категория конфигурации

`Campaign|partitions|partition[n]|server|internal`

Описание

Разрешает для этого раздела доступ к кампаниям, созданным до интеграции Marketing Operations и Campaign. Применяется, только если для **MO_UC_integration** задано `Yes`. В число унаследованных кампаний входят также кампании, созданные в Campaign 7.x и связанные с проектами Plan 7.x. Дополнительную информацию смотрите в публикации *IBM Marketing Operations и Campaign: Руководство по интеграции*.

Значение по умолчанию

Нет

Допустимые значения

Yes | No

IBM Marketing Operations - Интеграция предложения

Категория конфигурации

Campaign|partitions|partition[n]|server|internal

Описание

Дает возможность использовать Marketing Operations для выполнения задач управления жизненным циклом предложения в этом разделе, если для него разрешено **MO_UC_integration**. В параметрах интеграции **Platform** должна быть разрешена интеграция предложений. Дополнительную информацию смотрите в публикации *IBM Marketing Operations и Campaign: Руководство по интеграции*.

Значение по умолчанию

Нет

Допустимые значения

Yes | No

UC_CM_integration

Категория конфигурации

Campaign|partitions|partition[n]|server|internal

Описание

Включает интеграцию онлайн-сегмента цифровой аналитики для раздела Campaign. Если вы устанавливаете это значение на Да, то технологический блок-бокс Выбор в потоковой диаграмме предоставляет возможность выбрать **Сегменты цифровой аналитики** в качестве ввода. Чтобы сконфигурировать интеграцию цифровой аналитики для каждого раздела, выберите **Параметры > Конфигурация > Campaign | partitions | partition[n] | Coremetrics**.

Значение по умолчанию

Нет

Допустимые значения

Yes | No

numRowsReadToParseDelimitedFile

Категория конфигурации

Campaign|partitions|partition[n]|server|internal

Описание

Это свойство используется при отображении файла с разделителями как пользовательской таблицы. Оно также используется в технологическом блок-боксе Оценки при импорте выходного файла оценки из IBM SPSS Modeler Advantage Enterprise Marketing Management Edition. Чтобы импортировать или отобразить разграниченный файл, Campaign должен проанализировать файл для идентификации столбцов, типов данных (типов полей) и ширины столбцов (длины полей).

Значение по умолчанию 100 означает, что Campaign исследует первые 50 и последние 50 записей строки в разграниченном файле. Campaign затем выделяет длину поля на основе самого большого значения в записях. В большинстве случаев значение по умолчанию достаточно для определения длин полей. При этом в очень больших разграниченных файлах более

позднее поле может превысить предполагаемую длину, рассчитанную Campaign, что может вызвать ошибку во время выполнения потоковой диаграммы. Следовательно, при отображении очень большого файла это значение можно увеличить дляCampaign исследования большего количества строк. Например, значение 200 приводит к Campaign исследованию первых 100 записей строки и последних 100 записей строки файла.

При значении 0 исследуется весь файл. Как правило, это необходимо только при импорте или отображении файлов с изменяющимися значениями ширины данных полей, которые не могут быть определены путем считывания первых и последних нескольких строк. Считывание всего файла для крайне больших файлов может увеличить необходимое время обработки для табличного отображения и выполнения технологического блок-бокса Оценки.

Значение по умолчанию

100

Допустимые значения

0 (все строки) или любое положительное целое число

Campaign | monitoring

Свойства в этой категории указывают, включена ли функция Operational Monitoring, URL сервера Operational Monitoring, а также порядок кэширования. Operational Monitoring появится и позволит вам управлять активными потоковыми диаграммами.

cacheCleanupInterval

Описание

Свойство `cacheCleanupInterval` задает интервал (в секундах) между автоматическими очистками кэша состояния потоковых диаграмм.

Это свойство недоступно в более ранних версиях Campaign, чем 7.0.

Значение по умолчанию

`600` (10 минут)

cacheRunCompleteTime

Описание

Свойство `cacheRunCompleteTime` указывает время в минутах, в течение которого завершенные запуски кэшируются и появляются на странице Мониторинг.

Это свойство недоступно в более ранних версиях Campaign, чем 7.0.

Значение по умолчанию

`4320`

monitorEnabled

Описание

Свойство `monitorEnabled` указывает, включен ли монитор.

Это свойство недоступно в более ранних версиях Campaign, чем 7.0.

Значение по умолчанию

FALSE

Допустимые значения

TRUE | FALSE

serverURL

Описание

Свойство Campaign > monitoring > serverURL задает URL сервера Operational Monitoring. Это обязательный параметр; измените значение, если URL сервера Operational Monitoring отличается от значения по умолчанию.

Если компонент Campaign сконфигурирован для использования связи Secure Sockets Layer (SSL), задайте это свойство, так чтобы использовать HTTPS. Например: serverURL=https://*хост:порт_SSL*/Campaign/OperationMonitor, где:

- *хост* - это имя или IP-адрес компьютера, на котором установлена веб-программа.
- *порт_SSL* - это порт SSL веб-программы.

Обратите внимание на https в URL.

Значение по умолчанию

http://localhost:7001/Campaign/OperationMonitor

monitorEnabledForInteract

Описание

Если задано значение TRUE, включает сервер соединителя Campaign JMX для Interact. В Campaign нет безопасности JMX.

Если задано значение FALSE, вы не сможете соединиться с сервером соединителя Campaign JMX.

Этот мониторинг JMX предназначается только для модуля хронологии контактов и ответов Interact.

Значение по умолчанию

FALSE

Допустимые значения

TRUE | FALSE

Доступность

Это свойство применимо, только если вы установили Interact.

protocol

Описание

Протокол приема для сервера соединителя Campaign JMX, если для monitorEnabledForInteract задано значение yes.

Этот мониторинг JMX предназначается только для модуля хронологии контактов и ответов Interact.

Значение по умолчанию

JMXMP

Допустимые значения

JMXMP | RMI

Доступность

Это свойство применимо, только если вы установили Interact.

port

Описание

Порт приема для сервера соединителя Campaign JMX, если для `monitorEnabledForInteract` задано значение yes.

Этот мониторинг JMX предназначается только для модуля хронологии контактов и ответов Interact.

Значение по умолчанию

2004

Допустимые значения

Целое число от 1025 до 65535.

Доступность

Это свойство применимо, только если вы установили Interact.

Campaign | partitions | partition[n] | Interact | outboundChannels

Эти свойства конфигурации позволяют настроить исходящие каналы для инициируемых сообщений.

имя категории

Описание

Это свойство задает имя для этого исходящего канала. Это имя должно быть уникальным среди всех исходящих каналов.

Имя

Описание

Имя вашего исходящего канала.

Примечание: Чтобы изменения вступили в силу, нужно перезапустить сервер прикладных программ.

Campaign | partitions | partition[n] | Interact | outboundChannels | Parameter Data

Эти свойства конфигурации позволяют настроить исходящие каналы для инициируемых сообщений.

category name

Описание

Это свойство задает имя для этого параметра. Имя должно быть уникальным среди всех параметров для этого исходящего канала.

value

Описание

Это свойство в формате пар имя-значение определяет параметры, необходимые этому исходящему шлюзу.

Campaign | partitions | partition[n] | Interact | Simulator

Эти свойства конфигурации определяют группу серверов, которую вы хотите использовать для выполнения симуляций API.

serverGroup

Описание

Задайте группу серверов времени выполнения, которая используется для выполнения симуляций API.

Значение по умолчанию

defaultServerGroup

Глава 15. Персонализация предложений в реальном времени на стороне клиента

Могут возникнуть ситуации, когда вам нужно обеспечить персонализацию предложений в реальном времени, не реализуя код Java низкого уровня или вызовы SOAP к серверу Interact. Например, посетитель может изначально загрузить веб-страницу, на которой для вашего расширенного программирования доступно только содержимое Javascript, или посетитель открывает сообщение электронной почты, в котором возможен только контент HTML. IBM Interact предоставляет несколько соединителей, обеспечивающих управление предложениями в реальном времени в ситуациях, когда вы контролируете только веб-контент, загруженный на стороне клиента, или когда вы хотите упростить интерфейс для Interact.

Ваша установка Interact включает в себя два соединителя для персонализации предложений, инициированной на стороне клиента:

- "О Interact Message Connector". При использовании Message Connector веб-контент в сообщениях электронной почты (например) или в других электронных средствах связи может содержать теги изображения и ссылки, чтобы выполнять вызовы на сервер Interact для представления предложения загружаемой страницей или начальными страницами по щелчку по ссылке перехода.
- "О программе Interact Web Connector" на стр. 316. При использовании соединителя Web (другое название - JS Connector) веб-страницы могут использовать JavaScript на стороне клиента для управления оценкой предложений, представлением и начальными страницами, на которые происходит переход при щелчке по ссылке перехода.

О Interact Message Connector

Interact Message Connector позволяет использовать сообщения электронной почты и другим электронные носители для вызовов IBM Interact, чтобы разрешить представление персонализированных предложений во время открытия, а когда покупатель делает щелчок по ссылке перехода, перенаправлять его на заданный сайт. Это осуществляется путем использования двух ключевых тегов: тег изображения (`IMG`) загружает персонализованные сообщения во время открытия, а тег ссылки (`A`) захватывает информацию о щелчке по ссылке перехода и перенаправляет пользователя на конкретную начальную страницу.

Пример

В следующем примере показан фрагмент кода HTML, который можно включить в маркетинговое средство (например, в сообщении электронной почты), содержащий и URL тега `IMG` (который передает информацию при открытии документа на сервер Interact и получает в ответ соответствующее изображение для предложения), и URL тега `A` (который определяет, какая информация передана на сервер Interact при щелчке по ссылке перехода):

```
<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>
```

В следующем примере тег IMG включен в тег A, что приводит к следующему поведению:

1. Когда сообщение электронной почты открывается, Message Connector получает требование с информацией, закодированной в теге IMG: msgID и linkID для этого сообщения, а также параметры покупателя, такие как ID пользователя, уровень дохода и тип дохода.
2. Эта информация передается через вызов API на сервер среды выполнения Interact.
3. Сервер среды выполнения возвращает предложение в Message Connector, который получает URL изображения для предложения и предоставляет этот URL (с любыми дополнительными включенными параметрами), а также перенаправляет требование изображения по этому URL предложения.
4. Покупатель видит предложение как изображение.

В данной точке покупатель может щелкнуть по этому изображению, некоторым образом ответив на предложение. Этот щелчок по ссылке перехода, используя тег A и его заданный атрибут HREF (задающий URL назначения), отправляет другое требование в Message Connector, а именно, требование начальной страницы, связанной с URL этого предложения. После этого браузер покупателя перенаправляется на начальную страницу, как сконфигурировано в предложении.

Обратите внимание на то, что тег A для щелчка по ссылке перехода не строго обязателен; предложение может состоять только из одного изображения, например, купона, который может напечатать покупатель.

Установка Message Connector

Файлы, необходимые для установки, внедрения и выполнения Message Connector, были автоматически включены при установке сервера среды выполнения IBM Interact. В этом разделе приведена сводка шагов, необходимые для подготовки Message Connector к использованию.

Установка и внедрение Message Connector включает в себя следующие задачи:

- Необязательно, конфигурирование параметров по умолчанию для Message Connector, как описано в разделе “Конфигурирование Message Connector”.
- Создание таблиц базы данных для хранения данных транзакций Message Connector, как описано в разделе “Создание таблиц соединителя сообщений” на стр. 311.
- Установка веб-программы Message Connector, как описано в разделе “Внедрение и запуск соединителя сообщений” на стр. 312.
- Создание тегов IMG и A в ваших маркетинговых носителях (например, в сообщении электронной почты или на веб-странице), необходимых для вызова предложений Message Connector при открытии или щелчке по ссылке перехода, как описано в разделе “Создание ссылок для Message Connector” на стр. 313.

Конфигурирование Message Connector

Прежде чем внедрять Message Connector, необходимо настроить файл конфигурации, включенный в установку, чтобы он соответствовал вашей конкретной среде. Можно изменить файл XML MessageConnectorConfig.xml, находящийся в вашем каталоге Message Connector на сервере среды выполнения Interact, например, <домашний_каталог_Interact>/msgconnector/config/MessageConnectorConfig.xml.

Об этой задаче

Файл MessageConnectorConfig.xml содержит несколько обязательных и несколько дополнительных параметров конфигурации. Это параметры, которые нужно

настроить для вашей конкретной установки. Следуйте описанным здесь шагам для изменения конфигурации.

Процедура

1. Если Message Connector внедрен и работает на вашем сервере веб-программ, прежде чем продолжать, отмените внедрение Message Connector.
2. На сервере среды выполнения Interact откройте файл `MessageConnectorConfig.xml` в любом текстовом редакторе или редакторе XML.
3. Измените параметры конфигурации, как вам требуется, убедившись, что следующие *обязательные* параметры соответствуют вашей установке.
   - `<interactUrl>`, URL сервера среды выполнения Interact; с этим сервером должны соединиться теги страниц Message Connector, и на нем работает Message Connector.
   - `<imageErrorLink>`, URL, по которому Message Connector перенаправит пользователя в случае возникновения ошибки при обработке требования для изображения предложения.
   - `<landingPageErrorLink>`, URL, по которому Message Connector перенаправит пользователя в случае возникновения ошибки при обработке требования для начальной страницы предложения.
   - `<audienceLevels>`, раздел файла конфигурации, содержащий один или несколько наборов параметров уровня аудитории и указывающий уровень аудитории по умолчанию, если никакой уровень не задан ссылкой Message Connector. Должен существовать по крайней мере один сконфигурированный уровень аудитории.

   Более подробно все параметры конфигурации описаны в разделе “Параметры конфигурации соединителя сообщений”.
4. После завершения изменения конфигурации сохраните и закройте файл `MessageConnectorConfig.xml`.
5. Продолжите конфигурирование и внедрение Message Connector.

Параметры конфигурации соединителя сообщений:

Чтобы сконфигурировать Message Connector, можно изменить файл XML `MessageConnectorConfig.xml`, находящийся в вашем каталоге Message Connector на сервере среды выполнения Interact, обычно это `<домашний_каталог_Interact>/msgconnector/config/MessageConnectorConfig.xml`. Здесь описаны все конфигурации из этого файла XML. Учтите, что в случае изменения этого файла после внедрения и запуска Message Connector необходимо отменить внедрение Message Connector и повторно внедрить его или перезапустить сервер прикладных программ, чтобы перезагрузить эти параметры после выполненных изменений файла.

Общие параметры

В следующей таблице содержится список обязательных и необязательных параметров, содержащихся в разделе `generalSettings` файла `MessageConnectorConfig.xml`.

Таблица 24. Общие параметры соединителя сообщений

Элемент	Описание	Значение по умолчанию
<interactURL>	URL сервера среды выполнения Interact для обработки вызовов от тегов страниц Message Connector, такой как сервер среды выполнения, на котором запущен Message Connector. Этот элемент - обязательный.	http://localhost:7001/interact
<defaultDateTimeFormat>	Формат даты по умолчанию.	дд.MM.гггг
<log4jConfigFileLocation>	Положение файла свойств Log4j. Если задано, относится к переменной среды *$ДОМАШНИЙ_КАТАЛОГ_MESSAGE_CONNECTOR*; в противном случае это значение относительно корневого пути веб-программы Message Connector.	config/MessageConnectorLog4j.properties

Значения параметров по умолчанию

В следующей таблице содержится список обязательных и необязательных параметров, содержащихся в разделе defaultParameterValues файла MessageConnectorConfig.xml.

Таблица 25. Значения по умолчанию параметров соединителя сообщений

Элемент	Описание	Значение по умолчанию
<interactiveChannel>	Имя интерактивного канала по умолчанию.	
<interactionPoint>	Имя точки взаимодействия по умолчанию.	
<debugFlag>	Определяет, включена ли отладка. Допустимые значения: true и false.	нет
<contactEventName>	Имя по умолчанию отправленного события контакта.	
<acceptEventName>	Имя по умолчанию отправленного события принятия.	
<imageUrlAttribute>	Имя по умолчанию атрибута предложения, содержащее URL изображения предложения, если ничего не задано в ссылке Message Connector.	
<landingPageUrlAttribute>	URL по умолчанию начальной страницы перехода по ссылке, если ничего не задано в ссылке Message Connector.	

Параметры поведения

В следующей таблице содержится список обязательных и необязательных параметров, содержащихся в разделе behaviorSettings файла MessageConnectorConfig.xml.

Таблица 26. Параметры поведения соединителя сообщений

Элемент	Описание	Значение по умолчанию
<imageErrorLink>	URL по которому соединитель выполняет перенаправление, если произошла ошибка при обработке требования для изображения предложения. Этот параметр обязателен.	/images/default.jpg
<landingPageErrorLink>	URL, по которому соединитель выполняет перенаправление, если произошла ошибка при обработке требования для начальной страницы перехода при щелчке по ссылке. Этот параметр обязателен.	/jsp/default.jsp
<alwaysUseExistingOffer>	Определяет, должно ли возвращаться кэшированное предложение, даже если срок его годности уже истек. Допустимые значения: true и false.	нет
<offerExpireAction>	Действие, которое следует выполнить, если исходное предложение найдено, но срок его годности уже истек. Допустимые значения: • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

Параметры системы хранения

В следующей таблице содержится список обязательных и необязательных параметров, содержащихся в разделе storageSettings файла MessageConnectorConfig.xml.

Таблица 27. Параметры системы хранения Message Connector

Элемент	Описание	Значение по умолчанию
<persistenceMode>	Когда кэш сохраняет новые записи в базу данных. Разрешенные значения - WRITE-BEHIND (когда первоначально данные записываются в кэш, а позднее передаются в базу данных) и WRITE-THROUGH (когда данные записываются в кэш и базу данных одновременно).	WRITE-THROUGH
<maxCacheSize>	Максимальное число записей в кэше памяти.	5000
<maxPersistenceBatchSize>	Максимальный размер пакета при сохранении записей в базу данных.	200
<macCachePersistInterval>	Максимальный срок в секундах, в течение которого запись остается в кэше, прежде чем она сохраняется в базу данных.	3
<maxElementOnDisk>	Максимальное число записей в кэше диска.	5000

Таблица 27. Параметры системы хранения Message Connector (продолжение)

Элемент	Описание	Значение по умолчанию
<cacheEntryTimeToExpireInSeconds>	Максимальное время хранения записей в кэше диска, прежде чем срок их хранения истекает.	60000
<jdbcSettings>	Раздел файла XML, содержащий конкретную информацию, если используется соединение JDBC. Это взаимоисключающие данные с разделом <dataSourceSettings>.	По умолчанию сконфигурировано соединение с базой данных SQLServer, которая сконфигурирована на локальном сервере, но если включить этот раздел, необходимо предоставить действительные параметры JDBC и регистрационные данные для входа в систему.
<dataSourceSettings>	Раздел файла XML, содержащий конкретную информацию, если используется соединение с источником данных. Это взаимоисключающие данные с разделом <jdbcSettings>.	Конфигурируется по умолчанию для соединения с источником данных InteractDS, определенным на локальном сервере веб-программ.

Уровни аудитории

В следующей таблице содержится список обязательных и необязательных параметров, содержащихся в разделе audienceLevels файла MessageConnectorConfig.xml.

Обратите внимание на то, что дополнительно используется элемент уровней аудитории audienceLevels для задания используемого уровня аудитории по умолчанию, если ничего не задано в ссылке Message Connector, как в следующем примере:

```
<audienceLevels default="Customer">
```

В этом примере значение для атрибута по умолчанию совпадает с именем audienceLevel, заданным в этом разделе. В этом файле конфигурации должен быть определен по крайней мере один уровень аудитории.

Таблица 28. Параметры уровня аудитории соединителя сообщений

Элемент	Элемент	Описание	Значение по умолчанию
<audienceLevel>		Элемент, содержащий конфигурацию уровня аудитории. Предоставьте атрибут имени, как в <audienceLevel name="Customer">	
	<messageLogTable>	Имя таблицы журнала. Это обязательное значение.	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<поле>	Определение одного или нескольких полей ID аудитории для этого audienceLevel.	
	<name>	Имя поля ID аудитории, как задано в среде выполнения Interact.	
	<httpParameterName>	Соответствующее имя параметра для этого поля ID аудитории.	

Таблица 28. Параметры уровня аудитории соединителя сообщений (продолжение)

Элемент	Элемент	Описание	Значение по умолчанию
	<dbColumnName>	Соответствующее имя столбца в базе данных для этого поля ID аудитории.	
	<type>	Тип поля ID аудитории, как задано в среде выполнения Interact. Значением может быть string или numeric.	

Создание таблиц соединителя сообщений

Прежде чем внедрять IBM Interact Message Connector, надо создать таблицы в базе данных, где будут храниться данные среды выполнения Interact. Нужно создать по одной таблице для каждого определенного вами уровня аудитории. Для каждого уровня аудитории Interact будет использовать созданные вами таблицы для записи информации о транзакциях Message Connector.

Об этой задаче

Используйте клиент базы данных, чтобы запустить сценарий SQL Message Connector для соответствующей базы данных или схемы и создать необходимые таблицы. Эти сценарии SQL для вашей поддерживаемой базы данных устанавливаются автоматически при установке сервера среды выполнения Interact. Смотрите в рабочих листах, которые вы заполнили в *Руководстве по установке IBM Interact*, подробности о соединениях с базой данных, где содержатся таблицы среды выполнения Interact.

Процедура

1. Запустите свой клиент базы данных и соединитесь с базой данных, в которой в настоящее время хранятся ваши таблицы среды выполнения Interact.
2. Запустите соответствующий сценарий в каталоге *<домашний_каталог_Interact>*/msgconnector/scripts/ddl. В следующей таблице перечислены образцы сценариев SQL, которые можно использовать для создания таблиц Message Connector вручную:

Таблица 29. Сценарии для создания таблиц соединителя сообщений

Тип источника данных	Имя сценария
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

Обратите внимание на то, что эти сценарии представлены как образцы. Вы можете использовать другое соглашение об именовании или структуру значений ID аудиторий, поэтому может потребоваться изменить сценарий перед его запуском. В общем случае рекомендуется, чтобы для каждого уровня аудитории была выделена одна таблица.

Эти таблицы создаются для следующей информации:

Таблица 30. Информация, создаваемая образцами сценариев SQL

Имя столбца +	Описание
LogId	Первичный ключ этой записи.
MessageId	Уникальный идентификатор для каждого экземпляра сообщения.

Таблица 30. Информация, создаваемая образцами сценариев SQL (продолжение)

Имя столбца +	Описание
LinkId	Уникальный идентификатор для каждой ссылки в электронном носителе (таком как сообщение электронной почты).
OfferImageUrl	URL для связанного изображения возвращенного предложения.
OfferLandingPageUrl	URL для связанной начальной страницы возвращенного предложения.
TreatmentCode	Код процедуры для возвращенного предложения.
OfferExpirationDate	Дата и время истечения срока возвращенного предложения.
OfferContactDate	Дата и время, когда предложение было возвращено клиенту.
AudienceId	ID аудитории электронного носителя.

Обратите внимание на следующее для этой таблицы:

- В зависимости от уровня аудитории будет существовать один столбец ID аудитории (AudienceId) для каждого компонента ключа аудитории.
- Сочетание ID сообщения, ID ссылки и ID аудитории образует ключ уникальности этой таблицы.

После завершения выполнения сценария у вас будут созданы необходимые для Message Connector таблицы.

Результаты

Теперь вы готовы к внедрению веб-программы Message Connector.

Внедрение и запуск соединителя сообщений

IBM Interact Message Connector внедряется как автономная веб-программа на поддерживаемом сервере веб-программ.

Прежде чем начать

Прежде чем внедрять Message Connector, убедитесь, что выполнены следующие задачи:

- У вас должен быть установлен сервер среды выполнения IBM Interact. Доступная для внедрения прикладная программа Message Connector автоматически устанавливается вместе с сервером среды выполнения и готова для внедрения из вашего домашнего каталога Interact.
- Необходимо запустить также сценарии SQL, представленные с вашей установкой, чтобы создать нужные таблицы в базе данных среды выполнения Interact для использования программой Message Connector, как описано в разделе “Создание таблиц соединителя сообщений” на стр. 311

Об этой задаче

Так же, как вы внедряете другие прикладные программы IBM на сервере веб-программ, прежде чем запускать их, необходимо внедрить прикладную программу Message Connector, чтобы она была доступна для представления предложений.

Процедура

1. Чтобы внедрить прикладную программу, соединитесь с интерфейсом управления сервера веб-программ с необходимыми привилегиями.

2. Следуйте инструкциям для вашего сервера веб-программ, чтобы внедрить и запустить файл *домашний_каталог_<Interact>*/msgconnector/MessageConnector.war Замените каталог *домашний_каталог_<Interact>* на правильный каталог, где установлен сервер среды выполнения Interact.

Результаты

Теперь Message Connector доступен для использования. После конфигурирования вашей установки Interact для создания основных данных, которые будет использовать Message Connector при предоставлении предложений, таких как интерактивные каналы и стратегии, потоковые диаграммы, предложения и т.д., можно создать ссылки в ваших электронных носителях, принимаемые Message Connector.

Создание ссылок для Message Connector

Чтобы использовать Message Connector для предоставления пользовательских изображений предложения, когда конечный пользователь взаимодействует с вашим электронным носителем (например, открывает сообщение электронной почты), и пользовательских начальных страниц, когда конечный пользователь щелкает по ссылке для перехода в предложении, необходимо создать ссылки для включения в ваше сообщение. В этом разделе представлена сводка по заданию тегов HTML для этих ссылок.

Об этой задаче

Независимо от системы, используемой для генерирования выходных сообщений для конечных пользователей, вам нужно сгенерировать задание тегов HTML с соответствующими полями (представленными в тегах HTML как атрибуты), содержащими информацию для передачи на сервер среды выполнения Interact. Следуйте перечисленным ниже шагам, чтобы сконфигурировать минимальную информацию для сообщения Message Connector.

Обратите внимание на то, что хотя эти инструкции относятся конкретно к сообщениям со ссылками Message Connector, вы можете следовать тем же шагам и выполнить такое же конфигурирование для добавления ссылок на веб-страницы или любые другие электронные носители.

Процедура

1. Создайте ссылку IMG, которая появится в вашем сообщении, как минимум, со следующими параметрами:
   - msgID, определяет уникальный идентификатор для этого сообщения.
   - linkID, определяет уникальный идентификатор для ссылки в этом сообщении.
   - audienceID, идентификатор аудитории, к которой принадлежит получатель этого сообщения.

     Обратите внимание на то, что если ID аудитории - составной ID, все его компоненты должны быть включены в ссылку.

   Можно включить также дополнительные параметры, содержащие уровень аудитории, имя интерактивного канала, имя точки взаимодействия, URL положения изображения и ваши пользовательские параметры, не используемые специально Message Connector.
2. Дополнительно можно создать ссылку A, включающую в себя ссылку IMG, так что при щелчке по изображению браузер загрузит страницу, содержащую предложение для пользователя. Ссылка A должна содержать также три перечисленные выше параметра (msgID, linkID и audienceID), а также любые

дополнительные параметры (уровень аудитории, имя интерактивного канала и имя точки взаимодействия) и ваши пользовательские параметры, не используемые специально Message Connector. Обратите внимание на то, что ссылка A обычно содержит ссылку Message Connector IMG, но при необходимости можно располагать ее на странице и автономно. Если эта ссылка не содержит ссылки IMG, ссылка IMG должна содержать тот же набор параметров, что и объемлющая ссылка A (в том числе любые дополнительные или пользовательские параметры).

3. После того, как ваши ссылки будут правильно определены, сгенерируйте и отправьте сообщение электронной почты.

Результаты

Подробную информацию о доступных параметрах и образцы ссылок смотрите в разделе “Параметры требования HTTP с тегами "IMG" и "A"”

Параметры требования HTTP с тегами "IMG" и "A"

Когда Message Connector получает требование - либо при открытии конечным пользователем сообщения электронной почты с закодированным для Message Connector тегом IMG, либо при щелчке конечного пользователя по тегу перехода A, программа анализирует параметры, включенные в требование, чтобы возвратить соответствующие данные предложения. В этом разделе предоставлен список параметров, которые могут быть включены в URL требования - либо в тег IMG (автоматически загружаемый при показе изображения с тегом при открытии сообщения электронной почты), либо в тег A (загружаемый, когда получатель сообщения электронной почты щелкает в сообщении по ссылке перехода на заданный сайт).

Параметры

Когда Message Connector получает требование, он анализирует параметры, включенные в это требование. Эти параметры включают в себя некоторые или все из следующих:

Имя параметра	Описание	Обязательно?	Значение по умолчанию
msgId	Уникальный идентификатор экземпляра сообщения электронной почты или веб-страницы.	Да	Нет. Представляется системой, создающей уникальный экземпляр сообщения электронной почты или веб-страницы с тегом.
linkId	Уникальный идентификатор ссылки в этом сообщении электронной почты или на веб-странице.	Да	Нет. Представляется системой, создающей уникальный экземпляр сообщения электронной почты или веб-страницы с тегом.
audienceLevel	Уровень аудитории, к которому принадлежит получатель этого сообщения.	Нет	Уровень аудитории audienceLevel, заданный по умолчанию в элементе audienceLevels в файле MessageConnectorConfig.xml.
ic	Имя интерактивного канала назначения	Нет	Имя элемента interactiveChannel в разделе defaultParameterValues файла MessageConnectorConfig.xml, по умолчанию "interactiveChannel".
ip	Имя применяемой точки взаимодействия (interaction point, IP)	Нет	Имя элемента interactionPoint в разделе defaultParameterValues файла MessageConnectorConfig.xml, по умолчанию "headBanner".
offerImageUrl	URL изображения предложения назначения для IMG URL в сообщении.	Нет	Нет.

Имя параметра	Описание	Обязательно?	Значение по умолчанию
offerImageUrlAttr	Имя атрибута предложения с URL изображения назначения предложения	Нет	Значение элемента imageUrlAttribute в разделе defaultParameterValues файла MessageConnectorConfig.xml.
offerLandingPageUrl	URL начальной страницы, соответствующей предложению назначения.	Нет	Нет.
offerLandingPageUrlAttr	Имя атрибута предложения с URL начальной страницы, соответствующей предложению назначения.	Нет	Значение элемента landingPageUrlAttribute в разделе defaultParameterValues файла MessageConnectorConfig.xml.
contactEvent	Имя события контакта.	Нет	Имя элемента contactEventName в разделе defaultParameterValues файла MessageConnectorConfig.xml, по умолчанию "contact".
responseEvent	Имя события восприятия.	Нет	Имя элемента acceptEventName в разделе defaultParameterValues файла MessageConnectorConfig.xml, по умолчанию "accept".
debug	Флаг отладки. Задайте для этого параметра значение "true" только при поиске и устранении неисправностей по указанию службы технической поддержки IBM .	Нет	Имя элемента debugFlag в разделе defaultParameterValues файла MessageConnectorConfig.xml, по умолчанию "false".
<*audience id*>	ID аудитории этого пользователя. Имя этого параметра определено в файле конфигурации.	Да	Нет.

Когда Message Connector получает нераспознаваемый параметр (то есть отсутствующий в приведенном выше списке), он обрабатывает этот параметр одним из двух возможных способов:

- Если представлен нераспознаваемый параметр (например, "attribute", как в attribute="attrValue") и существует подходящий параметр с таким же именем плюс слово "Type" (например, "attributeType", как в attributeType="string"), Message Connector создаст совпадающий параметр Interact и передаст его в среду выполнения Interact.

  Значение для параметра Type может быть одним из следующих:

  – string
  – numeric
  – datetime

    Для параметра типа "datetime" Message Connector ищет также параметр с таким же именем плюс слово "Pattern" (например, "attributePattern"), значение которого в допустимом формате даты/времени. Например, можно предоставить параметр attributePattern="MM/dd/yyyy".

    Обратите внимание на то, что если задать тип параметра "datetime", но не предоставить соответствующий паттерн даты, будет использоваться значение, указанное в файле конфигурации Message Connector (находится в <каталог_установки>/msgconnector/config/MessageConnectorConfig.xml) на сервере Interact.
- Если предоставлен нераспознаваемый параметр и отсутствует соответствующее значение Type, Message Connector передает этот параметр по URL перенаправления назначения.

Все нераспознаваемые параметры Message Connector передает на сервер среды выполнения Interact без обработки и сохранения.

Пример кода Message Connector

Следующий тег A содержит пример набора ссылок Message Connector, которые могут появиться в сообщении электронной почты:

```
<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>
```

В этом примере тег IMG загружается автоматически, когда открывается сообщение электронной почты. Получив изображение с указанной страницы, сообщение передает параметры для уникального идентификатора сообщений (msgID), уникального идентификатора ссылки (linkID) и уникального идентификатора пользователя (userid) вместе с двумя дополнительными параметрами (incomeCode и incomeType), которые должны быть переданы в среду выполнения Interact.

Тег A предоставляет атрибут HREF (Hypertext Reference), который преобразует изображение предложения в работающую по щелчку ссылку в сообщении электронной почты. Если просматривающий сообщение, увидев его, щелкает по ссылке перехода на начальную страницу, уникальный идентификатор сообщения (msgId), идентификатор ссылки (linkId)и идентификатор пользователя (userid) передаются на сервер, как и один дополнительный параметр (referral), передающийся по URL перенаправления назначения.

О программе Interact Web Connector

Программа Interact WebConnector (ее называют также JavaScript Connector или JSConnector) предоставляет службу на сервере времени выполнения Interact, которая позволяет коду JavaScript вызывать API Interact Java. Это позволяет веб-страницам выполнять вызовы в Interact для персонализации предложений в реальном времени с использованием только встроенного кода JavaScript, без использования языков веб-разработки (таких, как Java, PHP, JSP и т.п.). Например, на каждой странице вашего веб-сайта, предназначенной для предложений, которые рекомендованы Interact, можно встроить небольшой фрагмент кода JavaScript, выполняющий при каждой загрузке страницы вызовы API Interact, чтобы для посетителя сайта при загрузке страницы гарантированно выводились лучшие предложения.

Веб-соединитель Interact используется там, где вы хотите вывести для посетителей предложения на странице, программного управления выводом которой у вас может быть не быть на стороне сервера (как, например, в случае сценариев на основе сервера PHP или других серверов), но там, где вы по-прежнему можете встроить в содержимое страницы код JavaScript, который будет выполняться браузером посетителя.

Совет: Файлы веб-соединителя Interact устанавливаются на сервер среды выполнения Interact автоматически в каталоге *<домашний_каталог_Interact>*/jsconnector. В каталоге *<домашний_каталог_Interact>*/jsconnector вы найдёте файл ReadMe.txt, содержащий важные замечания и подробности о функциональных возможностях веб-соединителя, а также файлы примеров и исходный код веб-соединителя, которые можно взять за основу для разработки ваших собственных решений. Если вы не найдёте здесь информацию, отвечающую на ваши вопросы, смотрите дополнительную информацию в каталоге jsconnector.

Установка веб-соединителя на сервере среды выполнения

Экземпляр веб-соединителя устанавливается с сервером среды выполнения IBM Interact автоматически, и его поддержка включается по умолчанию. Однако есть некоторые параметры, значения которых нужно изменить прежде, чем можно будет сконфигурировать и использовать веб-соединитель.

Об этой задаче

Параметры, значения которых нужно изменить прежде, чем можно будет использовать веб-соединитель, устанавливаемый на сервере среды выполнения, добавляются в конфигурацию сервера веб-программ. По этой причине сервер веб-программ по завершении этих действий нужно будет перезапустить.

Процедура

1. Для сервера веб-программ, на котором установлен сервер среды выполнения Interact, задайте следующие свойства Java.

   `-DUI_JSCONNECTOR_ENABLE_INPROCESS=true`

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Замените *<домашний_каталог_соединителя_jsconnector>* на путь к каталогу `jsconnector` на сервере среды выполнения, а именно, к каталогу `<домашний_каталог_Interact>/jsconnector`.

   Способ, которым вы задаёте свойства Java, зависит от используемого сервера веб-программ. Например, в WebLogic можно отредактировать файл `startWebLogic.sh` или `startWebLogic.cmd` файл, чтобы изменить значение параметра `JAVA_OPTIONS`, как в следующем примере:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
   jsconnector"
   ```

   В WebSphere Application Server это свойство задаётся на панели виртуальной Java-машины консоли администрирования.

   Конкретные инструкции по заданию свойств Java смотрите в документации используемого сервера веб-программ.

2. Перезапустите используемый сервер веб-программ, если он уже был запущен, или запустите его сейчас для гарантии, что будут использоваться новые свойства Java.

Результаты

По завершении сервером веб-программ своего процесса запуска установка веб-соединителя на сервере среды выполнения будет закончена. Следующий шаг - это соединение с веб-страницей конфигурации веб-соединителя по адресу `http://<хост>:<порт>/interact/jsp/WebConnector.jsp`, где *<хост>* - имя сервера среды выполнения Interact, а *<порт>* - порт, на котором веб-соединитель осуществляет приём (как задано сервером веб-программ).

Установка веб-соединителя как отдельной веб-программы

Экземпляр веб-соединителя устанавливается с сервером среды выполнения IBM Interact автоматически, и его поддержка включается по умолчанию. Однако веб-соединитель можно также внедрить как его собственную веб-программу (например, на сервере веб-программ в отдельной системе) и сконфигурировать его для связи с удалённым сервером среды выполнения Interact.

Об этой задаче

Приведённые инструкции описывают процесс внедрения веб-соединителя как отдельной веб-программы с доступом к удалённому серверу среды выполнения Interact.

Для возможности внедрения веб-соединителя уже должен быть установлен сервер среды выполнения IBM Interact, и у вас должен быть сервер веб-программ в другой системе с сетевым доступом (не блокированным никаким брандмауэром) к серверу среды выполнения Interact.

Процедура

1. Скопируйте каталог `jsconnector`, содержащий файлы веб-соединителя, с сервера среды выполнения Interact в систему, где сервер уже сконфигурирован и работает сервер веб-программ (такой как WebSphere Application Server). Каталог `jsconnector` можно найти в каталоге установки Interact.
2. В системе, где будет внедрён экземпляр веб-соединителя, сконфигурируйте файл `jsconnector/jsconnector.xml` при помощи любого текстового редактора или редактора XML, чтобы изменить атрибут `interactURL`.

   Он задается умолчанию как `http://localhost:7001/interact`, но его нужно изменить, чтобы он соответствовал URL удаленного сервера среды выполнения Interact, такому как `http://runtime.example.com:7011/interact`.

   После внедрения веб-соединителя можно при помощи веб-интерфейса настроить остальные параметры в файле `jsconnector.xml`. Более подробную информацию смотрите в разделе “Конфигурирование веб-соединителя” на стр. 319.
3. Для сервера веб-программ, на котором будет внедрён веб-соединитель, задайте следующее свойство Java:

   `-DUI_JSCONNECTOR_HOME=<`*`домашний_каталог_соединителя_js`*`>`

   Замените *<домашний_каталог_соединителя_js>* на фактический путь, куда вы скопировали свойство `jsconnector`, на сервере веб-программ.

   Способ, которым вы задаёте свойства Java, зависит от используемого сервера веб-программ. Например, в WebLogic можно отредактировать файл `startWebLogic.sh` или `startWebLogic.cmd` файл, чтобы изменить значение параметра `JAVA_OPTIONS`, как в следующем примере:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/InteractFiles/
   jsconnector"
   ```

   В WebSphere Application Server это свойство задаётся на панели виртуальной Java-машины консоли администрирования.

   Конкретные инструкции по заданию свойств Java смотрите в документации используемого сервера веб-программ.
4. Перезапустите используемый сервер веб-программ, если он уже был запущен, или запустите его на этом шаге для гарантии, что будет использоваться новое свойство Java.

   Перед тем, как продолжить, подождите, пока сервер веб-программ не завершит процесс запуска.
5. Соединитесь с интерфейсом управления сервером веб-программ с необходимыми привилегиями для внедрения прикладной программы.
6. Выполните инструкции для используемого сервера веб-программ, чтобы внедрить и запустить следующий файл: `jsConnector/jsConnector.war`

Результаты

Теперь веб-соединитель внедрён в веб-программе. После того, как сервер Interact полностью сконфигурирован и работает, следующий шаг - это соединение с веб-страницей конфигурации веб-соединителя по адресу `http://` *<хост>*`:` *<порт>*`/interact/jsp/WebConnector.jsp`, где *<хост>* - система, где запускается сервер веб-программ, на котором вы только что внедрили веб-соединитель, а *<порт>* - порт, на котором веб-соединитель осуществляет приём (как задано сервером веб-программ).

Конфигурирование веб-соединителя

Параметры конфигурации для веб-соединителя Interact сохраняются в файле `jsconnector.xml`, хранимом в системе, где внедряется веб-соединитель (в такой, как система самого сервера среды выполнения Interact или в отдельной системе, где запускается сервер прикладных программ). Файл `jsconnector.xml` можно отредактировать непосредственно при помощи любого текстового редактора или редактора XML; однако более простой способ сконфигурировать почти все доступные параметры конфигурации - это использовать страницу конфигурации веб-соединителя из вашего браузера.

Прежде чем начать

Для возможности конфигурирования веб-соединителя при помощи веб-интерфейса сначала нужно установить и внедрить веб-программу, предоставляющую веб-соединитель. На сервере среды выполнения Interact экземпляр веб-соединителя устанавливается автоматически при установке и внедрении Interact. На любом другом сервере веб-программ нужно установить и внедрить веб-программу соединителя, как описано в разделе “Установка веб-соединителя как отдельной веб-программы” на стр. 317.

Процедура

1. Откройте поддерживаемый веб-браузер и раскройте URL, подобный следующему:

   `http://`*<хост>*`:`*<порт>*`/interact/jsp/WebConnector.jsp`

   - Замените *<хост>* на имя сервера, где запускается веб-соединитель, такое как имя хоста сервера среды выполнения или имя сервера, на котором внедряется отдельный экземпляр веб-соединителя.
   - Замените *<порт>* на номер порта, где веб-программа соединителя выполняет приём соединений (обычно он совпадает с портом по умолчанию сервера веб-программ).
2. На появившейся странице Конфигурации заполните следующие разделы:

Таблица 31. Сводка параметров конфигурации веб-соединителя.

Раздел	Параметры
Основные параметры	Страница Основные параметры используется для конфигурирования всего поведения веб-соединителя для сайта, на котором будут развернуты помеченные тегами страницы. В состав этих параметров входит базовый URL для сайта, информация о посетителях сайта, которую должен использовать Interact, и подобные им параметры, применяемые ко всем страницам, которые планируется помечать тегами при помощи кода веб-соединителя. Подробную информацию смотрите в разделе “Основные опции конфигурации веб-соединителя” на стр. 321.

Таблица 31. Сводка параметров конфигурации веб-соединителя (продолжение).

Раздел	Параметры
Типы вывода HTML	Страница Типы вывода HTML используется для определения кода HTML, который будет предоставляться для каждой точки взаимодействия на странице. Возможен выбор в списке шаблонов по умолчанию (файлов .flt), содержащих определённое сочетание кода каскадной таблицы стилей (cascading style sheet, CSS), кода HTML и кода JavaScript для использования в каждой точке взаимодействия. Можно использовать предоставляемые шаблоны, настроить их должным образом или создать собственные. Параметры конфигурации на этой странице соответствуют разделу `interactionPoints` файла конфигурации `jsconnector.xml`. Подробную информацию смотрите в разделе “Типы вывода HTML конфигурации веб-соединителя” на стр. 323.
Дополнит. страницы	Страница Дополнительные страницы используются для отображения конкретных для страницы параметров на паттерн URL. Например, можно сконфигурировать отображение страницы таким образом, чтобы в любом URL, содержащем текст "index.htm", выводилась ваша общая страница приветствия с конкретными событиями загрузки страницы и точками взаимодействия, определёнными для этого отображения. Параметры конфигурации на этой странице соответствуют разделу `pageMapping` файла конфигурации `jsconnector.xml`. Подробную информацию смотрите в разделе “Дополнительные страницы конфигурации веб-соединителя” на стр. 325.

3. На странице Основные параметры убедитесь, что параметры уровня сайта допустимы для установки, и (необязательно) задайте режим отладки (рекомендуется только для устранения неисправностей), интеграцию Digital Analytics for On Premises Page Tag и параметры по умолчанию для большинства точек взаимодействия, после чего щёлкните на странице Конфигурации по ссылке Типы вывода HTML.
4. На странице Типы вывода HTML выполните приведённые действия, чтобы добавить или изменить шаблоны вывода, определяющие точки взаимодействия на веб-странице покупателя.

   По умолчанию шаблоны вывода (файлы .flt) сохраняются в каталоге *<домашний_каталог_jsconnector>*`/conf/html`.

   a. Выберите файл .flt в списке, который вы хотите исследовать или использовать в качестве начальной точки, или нажмите кнопку Добавить тип, чтобы создать новый, пустой шаблон точки взаимодействия для использования.

      Рядом со списком шаблонов появится информация о содержимом шаблона (если оно есть).

   b. (Необязательно) Измените имя шаблона в поле **Имя файла для этого типа вывода**. Для нового шаблона измените имя `CHANGE_ME.flt` на более подходящее.

      При переименовании здесь шаблона веб-соединитель создаст новый файл с этим именем при следующем сохранении этого шаблона. Шаблоны сохраняются при изменении тела текста и последующем переходе к любому другому полю.

   c. Измените или заполните информацию фрагмента HTML должным образом, включая любой код таблицы стилей (CSS), JavaScript и HTML, который вы хотите включить. Обратите внимание на то, что можно также включать переменные, которые будут заменены параметрами Interact в среде

выполнения. Например, строка ${offer.HighlightTitle} будет автоматически заменена заголовком предложения в заданном положении точки взаимодействия.

В примерах появляющихся под полем Фрагмент HTML, указывается, как форматировать блоки кода CSS, JavaScript или HTML.

5. Страница Расширенные страницы используется при необходимости для конфигурирования отображений страниц, определяющих способ, как будут обрабатываться на страницах конкретные паттерны URL.
6. Закончив задание свойств конфигурации, нажмите кнопку **Передать изменения на выполнение**. При нажатии кнопки **Передать изменения на выполнение** выполняются следующие действия:
   - Выводится тег страницы веб-соединителя IBM Interact, содержащий код JavaScript, который можно скопировать со страницы веб-соединителя и вставить его в используемые веб-страницы.
   - Создаётся резервная копия файла конфигурации веб-соединителя на сервере Interact (файла jsconnector.xml на сервере, где устанавливается веб-соединитель), и создаётся новый файл конфигурации с заданными вами параметрами.

     Резервные копии файлов конфигурации сохраняются в виде *<домашний_каталог_jsconnector>*/conf/archive/jsconnector.xml.*<дата>*.*<время>* (как в jsconnector.xml.20111113.214933.750-0500, где строка даты - 20111113, а строка времени, включая индикатор часового пояса - 214933.750-0500).

Результаты

Вы завершили конфигурирование веб-соединителя.

Чтобы изменить конфигурацию, можно либо вернуться к началу этих действий и выполнить их повторно с новыми значениями, либо открыть файл конфигурации (<домашний_каталог_Interact>/jsconnector/conf/jsconnector.xml) в любом текстовом редакторе или редакторе XML и изменить его должным образом.

Основные опции конфигурации веб-соединителя

Страница Основные параметры страницы Конфигурации веб-соединителя используется для конфигурирования всего поведения веб-соединителя для сайта, на котором будут развернуты помеченные тегами страницы. В состав этих параметров входит базовый URL для сайта, информация о посетителях сайта, которую должен использовать Interact, и подобные им параметры, применяемые ко всем страницам, которые планируется помечать тегами при помощи кода веб-соединителя.

Параметры уровня сайта

Опции конфигурации Параметры уровня сайта - это глобальные параметры, влияющие на всё поведение установки конфигурируемого вами веб-соединителя. Можно задать следующие значения:

Таблица 32. Параметры уровня сайта для установки веб-соединителя

Значение	Описание	Эквивалентная установка в jsconnector.xml
URL API Interact	Базовый URL сервера среды выполнения Interact. **Примечание:** Этот параметр используется, только если веб-соединитель запускается не на сервере среды выполнения Interact (то есть внедрён отдельно).	<*interactURL*>
URL веб-соединителя	Базовый URL, используемый для генерирования URL перехода	<*jsConnectorURL*>
Имя интерактивного канала для веб-сайта назначения	Имя интерактивного канала, определённого вами на сервере Interact, который представляет отображение этой страницы.	<*interactiveChannel*>
Уровень аудитории посетителей	Уровень аудитории Campaign для входящего посетителя; используется в вызове API среды выполнения Interact.	<*audienceLevel*>
Имя поля ID аудитории в таблице профилей	Имя поля ID аудитории, которое будет использоваться в вызове API Interact. Имейте в виду, что в текущий момент идентификаторы аудиторий нескольких полей не поддерживаются.	<*audienceIdField*>
Тип данных для поля ID аудитории	Тип данных поля ID аудитории ("numeric" или "string"), используемый в вызове API Interact.	<*audienceIdFieldType*>
Имя опознавательного файла (cookie), представляющее ID сеанса	Имя опознавательного файла (cookie), который будет содержать ID сеанса.	<*sessionIdCookie*>
Имя опознавательного файла (cookie), представляющего ID посетителя	Имя опознавательного файла (cookie), который будет содержать ID посетителя.	<*visitorIdCookie*>

Необязательные возможности

Опции конфигурации Необязательные возможности - это необязательные глобальные параметры для установки конфигурируемого вами веб-соединителя. Можно задать следующие значения:

Таблица 33. Необязательные параметры уровня сайта для установки веб-соединителя

Значение	Описание	Эквивалентная установка в jsconnector.xml
Включить режим отладки	Указывает (при ответе `yes` или `no`), использовать ли особый режим отладки. При включении этой возможности в содержимое, возвращаемое из веб-соединителя, включается вызов JavaScript 'alert', оповещающий клиента об отображении конкретной страницы, которое только что имело место. Для получения оповещения у клиента должна быть запись в файле, задаваемом параметром <*authorizedDebugClients*>.	<*enableDebugMode*>
Файл хостов авторизованных клиентов отладки	Путь к файлу, содержащему список хостов или IP-адресов, которые соответствуют критериям режима отладки. Имя хоста или IP-адрес клиента должны содержаться в указанном файле в качестве собираемой отладочной информации.	<*authorizedDebugClients*>
Включить интеграцию тега страницы Digital Analytics for On Premises	Указывает (при ответе `yes` или `no`), должен ли веб-соединитель присоединить заданный тег IBM Digital Analytics for On Premises в конец содержимого страницы.	`<enableNetInsightTagging>`
Файл шаблона HTML тегов Digital Analytics for On Premises	Шаблон HTML/Javascript, используемый для интеграции вызова тега Digital Analytics for On Premises. В общем случае следует принять значение этого параметра по умолчанию, если вам не будет указано задать другой шаблон.	`<netInsightTag>`

Типы вывода HTML конфигурации веб-соединителя

Страница Типы вывода HTML используется для определения кода HTML, который будет предоставляться для каждой точки взаимодействия на странице. Их можно выбрать в списке шаблонов по умолчанию (файлов .flt), содержащих определённое сочетание кода каскадной таблицы стилей (cascading style sheet, CSS), кода HTML и кода JavaScript для использования в каждой точке взаимодействия. Можно использовать предоставляемые шаблоны, настроить их должным образом или создать собственные.

Примечание: Параметры конфигурации на этой странице соответствуют разделу `interactionPoints` файла конфигурации `jsconnector.xml`.

Точка взаимодействия может также содержать маркеры подстановки (зоны), в которые могут автоматически вставляться атрибуты предложения. Например, можно включить строку `${offer.TREATMENT_CODE}`, которая будет заменена на код процедуры, назначаемый предложению во время взаимодействия.

Шаблоны, появляющиеся на этой странице, загружаются автоматически из файлов, хранимых в каталоге `<домашний_каталог_Interact>/jsconnector/conf/html` веб-соединителя. Все новые шаблоны, создаваемые здесь, также сохраняются в этом каталоге.

Чтобы просмотреть или изменить при помощи страницы Типы вывода HTML какие-либо существующие шаблоны, выберите в списке файл `.flt`.

Чтобы создать на странице Типы вывода HTML новый шаблон, нажмите кнопку **Добавить тип**.

Независимо от метода, выбираемого для создания или изменения шаблона, рядом со списком шаблонов появляется следующая информация:

Значение	**Описание**	**Эквивалентная установка в jsconnector.xml**
Имя файла для этого типа вывода	Имя, назначаемое редактируемому вами шаблону. Это имя должно быть допустимым для операционной системы, в которой запускается веб-соединитель; например, если операционная система - Microsoft Windows, в имени нельзя использовать символ дробной черты (/). Если вы создаете новый файл, в этом поле будет заранее задано `CHANGE_ME.flt`. Перед тем, как продолжить, его надо изменить на значащее значение.	*<htmlSnippet>*

Значение	Описание	Эквивалентная установка в jsconnector.xml
Фрагмент HTML	Конкретное содержимое, которое веб-соединитель должен вставить в точку взаимодействия на веб-странице. Этот фрагмент может содержать код HTML, информацию форматирования CSS или JavaScript для выполнения на странице. Каждый из этих трех типов содержимого должен быть заключен в коды BEGIN и END, как в следующих примерах: • ${BEGIN_HTML} <*ваше содержимое HTML*> ${END_HTML} • ${BEGIN_CSS} <*информация вашей таблицы стилей для точки взаимодействия*> ${END_CSS} • ${BEGIN_JAVASCRIPT} <*ваш код JavaScript для точки взаимодействия*> ${END_JAVASCRIPT} Можно также ввести ряд заранее заданных специальных кодов, заменяемых автоматически при загрузке страницы, включая следующие: • `${logAsAccept}`: Макрокоманда, принимающая два параметра (URL назначения и TreatmentCode, используемые для идентификации принятия предложения) и выполняющий их замену на URL перехода. • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}` • `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` Каждый из перечисленных здесь кодов предложения представляет атрибуты предложения, определенные в шаблоне предложения в IBM Campaign, использовавшегося маркетологом для создания предложений, возвращаемых Interact. Имейте в виду, что веб-соединитель использует механизм шаблонов FreeMarker, предоставляющий множество дополнительных опций, которые могут оказаться полезными при задании кодов шаблонов страницы. Более подробную информацию смотрите в разделе http://freemarker.org/docs/index.html.	Эквивалента нет, поскольку фрагмент HTML находится в своем собственном файле, отдельном от jsconnector.xml.
Пример специальных кодов	Содержит примеры типов специальных кодов, включая коды, идентифицирующие блоки, такие как HTML, CSS, или JAVASCRIPT, и зоны, куда можно выполнять вставки для обращения к конкретным метаданным предложений.	Нет эквивалента.

Вносимые в эту страницу изменения сохраняются автоматически при переходе к другой странице конфигурации веб-соединителя.

Дополнительные страницы конфигурации веб-соединителя

Страница Дополнительные страницы используются для отображения параметров, конкретных для страницы, на паттерн URL. Например, можно сконфигурировать отображение страницы таким образом, чтобы в любом входящем URL, содержащем текст "index.htm", выводилась ваша общая страница приветствия с конкретными событиями загрузки страницы и точками взаимодействия, определёнными для этого отображения.

Примечание: Параметры конфигурации на этой странице соответствуют разделу `pageMapping` файла конфигурации `jsconnector.xml`.

Чтобы создать при помощи страницы Дополнительные страницы новое отображение страницы, щёлкните по ссылке **Добавить страницу** и заполните поля необходимой информацией для отображения.

Информация о странице

Опции конфигурации Информация о странице для отображения страницы определяют паттерн URL, действующий как триггер для этого отображения, плюс некоторые дополнительные параметры для способа, которым это отображение страницы будет обрабатывать Interact.

Значение	Описание	Эквивалентная установка в jsconnector.xml
URL содержит	Это паттерн URL, за которым должен наблюдать веб-соединитель во входящем требовании страницы. Например, если запрашиваемый URL содержит "mortgage.htm", его можно сопоставить со страницей с информацией об ипотеке.	<*urlPattern*>
Псевдоним для этой страницы или набора страниц	Понятное имя для вашей собственной ссылки, описывающей, для чего предназначено это отображение, например: "Страница с информацией об ипотеке".	<*friendlyName*>
Возвращать также предложения как данные JSON для использования JavaScript	Выпадающий список, указывающий, хотите ли вы, чтобы веб-соединитель включал в конце содержимого страницы необработанные данные предложений в формате JavaScript Object Notation (http://www.json.org/).	<*enableRawDataReturn*>

События для активации (утилиты onload) при посещении этой страницы или набора страниц

Этот набор опций конфигурации Информация о странице для отображения страницы определяет паттерн URL, действующий как триггер для этого отображения, плюс некоторые дополнительные параметры для способа, которым это отображение страницы будет обрабатываться Interact.

Примечание: Параметры конфигурации в этом разделе соответствуют разделу `<pageLoadEvents>` файла `jsconnector.xml`.

Значение	Описание	Эквивалентная установка в jsconnector.xml
Отдельные события	Список событий, доступных для этой страницы или набора страниц. События в этом списке - это события, определённые вами в Interact; выберите одно или несколько событий, которые (как вы хотите) будут происходить при загрузке страницы. Последовательность вызовов API Interact будет следующей: 1. `startSession` 2. `postEvent` для каждого отдельного события загрузки страницы (при условии, что вы определили отдельные условия в Interact) 3. Для каждой точки взаимодействия: • `getOffers` • `postEvent(ContactEvent)`	<*событие*>

Точки взаимодействия (положения вывода предложений) на этой странице или наборе страниц

Этот набор опций конфигурации для отображения страницы позволяет выбирать, какие точки взаимодействия будут появляться на странице Interact.

Примечание: Параметры конфигурации в этом разделе соответствуют разделу `<pageMapping>` | `<page>` | `<interactionPoints>` файла `jsconnector.xml`.

Значение	Описание	Эквивалентная установка в jsconnector.xml
Переключатель Имя точки взаимодействия	Каждая точка взаимодействия, определённая в файле конфигурации, появляется в этом разделе страницы. Если включить переключатель рядом с именем точки взаимодействия, будет выводиться ряд опций, доступных для этой точки взаимодействия.	<*interactionPoint*>
ID элемента HTML (Interact задаст innerHTML)	Имя элемента HTML, который должен возвращать содержимое для этой точки взаимодействия. Например, если на странице задано `<div id="welcomebanner">`, введите в этом поле `welcomebanner` (значение ID).	<*htmlElementId*>
Тип вывода HTML	Выпадающий список, позволяющий выбрать тип вывода HTML (фрагменты HTML или файлы .flt, определённые на предыдущей странице конфигурации веб-соединителя), используемый для этой точки взаимодействия.	<*htmlSnippet*>

Значение	Описание	Эквивалентная установка в jsconnector.xml
Максимальное число представляемых предложений (если это карусель или флипбук)	Максимальное число предложений, которые веб-соединитель может получить с сервера Interact сервер для этой точки взаимодействия; это поле необязательное, и применяется только для точки взаимодействия, регулярно обновляющей представляемые предложения, не перезагружая страницу, как в карусельном сценарии, где возвращаются несколько предложений так, что их можно сделать доступными по одному.	<*maxNumberOfOffers*>
Событие для активации при представлении предложения	Имя события контакта, размещаемого для этой точки взаимодействия.	<*contactEvent*>
Событие для активации при принятии предложения	Имя события принятия, размещаемого для этой точки взаимодействия при щелчке по ссылке на предложение.	<*acceptEvent*>
Событие для активации при отклонении предложения	Имя события отклонения, размещаемого для этой точки взаимодействия. **Примечание:** На данный момент эта возможность еще не используется.	<*rejectEvent*>

Опции конфигурации веб-соединителя

Как правило, для конфигурирования параметров веб-соединителя можно использовать графический интерфейс веб-соединителя. Все задаваемые вами параметры сохраняются также в файле jsconnector.xml, находящемся в каталоге `jsconnector/conf`. Здесь описан каждый из параметров, сохраняемых в файле конфигурации jsconnector.xml.

Параметры и их описания

Следующие параметры сохраняются в файле `jsconnector.xml` и используется для взаимодействий веб-соединителя. Существует два способа измените эти параметры:

- Использование веб-страницы конфигурации веб-соединителя, автоматически становящейся доступной после внедрения и запуска прикладной программы веб-соединителя. Для использования этой веб-страницы конфигурации раскройте при помощи браузера URL, подобный следующему: `http://<хост>:<порт>/interact/jsp/WebConnector.jsp`.

  Изменения, вносимые вами на веб-странице администрирования, сохраняются в файле `jsconnector.xml` на сервере, где внедряется веб-соединитель.
- Отредактируйте файл `jsconnector.xml` непосредственно при помощи любого текстового редактора или редактора XML. Перед использованием этого метода убедитесь, что вы готовы редактировать теги и значения XML.

  Примечание: Каждый раз, когда вы редактируете файл `jsconnector.xml` вручную, можно перезагружать эти параметры, открыв страницу администрирования веб-соединителя (находящуюся по адресу `http://<хост>:<порт>/interact/jsp/jsconnector.jsp`) и нажав кнопку **Перезагрузить конфигурацию**.

В следующей таблице описаны опции конфигурации, которые можно задать, так как они содержатся в файле `jsconnector.xml`.

Таблица 34. Опции конфигурации веб-соединителя

Группа параметров	Параметр	Описание
`defaultPageBehavior`		
	`friendlyName`	Удобочитаемый идентификатор паттерна URL для вывода на странице конфигурации веб-соединителя.
	`interactURL`	Базовый URL сервера среды выполнения Interact. Примечание: Этот параметр нужно задать, только если служба веб-соединителя (jsconnector) запускается как внедрённая веб-программа. Если веб-соединитель запускается автоматически в составе сервера среды выполнения Interact, задавать этот параметр не требуется.
	`jsConnectorURL`	Базовый URL, используемый для генерирования URL перехода, такого как `http://хост:порт/jsconnector/clickThru`
	`interactiveChannel`	Имя интерактивного канала, представляющего это отображение страницы.
	`sessionIdCookie`	Имя опознавательного файла (cookie), содержащего ID сеанса, который используется в вызовах API Interact.
	`visitorIdCookie`	Имя опознавательного файла (cookie), который будет содержать ID аудитории.
	`audienceLevel`	Уровень аудитории кампании для входящего посетителя, используемый в вызове API среды выполнения Interact.
	`audienceIdField`	Имя поля `audienceId`, используемое в вызове API среды выполнения Interact. **Примечание:** Примечание: В текущий момент идентификаторы аудитории для нескольких полей не поддерживаются.
	`audienceIdFieldType`	Тип данных поля ID аудитории [`numeric | string`], используемый в вызове API среды выполнения Interact
	`audienceLevelCookie`	Имя опознавательного файла (cookie), который должен содержать уровень аудитории. Это необязательный параметр. Если не задать этот параметр, система будет использовать значение, определённое для поля `audienceLevel`.
	`relyOnExistingSession`	Используется в вызове API среды выполнения Interact. Как правило, для этого параметра задаётся "true".
	`enableInteractAPIDebug`	Используется в вызове API среды выполнения Interact для включения поддержки вывода отладки в файлы журнала.

Таблица 34. Опции конфигурации веб-соединителя (продолжение)

Группа параметров	Параметр	Описание
	pageLoadEvents	Событие, размещаемое после загрузки этой конкретной страницы. Задайте в этом теге одно или несколько событий в формате, подобном следующему: <event>событие1</event>..
	interactionPointValues	Все элементы в этой категории действуют как значения по умолчанию для отсутствующих значений в категориях, относящихся к конкретным точкам взаимодействия.
	interactionPointValuescontactEvent	Имя по умолчанию для события контакта, размещаемого для этой конкретной точки взаимодействия.
	interactionPointValuesacceptEvent	Имя по умолчанию для события принятия, размещаемого для этой конкретной точки взаимодействия.
	interactionPointValuesrejectEvent	Имя по умолчанию для события отклонения, размещаемого для этой конкретной точки взаимодействия. (Примечание: на данный момент эта возможность не используется.)
	interactionPointValueshtmlSnippet	Имя по умолчанию шаблона HTML, предоставляемого для этой точки взаимодействия.
	interactionPointValuesmaxNumberOfOffers	Максимальное число предложений по умолчанию, получаемых из Interact для этой точки взаимодействия.
	interactionPointValueshtmlElementId	Имя по умолчанию элемента HTML, получающего содержимое для этой точки взаимодействия.
	interactionPoints	Эта категория содержит конфигурацию для каждой точки взаимодействия. Для любых недостающих свойств система будет использовать значения, сконфигурированные в категории interactionPointValues.
	interactionPointname	Имя точки взаимодействия.
	interactionPointcontactEvent	Имя события контакта, размещаемого для этой конкретной точки взаимодействия.
	interactionPointacceptEvent	Имя события принятия, размещаемого для этой конкретной точки взаимодействия.
	interactionPointrejectEvent	Имя события отклонения, размещаемого для этой конкретной точки взаимодействия. (Имейте в виду, что эти возможности еще не используется.)
	interactionPointhtmlSnippet	Имя шаблона HTML, предоставляемого для этой точки взаимодействия.
	interactionPointmaxNumberOfOffers	Максимальное число предложений, получаемых из Interact для этой точки взаимодействия.
	interactionPointhtmlElementId	Имя элемента HTML, получающего содержимое для этой точки взаимодействия.

Таблица 34. Опции конфигурации веб-соединителя (продолжение)

Группа параметров	Параметр	Описание
	enableDebugMode	Логический флаг (принимаемые значения: true или false) для включения особого режима отладки. При задании для этого флага значения true в содержимое, возвращаемое из веб-соединителя, включается вызов JavaScript 'alert', оповещающий клиента об отображении конкретной страницы, которое только что имело место. Для генерирования оповещения у клиента должна быть запись в файле authorizedDebugClients.
	authorizedDebugClients	Используемый особым режимом отладки файл, содержащий список имён хостов или IP-адресов, соответствующих критериям режима отладки.
	enableRawDataReturn	Логический флаг (принимаемые значения: true или false), определяющий, будет ли веб-соединитель присоединять необработанные данные предложений в формате JSON в самый конец содержимого.
	enableNetInsightTagging	Логический флаг (принимаемые значения: true или false), определяющий, будет ли веб-соединитель присоединять Digital Analytics for On Premises в самый конец содержимого.
	apiSequence	Представляет реализацию интерфейса APISequence, указывающего последовательность вызовов API, которые генерируются веб-соединителем при вызове pageTag. По умолчанию эта реализация использует последовательность startSession, pageLoadEvents, getOffers и logContact, где два последние вызова - конкретные для каждой точки взаимодействия.
	clickThruApiSequence	Представляет реализацию интерфейса APISequence, указывающего последовательность вызовов API, генерируемых веб-соединителем при вызове clickThru. По умолчанию эта реализация использует последовательность StartSession и logAccept.
	netInsightTag	Представляет шаблон HTML и JavaScript, используемый для интеграции вызова тега Digital Analytics for On Premises. Как правило, изменять эту опцию не нужно.

Использование страницы администрирования веб-соединителя

Веб-соединитель содержит страницу администрирования, предоставляющую некоторые инструменты, которые помогают управлять конфигурацией и проверить

её на возможность использования с конкретными паттернами URL. Страницу Администрирование можно также использовать для перезагрузки изменённой вами конфигурации.

О странице администрирования

С помощью любого из поддерживаемых браузеров можно открыть URL `http://`*`хост`*`:`*`порт`*`/interact/jsp/jsconnector.jsp`, где *хост:порт* - имя хоста, где запускается веб-соединитель, и порт, на котором он осуществляет приём соединений (такой как `runtime.example.com:7001`).

Страницу Администрирование можно использовать любым из следующих способов:

Таблица 35. Опции страницы администрирования веб-соединителя

Опция	Назначение
Перезагрузить конфигурацию	Щёлкните по ссылке **Перезагрузить конфигурацию**, чтобы перезагрузить все изменения конфигурации, которые были сохранены на диске, в память. Это необходимо сделать, если изменения были внесены непосредственно в файл конфигурации веб-соединителя jsconnector.xml, а не с помощью веб-страниц конфигурации.
Просмотреть конфигурацию	Просмотрите конфигурацию веб-соединителя на основе паттерна URL, вводимого вами в поле **Просмотреть конфигурацию**. При вводе URL страницы и щелчке по полю **Просмотреть конфигурацию** веб-соединитель возвращает конфигурацию, которую система будет использовать на основе отображения этого паттерна. Если найти соответствие не удается, возвращается конфигурация по умолчанию. Это помогает проверить, правильная ли конфигурация используется для конкретной страницы.
Выполнить тег страницы	Заполнение полей на этой странице и нажатие кнопки **Выполнить тег страницы** приводит к возврату веб-соединителем результата `pageTag` на основе паттерна URL. При этом симулируется вызов тега страницы. Различие между вызовом pageTag из этого инструмента и использованием реального веб-сайта состоит в том, что при использовании этой страницы администрирования выводятся все произошедшие ошибки и исключительные ситуации. Для реального веб-сайта исключительные ситуации не возвращаются, и их можно посмотреть только в файле журнала веб-соединителя.

Пример страницы веб-соединителя

В качестве примера в веб-соединитель Interact (в каталоге `<Interact_Home/jsconnector/webapp/html`) включен файл с именем `WebConnectorTestPageSA.html`, показывающий, сколько возможностей веб-соединителя можно задать как теги на странице. Для удобства эта страница примера показана также здесь.

Пример страницы HTML веб-соединителя

```
<?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[
/* ###############################################################
Это тестовая страница, содержащая pageTag WebConnector.
```

```
Поскольку в имя этого файла включено TestPage, веб-соединитель обнаружит,
что паттерн URL соответствует паттерну URL "testpage" в версии по умолчанию
jsconnector.xml - и здесь будет применено отображение определения конфигурации
на этот паттерн URL "testpage". Это означает, что данной странице надо дать
ID соответствующих элементов HTML, которые отвечают точкам взаимодействия для этого
паттерна URL (например, 'welcomebanner', 'crosssellcarousel' и 'textservicemessage')
###################################################################### */

/* ######################################################################
Этот раздел задает опознавательные файлы (cookies) для sessionId и visitorId.
Заметим, что на реальном производственном сайте это скорее всего делается
компонентом входа в систему.
Но ради тестирования это делается здесь...  имя опознавательного файла (cookie)
должно соответствовать сконфигурированному в XML jsconnector.
###################################################################### */
     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");

   /* ####################################################
   Теперь зададим ID элементов html, соответствующих этим IP
   #################################################### */
     document.writeln("<div id='welcomebanner'> Это надо изменить, "
  + "иначе что-то не так <\/div>");
     document.writeln("<div id='crosssellcarousel'> Это надо изменить, "
  + "иначе что-то не так <\/div>");
     document.writeln("<div id='textservicemessage'> Это надо изменить, "
  + "иначе что-то не так <\/div>");
   //]]&gt;
</script><!--
#############################################################################
это текст, который вставлен из файла pageTag.txt в каталоге conf
установки веб-соединителя. Переменную unicaWebConnectorBaseURL надо
подправить в соответствии с вашей локальной средой веб-соединителя
#############################################################################
-->
<!-- BEGIN: тег страницы веб-соединителя IBM Interact -->
<!--
# ****************************************************************************
#  Лицензированные материалы - собственность 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>Пример страницы веб-соединителя Interact</title>
   </head>
   <body>
   <!-- END: тег страницы веб-соединителя IBM Interact -->
   <!--
################################################################################
Конец вставки pageTag
```

```
################################################################################
-->
   </body>
</html>
```

Глава 16. Интеграция Interact и Digital Recommendations

IBM Interact можно интегрировать с IBM Digital Recommendations, чтобы создавать рекомендации по продуктам под управлением Interact. Оба эти продукта могут создавать рекомендации по продуктам для предложений, но используют разные методы. Digital Recommendations строит корреляции между посетителями и рекомендуемыми предложениями, используя веб-поведение посетителя (фильтр совместной работы). Interact учитывает поведение покупателя в прошлом, атрибуты, хронологию и в меньшей степени - предложения уровня просмотров, изучая, какие предложения лучше соответствуют профилю поведения покупателя (с учетом демографических и других сведений о покупателе). Коэффициенты принятия предложения помогают построить прогностическую модель путем самообучения. Пользуясь преимуществами обоих продуктов, можно при помощи Interact определить предложения с учетом персонального профиля и передать ID категории в Digital Recommendations, чтобы получить рекомендуемые продукты с учетом популярности ("коллективный разум") и вывести их для покупателя в числе выбранных предложений. Так можно улучшить рекомендации для покупателя, повысить число результативных щелчков и улучшить итоговые результаты по сравнению с использованием только одного из продуктов.

В следующих разделах описывается, как работает эта интеграция и как использовать предлагаемый пример прикладной программы, чтобы создать свою пользовательскую интеграцию предложения.

Обзор интеграции Interact с Digital Recommendations

В этом разделе описывается, как интегрировать IBM Interact с IBM Digital Recommendations, чтобы получать рекомендации по продуктам под управлением Interact, в частности, описывается процедура и механизмы такой интеграции.

IBM Interact интегрируется с IBM Digital Recommendations через API (application programming interface, интерфейс прикладного программирования) стандарта REST (Representational state transfer), доступный после установки Digital Recommendations. При помощи вызовов API REST с указанием ID категории Interact может получать рекомендуемые товары и включать их в информацию предложения, которое выводится на страницу, просматриваемую посетителем.

Когда посетитель просматривает URL веб-страницы (например, пример страницы JSP, который входит в установку Interact), страница вызывает Interact и запрашивает предложение. Если предложение было сконфигурировано в Interact с правильными параметрами, далее, в простейшем случае, выполняются такие шаги:

1. Логика страницы определяет ID покупателя для посетителя.
2. API отправляет вызов Interact, передавая необходимую информацию, чтобы сгенерировать предложение для этого покупателя.
3. Возвращенное предложение содержит веб-страницу, на которой есть как минимум три атрибута: URL изображения для предложения, URL целевой страницы, открываемой щелчком, и ID категории, который учитывается при выборе рекомендуемых продуктов.
4. Затем ID категории передается в вызове Digital Recommendations, чтобы получить рекомендуемые продукты. Этот набор продуктов представлен в формате JSON (JavaScript Object Notation) и упорядочен начиная с самых популярных продуктов в этой категории.

5. Затем предложение и продукты выводятся в браузере посетителя.

Такая интеграция позволяет сочетать рекомендацию предложения с рекомендациями продуктов. Например, на одной веб-странице можно предусмотреть две точки взаимодействия: одну для предложения, другую для рекомендаций, соответствующих этому предложению. Для достижения такого результата веб-страница вызывает Interact, чтобы выполнить сегментацию в реальном времени и определить лучшее предложение (например, 10% скидки на все бытовые электроприборы). Когда страница получает предложение от Interact, оно должно содержать ID категории (в данном примере - категория бытовых электроприборов). Затем страница передает ID категории бытовых электроприборов Digital Recommendations при помощи вызова API и в ответ получает рекомендации лучших продуктов для этой категории с учетом их популярности.

В более простом примере веб-страница могла бы вызывать только Interact, чтобы найти категорию (например, элитные столовые приборы), соответствующую профилю покупателя. Затем она передавала бы полученный ID категории Digital Recommendations и получала рекомендации по товарам - столовым приборам.

Предварительные требования интеграции

Чтобы использовать интеграцию Digital Recommendations - Interact, сначала нужно убедиться, что выполнены предварительные требования, описанные в этом разделе.

Убедитесь, что выполняются следующие предварительные требования:

- Вы знакомы с использованием API Interact, описанным в других разделах *Руководства администратора* и в электронной справке.
- Вы знакомы с API Digital Recommendations REST, описанным в документации разработчика Digital Recommendations.
- Вы понимаете форматы HTML, JavaScript, CSS и JSON (JavaScript Object Notation).

  Формат JSON важен, потому что в нем возвращает затребованные данные о продукте API Digital Recommendations REST.
- Вы знакомы с программированием веб-страниц на стороне сервера, поскольку пример прикладной программы в составе Interact пользуется JSP (хотя использовать JSP не обязательно).
- У вас есть допустимая учетная запись Digital Recommendations и список ID категорий, в которых планируется получать рекомендации Interact (самые популярные продукты в указанной категории).
- У вас есть ссылка на API Digital Recommendations REST (URL для вашей среды Digital Recommendations).

  Дополнительную информацию смотрите в примере прикладной программы в установке Interact или в примере кода в разделе “Использование примера проекта интеграции” на стр. 340.

Конфигурирование предложение для интеграции Digital Recommendations

Чтобы ваша веб-страница могла вызвать Digital Analytics Digital Recommendations для получения рекомендуемого продукта, сначала нужно сконфигурировать предложение IBM Interact, задав информацию для передачи Digital Recommendations.

Об этой задаче

Чтобы сконфигурировать предложение, связанное с Digital Recommendations, сначала убедитесь, что соблюдены следующие условия:

- Убедитесь, что сервер среды выполнения Interact сконфигурирован и работает без ошибок.
- Убедитесь, что сервер среды выполнения может установить соединение с сервером Digital Recommendations. В частности, убедитесь, что ваш брандмауэр разрешает установить обычное исходящее веб-соединение (порт 80).

Чтобы сконфигурировать предложение для интеграции с Digital Recommendations, выполните следующие действия.

Процедура

1. Создайте или измените предложение для Interact.

   Информацию о создании и изменении предложений смотрите в *Руководстве пользователя IBM Interact* и в документации по IBM Campaign.
2. Убедитесь, что предложение содержит, помимо иных параметров, следующие атрибуты:
   - URL (uniform resource locator, унифицированный указатель ресурсов) изображения для этого предложения.
   - URL целевой страницы для этого предложения.
   - ID категории Digital Recommendations, связанной с этим предложением.

     ID категории можно получить вручную из конфигурации Digital Recommendations. Interact не умеет получать значения ID категории непосредственно.

   В демонстрационной веб-программе, входящей в установку Interact, эти атрибуты предложения называются `ImageURL`, `ClickThruURL` и `CategoryID`. Вы можете использовать любые содержательные с вашей точки зрения имена, лишь бы ваша веб-программа сопоставляла значения с теми, которые ожидает предложение.

   Например, пусть вы определили предложение с именем "10PercentOff", содержащее приведенные ниже атрибуты, где ID категории (полученный вами из конфигурации Digital Recommendations) - `PROD1161127`, URL щелчка предложения - `http://www.example.com/success`, а URL изображения выводимого для предложения, - `http://localhost:7001/sampleIO/img/10PercentOffer.jpg` (в данном случае это URL, локальный для сервера времени выполнения Interact).
3. Определите правила процедур для интерактивного канала, включив в него это предложение, и внедрите интерактивный канал, как обычно.

Результаты

Теперь предложение определено, и при этом сконфигурирована необходимая информация для интеграции Digital Recommendations. Чтобы дать возможность Digital Recommendations рекомендовать продукты Interact, осталось сконфигурировать веб-страницы, задав соответствующие вызовы API.

Конфигурируя веб-программу, предлагающую посетителям интегрированную страницу, убедитесь в наличии следующих файлов в каталоге `WEB-INF/lib`:

- *`домашняя_страница_Interact`*`/lib/interact_client.jar` - требуется для обработки вызовов API Interact с веб-страницы.

- *домашняя_страница_Interact*/lib/JSON4J_Apache.jar - требуется для обработки данных, возвращенных вызовом API REST Digital Recommendations. Эти данные возвращаются в формате JSON.

Дополнительную информацию о том, как представлять предложения покупателям, смотрите в разделе “Использование примера проекта интеграции”.

Использование примера проекта интеграции

В каждой установке среды выполнения Interact есть пример проекта, демонстрирующий процесс интеграции Digital Recommendations - Interact. Пример проекта демонстрирует от начала до конца, как создать веб-страницу, которая будет вызывать предложение, содержащее ID категории, и передавать его Digital Recommendations, чтобы получить список рекомендованных продуктов и представить его на странице в точках взаимодействия.

Обзор

Предлагаемый пример проекта можно использовать как есть, если нужно протестировать процесс интеграции, или как начальную точку для разработки пользовательских страниц. Пример проекта содержится в следующем файле:

домашний_каталог_Interact/samples/IntelligentOfferIntegration/MySampleStore.jsp

Этот файл содержит, наряду с полном работоспособным примером процесса интеграции, всесторонние комментарии, в которых объясняется, что нужно сконфигурировать в Interact, что нужно настроить в файле .jsp и как правильно внедрить страницу, чтобы она работала с вашей установкой.

MySampleStore.jsp

Для удобства здесь показан файл MySampleStore.jsp. Поскольку этот пример может быть изменен в последующих выпусках Interact, используйте как пример и начальную точку тот файл, который включен в состав вашей установки.

```
<!--
# ****************************************************************************
#  Лицензированные материалы - собственность 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" %>

<%

 /****************************************************************************
 * Этот пример программы jsp демонстрирует интеграцию Interact и Digital Recommendations.
 *
 * Когда URL этой программы jsp запрашивается в браузере, алгоритм вызовет Interact
 * для выборки предложения. На основе ID категории, связанного с предложением, алгоритм
```

```
* вызовет Digital Recommendations, чтобы выполнить выборку рекомендуемых товаров.
* Будут показаны предложение и товары.
* Чтобы переключиться на другой ID покупателя и продемонстрировать другие предложения, можно просто
* добавить cid=<id> в конец URL этой программы JSP.
*
* Предварительные требования для знакомства с этой демонстрационной программой:
* 1) знакомство с Interact и его API Java
* 2) знакомство с IntelligentOffer и его API Rest
* 3) знакомство с разметкой веб-страницы (форматы html, css, javascript)
* 4) технология для генерирования веб-страницы (в этой демонстрационной программе используется выполнение JSP на стороне сервера)
*
*
* Шаги, после которых эта демонстрационная программа заработает:
* 1) сконфигурируйте среду выполнения Interact, которая может делать предложения, задав следующие
* атрибуты предложения:
*  ImageURL : url изображения для данного предложения
*  ClickThruURL : url целевой страницы данного предложения
*  CategoryID : ID категории Digital Recommendations, связанной с данным предложением
* ПРИМЕЧАНИЕ: для атрибутов можно использовать альтернативные имена, если ссылки на эти
* атрибуты в этой программе jsp отредактировать соответственно.
* 2) Получите допустимый URL API REST среды Intelligent Offer
* 3) Встройте эту программу JSP в веб-программу Java
* 4) Убедитесь, что файл interact_client.jar находится в каталоге WEB-INF/lib (для связи с Interact)
* 5) Убедитесь, что файл JSON4J_Apache.jar (из установки Interact) находится в
 *    каталоге WEB-INF/lib (для связи с Intelligent Offer)
* 6) задайте свойства конкретной среды в следующих двух разделах
*******************************************************************************/

/*******************************************************************************
*   ****************ИЗМЕНИТЕ ЭТИ ПАРАМЕТРЫ ДЛЯ ВАШЕЙ СРЕДЫ**********************
* Задайте здесь свойства, специфичные для среды Interact...
*******************************************************************************/

final String sessionId="123";
final String interactiveChannel = "SampleIO";
final String audienceLevel = "Customer";
final String audienceColumnName="CustomerID";
final String ip="ip1";
int customerId=1;
final String interactURL="http://localhost:7011/interact/servlet/InteractJSService";
final boolean debug=true;
final boolean relyOnExistingSession=true;

/*******************************************************************************
  ****************ИЗМЕНИТЕ ЭТИ ПАРАМЕТРЫ ДЛЯ ВАШЕЙ СРЕДЫ**********************
* Задайте здесь свойства, специфичные для вашей среды Digital Recommendations...
*******************************************************************************/

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();

// получить customerID при передаче как параметра
String cid = request.getParameter("cid");
if(cid != null)
{
 customerId = Integer.parseInt(cid);
}

// Вызвать Interact для получения предложения
Offer offer=getInteractOffer(interactURL,sessionId,interactiveChannel,audienceLevel,
       audienceColumnName,ip,customerId,debug,relyOnExistingSession,interactErrorMsg);

// получить определенные атрибуты из предложения (url изображения, сквозной url и ID категории)
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("ID_категории"))
        {
         categoryId=offerAttribute.getValueAsString();
        }
    }
}

// вызов Digital Recommendations для получения товаров
JSONObject products=getProductsFromIntelligentOffer(ioURL, cID, zoneID, categoryId,
       intelligentOfferErrorMsg);

%>

<html>
 <head>
  <title>Мой любимый магазин</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>Добро пожаловать в наш магазин</b> <%=customerId %>
```

```
  <br><br>
<% if(offer != null) { %>
 <!-- HTML предложения Interact -->

 <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 { %>
 Нет доступных предложений. <br> <br>
 <%=interactErrorMsg.toString() %>
<% } %>

<% if(products != null) { %>
 <!-- HTML товара IntelligentOffer -->
 <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>
 Нет доступных товаров...<br> <br>
 <%=intelligentOfferErrorMsg.toString() %>
 </div>
<% } %>

 </body>
</html>



<%!
 /*****************************************************************************
 * Ниже приведены две удобные функции, которые запрашивают Interact и
 *   Digital Recommendations
 *****************************************************************************/

 /*****************************************************************************
 * Вызов Digital Recommendations для получения рекомендованных продуктов
 *****************************************************************************/
 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;

}

/*****************************************************************************
* Вызов Interact для получения предложения
*****************************************************************************/
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 };

      // вызов startSession
     Response response = api.startSession(sessionId, relyOnExistingSession,
         debug, interactiveChannel, audienceId, audienceLevel, null);

     if(response.getStatusCode() == Response.STATUS_ERROR)
     {
      printDetailMessageOfWarningOrError("startSession",response, interactErrorMsg);
  }

     // вызов 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("Вызывается "+command).append("<br>");
        AdvisoryMessage[] messages = response.getAdvisoryMessages();

        for(AdvisoryMessage msg : messages)
        {
         sb.append(msg.getMessage()).append(":");
         sb.append(msg.getDetailMessage());
         sb.append("<br>");
        }
        interactErrorMsg.append(sb.toString());
    }
%>
```

Глава 17. Интеграция Interact и Digital Data Exchange

При помощи Digital Data Exchange ваш сайт может ссылаться на Interact, где предлагается мощный всеканальный исполнительный механизм, который предоставляет лучшие предложения по оптимальным каналам и может развиваться (обучаться) благодаря обратной связи, постоянно повышая свою маркетинговую эффективность.

Этим инструментом можно воспользоваться, если ваша маркетинговая команда использует Interact для всеканального управления предложениями и хочет распространить такие персонализированные предложения на ваши сайты.

IBM Digital Data Exchange интегрирует IBM и маркетинговые решения других производителей с опытом в сфере электронных покупок посредством API синдикации данных в реальном времени и решения управления тегами промышленного уровня.

Без IBM Digital Data Exchange маркетологи поручают информационным технологам связь Interact со своим сайтом и вызов API Interact с различных веб-страниц. При использовании IBM Digital Data Exchange маркетологи могут обойтись без информационных технологов и непосредственно перейти в IBM Digital Data Exchange, чтобы добавить теги IBM Digital Data Exchange на различные веб-страницы.

Требования

Чтобы использовать интеграцию Interact и Digital Data Exchange, сначала нужно убедиться, что выполнены предварительные требования, описанные в этом разделе.

Убедитесь, что выполняются следующие предварительные требования.

- Вы ознакомились с API Interact, как описано в других разделах Руководства администратора и в электронной справке.
- Вы ознакомились с тегами Digital Data Exchange и группами страниц.
- У вас есть допустимая учетная запись Digital Data Exchange.
- Ваш файл `interactapi.js` размещен на общедоступном хосте и доступен в разделе **Настройки поставщиков** .

Интеграция IBM Interact с вашим сайтом при помощи IBM Digital Data Exchange

Выполните эти действия, чтобы интегрировать Interact с вашим сайтом при помощи Digital Data Exchange.

Процедура

1. Задайте положение файла `Interactapi.js`.
   a. Перейдите к **Поставщики > Настройки поставщиков** в Digital Data Exchange.
   b. Выберите IBM Interact в выпадающем меню **Поставщик**.
   c. В поле **Путь библиотеки** введите URL, где вы разместили `Interactapi.js`. В этом URL не указывайте протокол (http или https).
   d. В поле **Путь к общедоступному сервлету Rest** добавьте путь к сервлету Rest.

2. Перейдите к **Управление > Глобальные параметры** в Digital Data Exchange и задайте имя объекта, которое будет служить идентификатором страницы, в поле **Уникальный идентификатор страницы**. Например, как имя объекта можно задать `digitalData.pageInstanceID`.
3. Добавьте файл `eluminate.js` и идентификатор на веб-страницу, где нужно вставить теги при помощи Digital Data Exchange. Каждой веб-странице нужно назначить уникальный идентификатор, чтобы Digital Data Exchange мог различать разные страницы.

   Например, можно добавить следующий сценарий на домашнюю страницу.

```
<!-- Конфигурирование идентификатора страницы -->
          <script>
                  digitalData={pageInstanceID:"INTERACT_HomePage"};
          </script>

<!-- Добавление сценария eluminate -->
         <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. В Digital Data Exchange создайте теги, сегменты кода, функции и другие элементы, которые нужно добавить на веб-страницу.
5. Создайте группы страниц, чтобы определить значения, которые нужно внести на каждой странице.

   Смотрите дополнительную информацию в IBM Digital Data Exchange Руководстве пользователя.

Теги Interact в Digital Data Exchange

Используйте теги Digital Data Exchange по умолчанию, чтобы определить различные варианты тегов для веб-страниц, на которых представлены данные из разных мест. Теги, определенные вами один раз, добавляются в список тегов Interact. У тегов может не быть определяемых полей и может не быть обязательных полей; теги можно использовать непосредственно.

Следующие теги Interact доступны в Digital Data Exchange в разделе **Теги**.

- Завершить сеанс
- Получить предложения
- Загрузить библиотеку
- Разместить событие
- Задать аудиторию
- Запустить сеанс

Чтобы использовать теги Interact, отредактируйте эти теги, определив для каждого тега Interact Поле тега, Метод, Имя объекта, Тип данных и Модификатор.

В тегах Разместить событие, Задать аудиторию и Запустить сеанс допустимы пользовательские поля. Чтобы определить пользовательский параметр, щелкните по значку Добавить поле тега, затем по значку Редактировать. Процедура аналогична определению любого параметра, однако имя параметра можно редактировать и оно должно включать в себя название параметра, двоеточие и тип данных параметра. Положение пользовательского параметра в теге можно изменять стрелками вверх и вниз.

Кроме того, теги можно связывать с функциями JavaScript или объектами HTML, чтобы инициировать после инициации функции или после события объекта HTML.

Дополнительную информацию о том, как определять теги, привязывать их и работать с ними, смотрите в Руководстве пользователя IBM Digital Data Exchange.

Конкретные варианты использования интеграции Interact и Digital Data Exchange смотрите в разделе https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Завершить сеанс

Теги завершения сеанса отмечают окончание веб-сеанса.

Для тега завершения сеанса доступны следующие поля.

Таблица 36. Теги завершения сеанса

Поле тега	Описание
*ID сеанса	Указывает ID сеанса.
Имя функции обратного вызова при успехе	Определяет имя функции, которая вызывается в случае успешного выполнения метода завершения сеанса.
Имя функции обратного вызова при неудаче	Определяет имя функции, которая вызывается в случае неудачного выполнения метода завершения сеанса.

Любое **Поле метки**, отмеченное * является обязательным.

Получить предложения

Тег получения предложений служит для затребования предложений от сервера времени выполнения.

Для тега получения предложений доступны следующие поля.

Таблица 37. Теги получения предложений

Поле тега	Описание
*ID сеанса	Указывает ID сеанса.
*Имя точки взаимодействия	Указывает имя точки взаимодействия, на которую ссылается этот метод. Это имя должно точно соответствовать имени точки взаимодействия, определенной в интерактивном канале.
*Затребованное количество	Указывает число затребованных предложений.
Имя функции обратного вызова при успехе	Определяет имя функции, которая вызывается в случае успешного выполнения метода получения предложений.
Имя функции обратного вызова при неудаче	Определяет имя функции, которая вызывается в случае неудачного выполнения метода получения предложений.

Любое **Поле метки**, отмеченное * является обязательным.

Тег получения предложений надо назначить в группу страниц, для которой задан контейнер `Default`.

Загрузить библиотеку

Тег Загрузить библиотеку загружает библиотеку Interact JavaScript в разделе заголовка страницы.

У тега Загрузить библиотеку нет параметров. Положение библиотеки он берет из поля Путь библиотеки в разделе **Настройки поставщиков**. Его следует включить в группу страниц, задав в контейнере значение Заголовок, и выполнять на каждой странице с тегами Interact.

Важное замечание: Ни один другой тег не будет работать, если не добавлен тег загрузки библиотеки. Если этот тег не добавлен, файл interact.js не загружается.

Разместить событие

Используйте тег размещения события для выполнения любого события, определенного в интерактивном канале.

Для тега размещения события доступны следующие поля.

Таблица 38. Теги размещения события

Поле тега	Описание
*ID сеанса	Указывает ID сеанса.
*Имя события	Указывает имя события. Имя события должно соответствовать имени события, как определено в интерактивном канале. Регистр символов в этом имени не учитывается.
Имя функции обратного вызова при успехе	Определяет имя функции, которая вызывается в случае успешного выполнения метода после события.
Имя функции обратного вызова при неудаче	Определяет имя функции, которая вызывается в случае неудачного выполнения метода после события.

Любое **Поле метки**, отмеченное * является обязательным.

Необязательные параметры можно добавить с помощью пользовательской функции поля тега. Пользовательские имена тегов должны состоять из названия параметра, двоеточия и типа данных.

Задать аудиторию

Используйте тег задания аудитории для задания ID и уровня аудитории для посетителя.

Для тега задания аудитории доступны следующие поля.

Таблица 39. Теги задания аудитории

Поле тега	Описание
*ID сеанса	Указывает ID сеанса.
*ID аудитории	Указывает ID аудитории. Эти имена должны соответствовать именам физических столбцов любой таблицы, содержащей ID аудитории. ID аудитории не может содержать более 17 значащих цифр. Если ID аудитории содержит более 17 значащих цифр, его надо разбить на несколько или же изменить тип ID аудитории на строковый.

Таблица 39. Теги задания аудитории (продолжение)

Поле тега	Описание
*Уровень аудитории	Определяет уровень аудитории.
Имя функции обратного вызова при успехе	Определяет имя функции, которая вызывается в случае успешного выполнения метода задания аудитории.
Имя функции обратного вызова при неудаче	Определяет имя функции, которая вызывается в случае неудачного выполнения метода задания аудитории.

Любое **Поле метки**, отмеченное * является обязательным.

Необязательные параметры можно добавить с помощью пользовательской функции поля тега. Пользовательские имена тегов должны состоять из названия параметра, двоеточия и типа данных.

Запустить сеанс

Тег начала сеанса создает и определяет веб-сеанс.

Для тега начала сеанса доступны следующие поля.

Таблица 40. Теги начала сеанса

Поле тега	Описание
*ID сеанса	Указывает ID сеанса.
*Канал взаимодействия	Определяет имя интерактивного канала, к которому относится данный сеанс. Это имя должно точно соответствовать имени интерактивного канала, определенного в Campaign.
*ID аудитории	Указывает ID аудитории. Эти имена должны совпадать с именами физических столбцов любой таблицы, содержащей ID аудитории.
*Уровень аудитории	Определяет уровень аудитории.
*Связь с существующим сеансом	Определяет, использует ли данный сеанс новый или же существующий сеанс
*Отладка	Включает или отключает отладочную информацию.
Имя функции обратного вызова при успехе	Определяет имя функции, которая вызывается в случае успешного выполнения метода запуска сеанса.
Имя функции обратного вызова при неудаче	Определяет имя функции, которая вызывается в случае неудачного выполнения метода запуска сеанса.

Любое **Поле метки**, отмеченное * является обязательным.

Необязательные параметры можно добавить с помощью пользовательской функции поля тега. Пользовательские имена тегов должны состоять из названия параметра, двоеточия и типа данных.

Тег начала сеанса надо назначить в группу страниц, для которой задан контейнер `Default`.

Пример конфигурирования тегов

В этом примере показана простое конфигурирование тегов начала сеанса, размещения события, получения предложений и завершения сеанса.

Для любого тега можно получить значения поля тега из опознавательного файла (cookie) при помощи метода cookie или из объекта JavaScript при помощи метода javascriptobject.

Эти теги поддерживают дополнительные параметры, не показанные в настоящем простом примере. Информацию о дополнительных параметрах можно найти в Руководстве пользователя IBM Digital Data Exchange.

Конкретные варианты использования интеграции Interact и Digital Data Exchange смотрите в разделе https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Пример конфигурирования тега начала сеанса

Выберите **Теги > Теги IBM > IBM Interact > Тип: Начало сеанса**, чтобы создать тег начала сеанса. Отредактируйте тег со следующими параметрами.

Параметры ID сеанса

- **Метод**: `Constant`
- **Constant**: `5555`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Параметры интерактивного канала

- **Метод**: `Constant`
- **Constant**: `WSCDemo`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Параметры ID аудитории

- **Метод**: `Constant`
- **Константа**: `USERS_ID,2002,numeric`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Параметры уровня аудитории

- **Метод**: `Constant`
- **Constant**: `WSCUserId`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Использовать параметры существующего сеанса

- **Метод**: `Constant`
- **Constant**: `False`
- **Тип данных**: `Boolean`
- **Модификатор**: `<null>`

Отладка

- **Метод**: `Constant`

- **Constant**: `True`
- **Тип данных**: `Boolean`
- **Модификатор**: `<null>`

Параметры имени функции обратного вызова при успехе

- **Метод**: `Unassigned`
- **Значение**: `<null>`

Параметры имени функции обратного вызова при неудаче

- **Метод**: `Unassigned`
- **Значение**: `<null>`

Пример конфигурирования тега получения предложений

Выберите **Теги > Теги IBM > IBM Interact > Тип: получение предложений**, чтобы создать тег получения предложений. Отредактируйте тег со следующими параметрами.

Параметры ID сеанса

- **Метод**: `Constant`
- **Constant**: `5555`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Конфигурирование имени точки Interact

- **Метод**: `Constant`
- **Constant**: `AuroraHomepageHeaderBannerLeft`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Конфигурирование требуемого числа

- **Метод**: `Constant`
- **Constant**: `1`
- **Тип данных**: `integer`
- **Модификатор**: `<null>`

Параметры имени функции обратного вызова при успехе

- **Метод**: `Constant`
- **Constant**: `onOfferReturnSuccess`
- **Тип данных**: `string`
- **Модификатор**: `<null>`

Параметры имени функции обратного вызова при неудаче

- **Метод**: `Constant`
- **Constant**: `onOfferReturnError`
- **Тип данных**: `string`
- **Модификатор**: `<null>`

Пример конфигурирования тега размещения события

Выберите **Теги > Теги IBM > IBM Interact > Тип: размещения события**, чтобы создать тег размещения события. Отредактируйте тег со следующими параметрами.

Параметры ID сеанса

- **Метод**: `Constant`
- **Constant**: `5555`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Конфигурирование имени события

- **Метод**: `Constant`
- **Constant**: `ACCEPTOFFER`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Параметры имени функции обратного вызова при успехе

- **Метод**: `Constant`
- **Constant**: `onSuccessTestFunction`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Параметры имени функции обратного вызова при неудаче

- **Метод**: `Constant`
- **Constant**: `onErrorTestFunction`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Конфигурирование поля дополнительного параметра

- **Поле тега**: `UACIOfferTrackingCode:string`
- **Метод**: `JavaScriptObject`
- **Имя объекта**: `oa.treatmentCode`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Пример конфигурирования тега завершения сеанса

Выберите **Теги > Теги IBM > IBM Interact > Тип: Завершение сеанса**, чтобы создать тег завершения сеанса. Отредактируйте тег со следующими параметрами.

Параметры ID сеанса

- **Метод**: `Constant`
- **Constant**: `5555`
- **Тип данных**: `String`
- **Модификатор**: `<null>`

Параметры имени функции обратного вызова при успехе

- **Метод**: `Unassigned`

- **Значение**: `<null>`

Параметры имени функции обратного вызова при неудаче

- **Метод**: `Unassigned`
- **Значение**: `<null>`

Примеры функций

Чтобы использовать функции при задании значений On Success Callback Function Name и On Failure Callback Function Name, достаточно указать имя функции при создании нового тега, если эта функция уже есть на вашей веб-странице.

Кроме того, можно использовать утилиты Digital Data Exchange, чтобы создать функции и добавить их на ваши веб-страницы.

В следующем примере показано, как вывести предложение, возвращенное Interact, на вашей веб-странице. Нужно добавить этот сценарий на страницу или использовать для вставки фрагмент кода Digital Data Exchange.

```
<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>
```

Проверьте конфигурацию интеграции

Используйте тестовую утилиту Digital Data Exchange и файл `Interact.log` для отладки и устранения любых ошибок конфигурации.

При помощи тестовой утилиты Digital Data Exchange можно проверить энциклопедию и узнать, работает ли ваша конфигурация, как ожидалось. Чтобы открыть тестовую утилиту, выберите **Внедрение > Тестовая утилита** в Digital Data Exchange.

Дополнительную информацию о тестовой утилите смотрите в Руководстве пользователя IBM Digital Data Exchange.

Можно просмотреть файл `Interact.log` и найти подробности о различных сделанных вызовах API Interact. Для отладки различных вызовов добавьте в каждый тег Функцию обратного вызова при успехе и Функцию обратного вызова при неудаче.

Глава 18. Сконфигурируйте шлюзы для инициированных сообщений

Используйте шлюзы инициированных сообщений, чтобы отправлять и получать информацию о предложении по входящим и исходящим каналам.

Для инициированных сообщений можно использовать следующие входные и выходные шлюзы.

- IBM Interact Inbound Gateway для IBM Universal Behavior Exchange
- IBM Interact Outbound Gateway для IBM Universal Behavior Exchange
- IBM Interact Email (Transact) Outbound Gateway для IBM Marketing Cloud
- IBM Interact Outbound Gateway для IBM Mobile Push Notification

Дополнительную информацию смотрите в разделе https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20Triggered%20Messages.

Использование IBM Interact Inbound Gateway для IBM Universal Behavior Exchange

Чтобы использовать IBM Interact Inbound Gateway для IBM Universal Behavior Exchange, надо сконфигурировать Interact, сконфигурировать конечную точку подписчика UBX и создать конечную точку и событие в UBX.

Используйте следующие конфигурации в качестве примера для вашей конфигурации.

Шлюз подписчика можно скачать с 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.

Конфигурирование Interact для IBM Interact Inbound Gateway для IBM Universal Behavior Exchange

Используйте следующие действия для конфигурирования Interact.

1. Добавьте нового получателя в свойстве конфигурации **Interact | activityOrceshtrator | receivers**. В качестве **Типа** задайте `IBMMQ` или `Пользовательский`. Если вы выбрали `Пользовательский`, введите **ClassName** и **ClassPath**. Если вы выбрали `IBMMQ`, оставьте поля **ClassPath** и **ClassName** пустыми.
2. Добавьте параметры `providerURL`, `queueManager`, `messageQueueName`, `authDS` и `asmUserFor...."` для вашего получателя.
3. Добавьте новый шлюз в свойстве конфигурации **Interact | activityOrceshtrator | gateways**. Как **ClassPath** задайте URI положения файла `OMO_InteractGateway_UBX.jar` и как **ClassName** -

   ```
   com.ibm.interact.offerorchestration.inboundgateway.ubx.
   UBXInboundGateway
   ```

4. Создайте папку Interactubx11 в папке UBX выходного шлюза и скопируйте в эту новую папку файлы свойств. Имя папки должно соответствовать имени конечной точки подписчика, которую вы создали в UBX.
5. В файле interactEventNamemapping.properties добавьте запись, чтобы отобразить значение поля события полезной нагрузки в имя события Interact. Например, recommededOffers=recommendedOffers.
6. В файле interactEventPayloadMapping.properties добавьте определения полей, задав как имена полей соответственно OMO-conf_inbound_UBX_interactEventNameMapping и OMO-conf_inbound_UBX_interactEventNameMapping

   Например:

   ```
   [SessionID]=(String)interactprofileid
   [EventName]=(String)code
   [AudienceIDFieldNames]=(String)"CustomerID"
   [AudienceIDFieldValues]=(Numeric)interactprofileid
   [AudienceLevel]=(String)"Customer"
   [InteractChannel]=(String"UBX_MM"
   ```
7. Добавьте положения ваших файлов Interactubx11/interactEven tNameMapping.properties и Interactubx11/interactEven tPayloadMapping.properties как параметры шлюза в **Interact | activityOrceshtrator | gateways | [gatewayname] | Parameter Data**.
8. Создайте интерактивный канал и добавьте в него событие.
9. Добавьте правило инициированных сообщений с событием recommendedOffers и назначьте этому правилу предложение.
10. Внедрите интерактивный канал.
11. Перезапустите сервер Interact.
12. Отправьте событие в UBX при помощи клиента API REST.

    Пример тела события:

    ```
    {
      "channel" : "mobile",
      "identifiers" : [
        {
          "name" : "interactprofileid",
          "value" : "55"
        }
      ],
    "events" : [
      {
        "code" : "recommendedOffers",
        "timestamp" : "2015-12-28T20:16:12Z"
       }
      ]
    }
    ```
13. Посмотрите журнал Interact, чтобы проверить, инициировано ли событие инициированных сообщений.

Конфигурирование IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange

Это образец конечной точки, и его можно использовать как пример.

Кроме того, используйте инструкции и сконфигурируйте следующее.

- Конечная точка UBX с IBM MQ
- Файл ubxInboundEndpoint.properties конечной точки
- Файл inboundProducerNameConfig.properties конечной точки

- Файл inboundQueueNameConfig.properties конечной точки
- Файл log4j.properties конечной точки

Внедрение IBM Interact Inbound Gateway для IBM Universal Behavior Exchange и конечной точки

1. Скачайте и разархивируйте `IBM_Interact_OMO_Gateway_for_UBX_Subscriber_2.0.zip` в каталог, где установили Interact на сервере среды выполнения Interact.
2. Скачайте и распакуйте `IBM_Interact_OMO_Endpoint_for_UBX_Subscriber_2.0.zip` в любой каталог (например, c:\ubxInboundEndpoint) на общедоступном сервере прикладных программ, поддерживающем JavaEE, или на веб-сервере. Этот сервер будет публиковать во входную очередь JMS Interact данные, которые затем будут обрабатываться IBM Interact Inbound Gateway для IBM Universal Behavior Exchange.

Конфигурирование IBM Interact Inbound Gateway для конечной точки для IBM Universal Behavior Exchange Interact Inbound Gateway

Конечная точка IBM Interact Inbound Gateway для IBM Universal Behavior Exchange сконфигурирована для приема требований от Universal Behavior Exchange и отправляет их через шлюз IBM Interact Inbound Gateway для IBM Universal Behavior Exchange.

Выполните следующие задачи, чтобы сконфигурировать конечную точку Universal Behavior Exchange Subscriber Gateway

1. Нужно сконфигурировать новое свойство системы Java (-DubxInboundEndpointConfigPath), отредактировав файл конфигурации на веб-сервере или на консоли управления сервера прикладных программ. В свойстве -D должен быть указан каталог установки конечной точки на сервере. Этот каталог содержит файлы конфигурации для очереди JMS назначения и различных уровней записи в журнал для конечной точки. Например, -DubxInboundEndpointConfigPath=c:\ubxInboundEndpoint.
2. Внедрите IBM Interact Inbound Gateway для файла веб-архива конечной точки IBM Universal Behavior Exchange (ubxInboundEndpoint.war) из каталога установки, как описано в документации веб-сервера или сервера прикладных программ.

Чтобы проверить, что конечная точка была установлена правильно, введите следующий адрес в любой браузер и найдите сообщение `UBX End Point is UP` (конечная точка UBX работает).

`http://[сервер]:[порт]/[корневой_контекст]/UBXEndPoint`

Примечание: Надо защитить общедоступный шлюз IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange, добавив необходимые правила брандмауэра, чтобы принимать запросы http только с сервера IBM Universal Behavior Exchange Server.

Например, можно использовать следующие инструкции, чтобы сконфигурировать и внедрить IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange на сервере WebSphere Application Server.

1. Откройте административную консоль.
2. Выберите **Серверы > (Раскрыть типы серверов) > имя_сервера > (Раскрыть Java™ и управление процессами) > Определение процесса > Виртуальная Java-машина**.

3. В разделе обычных аргументов JVM добавьте свойство -DubxInboundEndpointConfigPath=<каталог установки конечной точки Universal Behavior Exchange Subscriber Gateway на сервере прикладных программ>. Например, добавьте свойство -DubxInboundEndpointConfigPath=C:\ubxInboundEndpoint.
4. Нажмите кнопку **ОК**, чтобы сохранить изменения в главной конфигурации.
5. Перезапустите сервер прикладных программ.

Внедрите конечную точку в WebSphere Application Server.

1. Зарегистрируйтесь на консоли администратора .
2. Перейдите к **Программы > Типы программ > Программы предприятия WebSphere**. Нажмите кнопку **Установить**.
3. Воспользуйтесь опцией **Подготовка к установке программы**, чтобы найти файл war конечной точки (`ubxInboundEndpoint.war`), который нужно установить, и щелкните по **Далее**.
4. На следующих страницах нажимайте кнопку **Далее**, пока не дойдете до **Отобразить корни контекста для веб-модулей**.
5. Воспользуйтесь опцией **Отобразить корни контекста для веб-модулей**, чтобы найти Корень контекста, и измените значение на `/UBXEndPoint`. Это значение становится корнем контекста. Click **next**.
6. Нажмите кнопку **Готово**.
7. После завершения установки прикладной программы щелкните по **Сохранить**, чтобы сохранить изменения в главной конфигурации.
8. Вернувшись в список установленных прикладных программ, включите переключатель для ubxInboundEndpoint_war и щелкните по **Запустить**, чтобы загрузить файл.

Конфигурирование IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange с IBM MQ (необязательно)

По умолчанию IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange работает с ActiveMQ. Чтобы сконфигурировать конечную точку с IBM MQ, выполните следующие указания.

Подготовка файлов JAR IBM MQ:

Для работы фабрик соединений необходимо, чтобы на клиенте, где запускается конечная точка, были доступны определенные файлы JAR IBM MQ.

Если на компьютере конечной точки уже установили IBM MQ, нужные файлы JAR уже получены при установке IBM MQ. Добавьте следующие два файла JAR в переменную среды CLASSPATH уровня системы. В Windows файлы JAR автоматически добавляются в параметр classpath при установке IBM MQ.

```
[MQ_HOME]\java\bin\com.ibm.mq.jar
[MQ_HOME]\java\bin\com.ibm.mqjms.jar
```

Если же на компьютере не установлен IBM MQ, нужно скопировать com.ibm.mq.allclient.jar и jms.jar с сервера MQ на сервер конечной точки и вручную добавить их в CLASSPATH.

Дополнительную информацию об установке или перемещении файлов JAR IBM MQ смотрите в разделе http://www.ibm.com/support/docview.wss?uid=swg21376217.

На сервере прикладных программ должна работать версия Java 1.7 или новее, поскольку файлы JAR IBM MQ v8 не поддерживают Java 1.6.

WebSphere Application Server поставляется с поддержкой IBM MQ, и никаких дополнительных файлов JAR не требуется.

Конфигурирование конечной точки

1. Перейдите в каталог <каталог установки конечной точки на сервере прикладных программ>.
2. Создайте резервную копию или переименуйте `ubxInboundEndpoint-spring.xml` и `ubxInboundEndpoint.properties`.
3. Перейдите в подкаталог `IBMMQ`. Там будут храниться альтернативные версии указанных выше файлов.
4. Добавьте информацию о соединении с сервером MQ в эту версию `ubxInboundEndpoint.properties`.
5. Скопируйте `ubxInboundEndpoint-spring.xml` и `ubxInboundEndpoint.properties` из `/ubxInbdoundEndpoint/IBMMQ` в каталог `main/ubxInboundEndpoint`.

Конфигурирование файла ubxInboundEndpoint.properties IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange

Сконфигурируйте в файле `ubxInboundEndpoint.properties`, куда отправлять полезную нагрузку события Universal Behavior Exchange в IBM Interact Inbound Gateway для IBM Universal Behavior Exchange. Файл `ubxInboundEndpoint.properties` находится в каталоге <каталог установки конечной точки на сервере прикладных программ>.

jmsBrokerUrl
: Обязательный параметр - Информация об очереди JMS, в которую записывает данные генератор событий.

jmsMaximumRetries
: Обязательный параметр - Максимальное число повторных попыток отправить сообщение в очередь JMS.

jmsRetryDelay
: Обязательный параметр - Задержка повторной доставки в миллисекундах.

maximumEndPointThreadPoolSize
: Обязательный параметр - Максимальное число потоков в пуле потоков для обработки данных событий IBM Universal Behavior Exchange и записи в очередь JMS. Это целое число определяет размер пула потоков.

clientIDFieldName
: Необязательный параметр - Имя поля, используемое в полезной нагрузке для ID клиента (подкатегория). Подкатегория используется, когда эта программа запускается в нескольких экземплярах одного продукта. Например: clientIDFieldName=clientID

Для вступления в силу изменений, внесенных в этот файл, требуется перезапустить веб-программу конечной точки шлюза (ubxInboundEndpoint.war) на веб-сервере или сервере прикладных программ.

Конфигурирование файла inboundProducerNameConfig.properties IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange (необязательно)

IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange отправляет событие в Interact, записывая сообщение в очередь JMS. Сообщение о событии по умолчанию содержит как имя генератора событий значение UBX. Сконфигурируйте файл inboundProducerNameConfig.properties, чтобы переопределить имя генератора событий с учетом значения поля источника UBX из полезной нагрузки. Обычно это имя конечной точки UBX. Файл inboundProducerNameConfig.properties находится в каталоге <каталог установки конечной точки на сервере прикладных программ>.

SOURCE.{имя источника UBX}={имя генератора событий}

Пример: SOURCE.CustomerAEndpoint=UBX-CustomerAEndpoint.

Для вступления в силу изменений, внесенных в этот файл, требуется перезапустить веб-программу конечной точки шлюза (ubxInboundEndpoint.war) на веб-сервере или сервере прикладных программ.

Конфигурирование файла свойств inboundQueueNameConfig.properties конечной точки шлюза (необязательно)

IBM Interact Inbound Gateway для конечной точки IBM Universal Behavior Exchange отправляет событие в Interact, записывая сообщение в очередь JMS. Имя очереди по умолчанию совпадает с именем генератора событий. Сконфигурируйте файл inboundQueueNameConfig.properties, чтобы переопределить имя очереди JMS по умолчанию как имя генератора событий. Имя генератора событий по умолчанию - UBX, если оно не переопределено в файле inboundQueueNameConfig.properties. Файл inboundProducerNameConfig.properties находится в каталоге <каталог установки конечной точки на сервере прикладных программ>.

{имя генератора событий} = {имя очереди JMS}

```
Пример:
UBX=UBXInboundQueue.
UBX-CustomerAEndpoint=UBX-CustomerAEndpointQueue
```

Для вступления в силу изменений, внесенных в этот файл, требуется перезапустить веб-программу конечной точки шлюза (ubxInboundEndpoint.war) на веб-сервере или сервере прикладных программ.

Конфигурирование файла свойств log4j.properties конечной точки шлюза

Сконфигурируйте файл log4j.properties, чтобы задать другой уровень записи в журнал для конечной точки. Файл log4j.properties находится в каталоге <каталог установки конечной точки на сервере прикладных программ>.

Описание

Задайте соответствующий уровень записи в журнал для log4j.logger.com.ibm.x1solution.jms.producer, log4j.logger.com.ibm.web.offerorchestration.inbound.common и log4j.logger.com.ibm.web.offerorchestration.inbound.ubx.

Конфигурирование файла interactEventNameMapping.properties

Используйте этот файл, чтобы отобразить значение поля события полезной нагрузки, определенное в файле `interactEventPayloadMapping.properties` как [EventName], в имя события Interact. Самый простой вариант - использовать то имя события, которое указано в полезной нагрузке события Universal Behavior Exchange. Файл `interactEventNameMapping.properties` находится в каталоге <Каталог установки>\conf\inbound\UBX.

{имя события UBX}={имя события Interact}

Пример: matchedIdentity=recommendedOfferEven

Если нужна поддержка данных полезной нагрузки от конкретного источника, можно также поместить этот файл в каталог <Каталог установки>\conf\inbound\UBX\{источник}. Значение источника должно соответствовать значению поля source в полезной нагрузке события Universal Behavior Exchange. Обычно это имя конечной точки Universal Behavior Exchange. Если нужна поддержка данных, использующих конкретные версии, можно также поместить этот файл в каталог <Каталог установки>\conf\inbound\UBX\{источник}\version-{версия}. Значение версии должно соответствовать значению поля version в полезной нагрузке события Universal Behavior Exchange. Для поддержки данных нескольких экземпляров Universal Behavior Exchange можно также поместить этот файл в каталог <Каталог установки>\conf\inbound\UBX\{источник}\version-{версия}\account-{ID клиента}. Значение ID клиента должно соответствовать значению поля clientID в полезной нагрузке события Universal Behavior Exchange.

Конфигурирование файла interactEventPayloadMapping.properties

Сконфигурируйте файл `interactEventPayloadMapping.properties`, чтобы отобразить входное поле на параметры API Interact. Файл `interactEventPayloadMapping.properties` находится в каталоге <Каталог установки>\conf\inbound\UBX.

Параметры API Interact: Значение должно начинаться с определения типа поля, далее следует либо статическое значение в двойных кавычках, либо имя поля из данных полезной нагрузки. (ТИП_ПОЛЯ)"СТАТИЧЕСКОЕ_ЗНАЧЕНИЕ" или (ТИП_ПОЛЯ)ИМЯ_ПОЛЯ_ПОЛЕЗНОЙ_НАГРУЗКИ. Допустимый ТИП_ПОЛЯ - `String` (строковое), `Numeric` (числовое) или `DateTime` (дата и время).

```
Пример:
[SessionID]=(String)interactprofileid
[EventName]=(String)code
[AudienceIDFieldNames]=(String)"измени_меня"
[AudienceIDFieldValues]=(String)interactprofileid
[AudienceLevel]=(String)"измени_меня"
[InteractChannel]=(String)"измени_меня"
```

Данные события: Эти свойства служат для отображения атрибутов события, которые могут использоваться в сообщениях исходящих каналов. Левая сторона содержит имена переменных, используемые в сообщениях исходящих каналов.

Значение должно начинаться с определения типа поля, далее следует либо статическое значение в двойных кавычках, либо имя поля из данных полезной нагрузки. (ТИП_ПОЛЯ)"СТАТИЧЕСКОЕ_ЗНАЧЕНИЕ" или (ТИП_ПОЛЯ)ИМЯ_ПОЛЯ_ПОЛЕЗНОЙ_НАГРУЗКИ. Допустимый ТИП_ПОЛЯ - `String` (строковое), `Numeric` (числовое) или `DateTime` (дата и время).

Если нужна поддержка данных полезной нагрузки от конкретного источника, можно также поместить этот файл в каталог <Каталог установки>\conf\inbound\UBX\{источник}. Значение источника должно соответствовать значению поля source в полезной нагрузке события Universal Behavior Exchange. Обычно это имя конечной точки Universal Behavior Exchange. Если нужна поддержка данных, использующих конкретные версии, можно также поместить этот файл в каталог <Каталог установки>\conf\inbound\UBX\{источник}\version-{версия}. Значение версии должно соответствовать значению поля version в полезной нагрузке события Universal Behavior Exchange. Для поддержки данных нескольких экземпляров Universal Behavior Exchange можно также поместить этот файл в каталог <Каталог установки>\conf\inbound\UBX\{источник}\version-{версия}\account-{ID клиента}. Значение ID клиента должно соответствовать значению поля clientID в полезной нагрузке события Universal Behavior Exchange.

Создание конечной точки и события в UBX

Это образец конечной точки и события, и их можно использовать как пример.

Чтобы создать конечную точку и событие в UBX, выполните следующие действия.

1. В клиенте API REST опубликуйте запросы к UBX.
2. Зарегистрируйте конечную точку в UBX при помощи JSON. Смотрите следующий пример:

```
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
(Примечание: Это ключ аутентификации, сгенерированный в пользовательском интерфейсе UBX.)
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. Зарегистрируйте тип события в UBX при помощи JSON. Смотрите следующий пример:

```
Регистрация события для события Interact в UBX
Method Call: POST
URL: https://ubx-qa1-api.adm01.com/v1/eventtype

Headers:
Content-Type: application/json
Accept-Charset: UTF-8
Authorization: Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
Note: Это ключ аутентификации, сгенерированный в пользовательском интерфейсе UBX.
Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
Body
{
   "name": "recommendedOffers",
   "description": "рекомендуемые предложения от OMO",
   "code": "recommendedOffers"
}
```

4. Опубликуйте событие в UBX при помощи JSON. Смотрите следующий пример:

```
{
  "channel" : "mobile",
  "identifiers" : [
    {
      "name" : "interactprofileid",
      "value" : "55"
    }
  ],
"events" : [
  {
    "code" : "recommendedOffers",
    "timestamp" : "2015-12-28T20:16:12Z"
  }
 ]
}
```

Использование IBM Interact Outbound Gateway для IBM Universal Behavior Exchange

Чтобы использовать IBM Interact Outbound Gateway для IBM Universal Behavior Exchange, надо сконфигурировать Interact, UBX и шлюз.

Используйте следующие конфигурации в качестве примера для вашей конфигурации.

В случае использования UBX как выходного канала Interact играет роль конечной точки типа публикатор, которая публикует события в UBX. Из UBX эти события можно отправить подписчику.

Прежде чем начать конфигурирование, запросите исходящий доступ к компьютеру хоста. У вас должен быть разрешен сетевой доступ к компьютера хоста.

Шлюз можно скачать с 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.

Регистрация конечных точек и событий в UBX

1. В UBX перейдите на вкладку **Конечные точки**. Щелкните по **Зарегистрировать новую конечную точку**, чтобы получить ключ аутентификации. Ключ

аутентификации, сгенерированный в UBX, нужно использовать для конечной точки издателя и для добавления события. Для конечной точки подписчика нужно сгенерировать новый ключ аутентификации в UBX. Запишите этот ключ.

2. Зарегистрируйте свою конечную точку издателя.
   a. Откройте клиентский инструмент API REST.
   b. Выберите метод как PUT.
   c. Передайте заголовки как

      ```
      Content-Type : application/json
      Accept-Charset: UTF-8
      Authorization : Bearer 520301d7-7855-4ea7-b19d-0b395c1e6ae4
      (ключ аутентификации, сгенерированный в UBX)
      ```

   d. Передать URL как

      ```
      URL: https://ubx-qa1-api.adm01.com/v1/endpoint
      ```

   e. Как текст передайте соответствующее имя для конечной точки издателя.

      Например:

      ```
      {
          "name":"Interact_Publisher",
          "description":"Конечная точка для сервера, созданного 30 января",
          "providerName":"IBM",    "url":"",
          "endpointTypes":{
              "event": {
              "source":{
                 "enabled":true
              }
          }
      },
      "marketingDatabasesDefinition":{
          "marketingDatabases":[
            {
                "name":"IDSync",
                "identifiers":[
                  {
                "name":"interactprofileid",
           "type":"INTERACTID"
                  }
             ]
          }
         ]
      }
      }
      ```

3. Зарегистрируйте свое событие. Запишите код [Event], переданный в тексте. Его нужно отобразить в файле ubxContentMapping.properties. Регистр символов учитывается.
   a. Откройте клиентский инструмент API REST.
   b. Выберите метод как POST.
   c. Передайте те же самые заголовки, которые использовали для конечной точки на предыдущем шаге.
   d. Передать URL как

      ```
      URL: https://ubx-qa1-api.adm01.com/v1/eventtype
      ```

   e. Как текст передайте соответствующее имя для события.

      Например:

      ```
      {
         "name": "recommendedOffer",
         "description": "рекомендуемый
            контакт из UBX",    "код":
            "recommendedOffer"}
      ```

Примечание: Переданный код события нужно отобразить в файле ubxContentMapping.properties. В коде события учитывается регистр.

4. Добавьте конечную точку подписчика.
   a. Откройте клиентский инструмент API REST.
   b. Выберите метод как `PUT`.
   c. Передайте те же самые заголовки, которые использовали для конечной точки на предыдущем шаге.
   d. Для регистрации конечной точки подписчика создайте новый ключ аутентификации в UBX.
   e. Передать URL как

      ```
      URL: https://ubx-qa1-api.adm01.com/v1/endpoint
      ```

   f. Как текст передайте соответствующее имя для конечной точки издателя.

      Например:

      ```
      {
         "name":"UBX_Subscriber",
         "description":"UBX_Subscriber для подписки на события ",
         "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. После добавления конечных точек издателя и подписчика и события надо подписать подписчика в UBX на события от издателя.
   a. В UBX щелкните по **Подписаться на события** на вкладке **События**.
   b. Выберите событие и назначение.
   c. Щелкните по **Подписаться**.

Конфигурирование Interact и шлюза

1. Добавьте шлюз UBX в свойство конфигурации **Interact | triggeredMessage | gateways**. Задайте для **ClassPath** значение `file:///root/opt/OMO/lib/OMO_OutboundGateway_UBX.jar` и для **ClassName** значение

   ```
   com.ibm.interact.offerorchestration.outboundgateway.ubx.
   UBXOutboundGateway
   ```

2. Распакуйте файл `OMO_OutboundGateway_UBX.zip` на компьютере хоста и найдите файл jar UBX из внешнего пути.

3. Добавьте OMO-conf_outbound_common_httpConnectionConfig как параметр в **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Как **значение** задайте file:///opt/Interact<версия>/Interact/OMO/conf/outbound/common/httpConnectionConfig.properties. Это каталог установки Interact. Утилита установки шлюза скачает каталог шлюза в установленный каталог Interact.

   В файле httpConnectionConfig.properties в папке Interact укажите срок ожидания.

   Например:

   ```
   connectTimeoutMs=180000
   ```

4. Добавьте OMO-conf_outbound_ubx_ubxConfig как параметр в **Interact | triggeredMessage | gateways | [имя_шлюза] | Parameter Data**. Как **значение** задайте путь к файлу ubxConfig.properties в папке Interact.

   В файле ubxConfig.properties задайте ubxURL, authKey и interactProfileIdFieldName.

   Например:

   ```
   authKey=912586bf-190d-48f9-8488-26f1bf532ef3
   [Ключ аутентификации, использованный для регистрации конечной точки издателя и события в UBX]
   interactProfileIdFieldName=interactprofileid
   [Имя поля из файла ubxContentMapping.properties]
   ```

5. Добавьте OMO-conf_outbound_ubx_ubxContentAdditionalAttributes как параметр в **Interact | triggeredMessage | gateways | [имя_шлюза] | Parameter Data**. Как **значение** задайте путь к файлу ubxContentAdditionalAttributes.properties в папке Interact.

6. Добавьте OMO-conf_outbound_ubx_ubxContentMapping как параметр в **Interact | triggeredMessage | gateways | [имя_шлюза] | Parameter Data**. Как **значение** задайте путь к файлу ubxContentMapping.properties в папке Interact.

   Измените значения interactprofileid и eventName в файле ubxContentMapping.properties.

   Три формата допустимы при передаче имени события: если значение задано в двойных кавычках, это статическое значение; если значение задано в формате offer.offerAttributeName, оно отображается на атрибут предложения offerAttributeName; если же, наконец, значение дано в формате profile.profileAttributeName, оно отображается на атрибут профиля profileAttributeName. Значение имения события должно соответствовать коду, использованному при регистрации события в UBX. Регистр символов учитывается.

   Например:

   ```
   eventName="abandoned_shopping_carts"
   eventName=offer.Card
   eventName=profile.EMAIL
   ```

7. Добавьте канал в свойство конфигурации **Interact | triggeredMessage | channel**.
8. Определите тот же самый канал при разработке в параметре **Campaign | partitions | partition [n] |Interact | outboundChannels**
9. Перезапустите сервер прикладных программ.
10. Создайте правило инициированных сообщений с именем события и использующее канал, который вы добавили на предыдущих шагах.
11. Внедрите интерактивный канал.
12. В тестовом клиенте API запустите сеанс для интерактивного канала, где сконфигурировано правило инициированных сообщений, и опубликуйте событие, которым инициируется предложение для UBX.

Использование IBM Interact Outbound Gateway для IBM Mobile Push Notification

Чтобы использовать этот шлюз мобильной push-отправки или издателя, нужно сконфигурировать Interact, IBM Marketing Cloud и шлюз.

Используйте следующие конфигурации в качестве примера для вашей конфигурации.

Этот шлюз можно скачать с https://www-945.ibm.com/support/fixcentral/swg/downloadFixes

Конфигурирование IBM Marketing Cloud

1. Убедитесь, что у вас есть учетная запись IBM Marketing Cloud с доступом push. Кроме того, запишите ваши ID клиента, секретный пароль клиента и маркер обновления.
2. На вкладке **Данные** создайте новую базу данных. Добавьте в базу данных новый `ID мобильного пользователя` и поля по умолчанию.
3. На вкладке **Поиск** найдите поле `ID мобильного пользователя`. Поместите указатель мыши на первое поле `Нет сообщений электронной почты`. В нижней части окна браузера появится `ID получателя`. Добавьте этот `ID получателя` в таблицу профилей Interact.

Конфигурирование IBM Interact Outbound Gateway для IBM Mobile Push Notification

1. Скачайте и установите шлюз мобильной push-отправки с https://www-945.ibm.com/support/fixcentral/swg/downloadFixes
2. Сконфигурируйте файл `silverpopEngagePushConfig.properties`.

   Например:

   ```
   OauthServiceURL=https://apipilot.silverpop.com/oauth/token
   pushServiceURL=https://apipilot.silverpop.com/rest/channels/push/sends
   ```
3. Сконфигурируйте файл `silverpopEngagePushContentMapping.properties`.

   Например:

   ```
   Атрибуты таблицы Профиль Interact:
   appKey=appKey
   engageRecipientId=recipientId
   mobileUserId=mobileUserId
   deviceType=deviceType

   Атрибуты предложений Interact:
   simpleSubject=simpleSubjectAttr
   simpleMessage=simpleMessageAttr
   simpleActionData=simpleActionDataAttr
   simpleActionType=simpleActionTypeAttr
   simpleActionLabel=simpleActionLabelAttr
   personalizeAttributeList=personalizeAttributeList
   contentId=ContentID
   campaignId=campaignId
   ```

Производится конфигурирование Interact

1. Создайте следующие атрибуты предложения.

   ```
   simpleActionDataAttr: string
   simpleActionLabelAttr: String
   simpleActionTypeAttr: string
   simpleMessageAttr: string
   ```

```
simpleSubjectAttr: string
contentID: string
campaignId=string
personalizeAttributeList=string
```

2. Создайте шаблон предложения с атрибутами предложения и следующими значениями предложения.

   ```
   simpleActionDataAttr: www.ibm.com
   simpleActionLabelAttr: Open URL
   simpleActionTypeAttr: url
   simpleMessageAttr: <Введите здесь текст вашего сообщения>
   simpleSubjectAttr: <Введите здесь тему>
   contentID: ID шаблона push-сообщения, созданного в Engage.
   PersonalizeAttributeList: Список разделенных запятой пар имя атрибута
   - значение, которые нужно поместить в раздел personalizationDefaults
   полезной нагрузки, отправляемой в Engage.
   ```

   Если используется атрибут `contentID`, остальные атрибуты `simple..` игнорируются, поскольку подробности заполнения берутся из шаблона Engage.

   Example personalizedAttributeList

   ```
   personalizeAttributeList=discount=10,Offercost=20
   campaignId=имя, которое вы хотите использовать для этой кампании.
   ```

3. Ваша таблица профилей содержит следующие столбцы и значения.

   ```
   appKey: gcsTQo6v79
   recipientId: 13472242
   deviceType: android or ios
   ```

4. Добавьте шлюз в свойство конфигурации **Interact | triggeredMessage | gateways**. Задайте для **ClassName** значение

   ```
   com.ibm.interact.offerorchestration.outboundgateway.silverpop.engage.push.
   SilverpopEngagePushOutboundGateway
   ```

   Задайте для **ClassPath** значение `file://<домашний_каталог_EngagePushGateway>/lib/OMO_OutboundGateway_Silverpop_Engage_Push.jar`.

5. Добавьте `OMO-silverpopEngagePushConfig` как параметр в **Interact | triggeredMessage | gateways | [имя_шлюза] | Parameter Data**. Задайте как **значение** путь к файлу `silverpopEngagePushConfig.properties`.
6. Добавьте `OMO-silverpopEngagePushContentMapping` как параметр в **Interact | triggeredMessage | gateways | [имя_шлюза] | Parameter Data**. Задайте как **значение** путь к файлу `silverpopEngagePushContentMapping.properties`.
7. Добавьте `OMO-conf_outbound_common_httpConnectionConfig` как параметр в **Interact | triggeredMessage | gateways | [имя_шлюза] | Parameter Data**. Задайте как **значение** путь к файлу `httpConnectionConfig.properties`.

   В файле `httpConnectionConfig.properties` в папке Interact укажите срок ожидания.

   Например:

   ```
   connectTimeoutMs=6000
   ```

8. Создайте канал и обработчик в **Interact | triggeredMessage** и используйте шлюз [Mobile_Push], который создали выше в этом канале. Этот канал указывается в инициированном сообщении для отправки push-сообщений.
9. Создайте интерактивный канал и добавьте в правило триггера инициированное сообщение с предложением, которое вы создали ранее.
10. Внедрите интерактивный канал.
11. В тестовом клиенте API выполните `startSession` для интерактивного канала, в котором сконфигурировано инициированное сообщение, и `postEvent`, отправляющее предложение в Mobile Push.

12. Посмотрите журналы Interact и убедитесь в успешности push-отправки. Код состояния `202` означает успешную доставку.

Использование IBM выходного шлюза Email (Transact) Interact для IBM Marketing Cloud

Эту интеграцию можно использовать с выходным шлюзом Silverpop, Interact и IBM Interact Email (Transact) для IBM Marketing Cloud для отправки инициируемых предложений по электронной почте вашим клиентам.

Убедитесь, что выполнены следующие предварительные требования.

- Создайте таблицу профилей аудитории покупателей со столбцом электронной почты. Используйте эту таблицу профилей для вашего интерактивного канала.
- Затребуйте для вашего выходного канала включенный доступ через сеть к хост-компьютеру.
- Скопируйте и извлеките файл `OMO_OutboundGateway_Silverpop.zip` на хост-компьютере

Шлюз можно скачать с 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.

Добавление диспетчера для интеграции шлюза

Диспетчер добавляет ваше предложение в очередь для выходного шлюза Email (Transact) IBM Interact для IBM Marketing Cloud, после чего предложение может быть отправлено по электронной почте.

Об этой задаче

Для работы с выходным шлюзом Email (Transact) IBM Interact для IBM Marketing Cloud необходимо добавить диспетчер.

Процедура

1. Перейдите в свойствах конфигурации в **Interact | triggeredMessage | dispatchers | <имя_диспетчера>**.
2. Добавьте **Имя новой категории** для вашего диспетчера.
3. Выберите **тип**. Можно выбрать из InMemoryQueue, JMSQueue и Пользовательский.
4. Введите **имя_класса**.
5. Введите **путь_класса**.

Добавление шлюза для IBM выходного шлюза Email (Transact) Interact для IBM Marketing Cloud

В интеграции шлюз отправляет покупателям по электронной почте предложения, соответствующие требованиям.

Об этой задаче

Необходимо добавить шлюз для этой интеграции.

Примечание: Interact не поддерживает нескольких экземпляров одного шлюза.

Процедура

1. Перейдите в свойствах конфигурации в **Interact | triggeredMessage | gateways | <имя_шлюза>**.
2. Добавьте **Имя новой категории** для вашего шлюза.
3. Задайте для **className** следующий путь.

   ```
   com.ibm.interact.offerorchestration.outboundgateway.
   silverpop.SilverpopEmailOutboundGateway
   ```

4. Задайте для **classPath** положение выходного шлюза - путь файла JAR из извлеченной папки.

   Например:

   ```
   file:///opt/OMO_SilverPop/OMO_OutboundGateway_Silverpop/lib/
   OMO_OutboundGateway_Silverpop.jar
   ```

5. Добавьте для вашего шлюза следующие параметры.

   ```
   OMO-conf_outbound_common_httpConnectionConfig
   OMO-conf_outbound_silverpop_silverpopConfig
   OMO-conf_outbound_silverpop_silverpopContentMapping
   deliveryTimeoutMillis
   ```

Конфигурирование параметра OMO-conf_outbound_common_httpConnectionConfig

Нужно сконфигурировать параметр OMO-conf_outbound_common_httpConnectionConfig для вашего шлюза.

Процедура

1. Перейдите в свойствах конфигурации в **Interact | triggeredMessage | gateways | <имя_шлюза_Silverpop> | OMO-conf_outbound_common_httpConnectionConfig**.
2. Задайте **значение** - `file:///opt/Interact<версия>/Interact/OMO/conf/outbound/common/httpConnectionConfig.properties`. Это каталог установки Interact. Программа установки Interact скачивает файл `httpConnectionConfig.properties` в каталог установки Interact.
3. В файле `httpConnectionConfig.properties` папки Interact задайте срок ожидания.

   Например:

   ```
   connectTimeoutMs=60000
   ```

Конфигурирование параметра OMO-conf_outbound_silverpop_silverpopConfig

Нужно сконфигурировать параметр OMO-conf_outbound_silverpop_silverpopConfig для вашего шлюза.

Процедура

1. Перейдите в свойствах конфигурации в **Interact | triggeredMessage | gateways | <имя_шлюза_Silverpop> | OMO-conf_outbound_silverpop_silverpopConfig**.
2. Задайте **значение** - путь к файлу `silverpopConfig.properties` в папке `OMO_OutboundGateway_silverpop`.

   Например:

file:///opt/OMO_SilverPop/OMO_OutboundGateway_Silverpop/conf/outbound/silverpop/silverpopConfig.properties

3. Файл silverpopConfig.properties в извлеченной папке OMO_OutboundGateway_Silverpop.zip задает значения для OauthServiceURL, xmlAPIServiceURL, clientID, clientSecret и refreshToken. Обратитесь к вашему администратору Marketing Cloud, чтобы получить значения для конкретных клиентов из файла transact.xml.

Конфигурирование параметра OMO-conf_outbound_silverpop_silverpopContentMapping

Нужно сконфигурировать параметр OMO-conf_outbound_silverpop_silverpopContentMapping для вашего шлюза.

Процедура

1. Перейдите в свойствах конфигурации в **Interact | triggeredMessage | gateways | <имя_шлюза_Silverpop> | OMO-conf_outbound_silverpop_silverpopContentMapping**.
2. Задайте **значение** - путь к файлу silverpopContentMapping.properties в папке OMO_OutboundGateway_silverpop.
3. В файле silverpopContentMapping.properties в папке OMO_OutboundGateway_Silverpop.zip задайте значения для отображения вашего содержимого.
   a. Задайте свойство campaignId. Значение для этого свойства - имя атрибута предложения, заданное в ваших шаблонах предложений.
   b. Задайте свойство email. Значение для этого свойства - имя столбца в вашей таблице профилей. Добавьте столбец email в вашу таблицу профилей и укажите ID электронной почты. Это ID почты получателей.
   c. Определите атрибуты вашего предложения в additionalOfferPfAttributesUsedInEmail. Это свойство задает атрибуты, необходимые для шаблона почтовых сообщений, по шаблону вашего предложения. Чтобы определить поля по вашей таблице профилей, используйте additionalProfilePfAttributesUsedInEmail. Для обозначения всех атрибутов и значений столбцов используйте символ *.

Конфигурирование параметра deliveryTimeoutMillis

Чтобы увеличить срок ожидания сервера Interact для соединения с сервером Marketing Cloud, задайте значение параметра deliveryTimeoutMills.

Об этой задаче

Процедура

1. Перейдите в свойствах конфигурации в **Interact | triggeredMessage | gateways | <имя_шлюза_Silverpop> | deliveryTimeoutMillis**.
2. Задайте **значение**. Например, можно установить **значение** 60000. Это увеличит срок ожидания сервера до 60000 миллисекунд.

Добавьте обработчик канала для IBM выходного шлюза Email (Transact) Interact для IBM Marketing Cloud

Добавьте обработчик канала для среды выполнения Interact.

Процедура

1. Перейдите в свойствах конфигурации в **Interact | triggeredMessage | channels | <имя_канала_Silverpop> | <имя_обработчика>**.

2. Добавьте **Имя новой категории** для вашего обработчика канала.
3. Задайте имя для ранее добавленного диспетчера.
4. Задайте имя для ранее добавленного шлюза.
5. Задайте **режим**. Если выбрано Failover, этот обработчик будет использоваться только в том случае, если всем обработчикам с более высоким приоритетом в этом канале не удалось отправить предложения. Если выбрано **Addon**, этот обработчик используется независимо от того, успешно ли отправили предложения другие обработчики.
6. Задайте **приоритет** для этого обработчика.

Добавление выходного канала для IBM выходного шлюза Email (Transact) Interact для IBM Marketing Cloud

Добавьте выходной канал в среде разработки Interact.

Процедура

1. Перейдите в свойствах конфигурации в **Campaign | partitions | partition[n] | Interact | outboundChannels**.
2. Добавьте **Имя новой категории** для вашего выходного канала.
3. Добавьте **имя** для вашего выходного канала. Убедитесь, что имя канала совпадает с именем, добавленным вами в свойствах конфигурации **Interact | triggeredMessage | channels | <имя_канала_Silverpop>**.

Конфигурирование транзакционной почтовой отправки при помощи IBM выходного шлюза Email (Transact) Interact для IBM Marketing Cloud

Для отправки предложений по электронной почте необходимо сконфигурировать транзакционную почтовую отправку.

Процедура

1. В Marketing Cloud (Transact) выберите в меню **Данные > Создать базу данных**. Затем нажмите кнопку **Создать**, чтобы создать таблицу профилей. Можно также импортировать таблицу профилей, в которую вы добавили столбец электронной почты.
2. Выберите **Автоматизация > Транзакционные сообщения > Создать группу**. Выберите **Transact** в качестве **Триггера событий**. Надо также выбрать ранее созданный вами источник данных. Нажмите кнопку **Сохранить и активировать**.

   У предложения, отправленного через Marketing Cloud, должен быть тот же атрибут, который вы задали для `campaignId` в файле `silverpopContentMapping.properties`. Значение этого атрибута предложения - `campaignId`, генерируемый для группы автоматизированных сообщений.
3. Выберите в меню **Содержимое > Создать почтовые сообщения** и выберите источник содержимого из предыдущего шага. Введите тело почтового сообщения. Выберите **Автоматизировать**. Выберите **Назначить почтовое сообщение в существующую группу автоматизированных сообщений**. Нажмите кнопку **Передать и активировать**.

   Строку темы почтового сообщения и его тело можно персонализировать при помощи атрибутов предложения и атрибутов профиля. Для задания атрибутов используйте синтаксис `%%Attribute_Name%%`.

4. Сервер Marketing Cloud принимает передачи для выходных шлюзов только от заранее сконфигурированных IP-адресов. Чтобы добавить IP-адрес, перейдите в **Параметры > Администратор организации > Параметры защиты > Ограничения доступа**.
5. Если используется WebSphere Application Server, надо импортировать сертификат Marketing Cloud SSL. Для пользователей WebLogic это не требуется.
   a. На консоли WebSphere Application Server перейдите в **Сертификат SSL и управление ключами > Склады ключей и сертификаты > NodeDefaultTrustStore > Сертификаты подписавшего > Получить с порта**.
   b. Задайте хост и порт.
   c. Перезапустите WebSphere Application Server.

Прежде чем обращаться в службу технической поддержки IBM

Если вы столкнетесь с проблемой, которую невозможно разрешить при помощи документации, тот, кто назначен для вашей компании в качестве контактного лица для обращения в службу поддержки, может записать вызов в службу технической поддержки IBM. Используйте эти рекомендации, чтобы убедиться, что ваша проблема будет разрешена эффективно и успешно.

Если вы не отвечаете за поддержку в вашей компании, то за информацией обратитесь к своему администратору IBM.

Примечание: Служба технической поддержки не записывает и не создает сценарии API. За помощью по реализации наших предложений API обращайтесь в профессиональную службу IBM (IBM Professional Services).

Какую информацию нужно собрать

Перед тем как обратиться в службу поддержки IBM, соберите следующие сведения:

- Краткое описание характера проблемы.
- Подробно: сообщения об ошибках, появляющиеся при возникновении проблемы.
- Подробное описание шагов по воспроизведение проблемы.
- Связанные файлы журналов, файлы сеансов, файлы конфигурации и файлы данных.
- Информацию о среде продукта и системы , которую можно получить, как рассказывается в разделе "Информация о системе".

Информация о системе

При обращении в службу технической поддержки IBM вас могут попросить предоставить информацию о среде вашей системы.

Если проблема не мешает вам войти в систему, большая часть этой информации находится на странице О программе, где представлена информация об установленных программах IBM .

Доступ к странице О программе можно получить, выбрав **Справка > О программе**. Если страница О программе недоступна, смотрите файл `version.txt`, который находится в каталоге установки вашей программы.

Контактная информация для службы технической поддержки IBM

Как обратиться в службу технической поддержки IBM, можно узнать на веб-сайте технической поддержки продукта IBM: (http://www.ibm.com/support/entry/portal/open_service_request).

Примечание: Чтобы ввести запрос на поддержку, вы должны зарегистрироваться с учетной записью IBM. Эта учетная запись должна быть связана с вашим номером заказчика IBM. Чтобы подробнее узнать о связывании вашей учетной записи с вашим

номером заказчика IBM, смотрите **Support Resources>Entitled Software Support** (Ресурсы поддержки - Предоставляемая поддержка программ) в портале поддержки.

Замечания

Эта информация относится к продуктам и услугам, предоставляемым в США.

IBM может предлагать описанные продукты, услуги и возможности не во всех странах. Сведения о продуктах и услугах, доступных в настоящее время в вашей стране, можно получить в местном представительстве IBM. Любые ссылки на продукты, программы или услуги IBM не означают явным или неявным образом, что можно использовать только продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако при этом ответственность за оценку и проверку работы всех продуктов, программ или услуг не-IBM возлагается на пользователя.

IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данной публикации. Получение этого документа не означает предоставления каких-либо лицензий на эти патенты. Запросы по поводу лицензий следует направлять в письменной форме по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране или направьте запрос в письменной форме по адресу:

Intellectual Property Licensing
лицензированию интеллектуальной собственности
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Следующий абзац не применяется в Великобритании или в любой другой стране, где подобные заявления противоречат местным законам: INTERNATIONAL BUSINESS MACHINES CORPORATION ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ "КАК ЕСТЬ" БЕЗО ВСЯКИХ ГАРАНТИЙ, КАК ЯВНЫХ, ТАК И ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ТАКОВЫМИ, ПОДРАЗУМЕВАЕМЫЕ ГАРАНТИИ СОБЛЮДЕНИЯ ЧЬИХ-ЛИБО АВТОРСКИХ ПРАВ, ВОЗМОЖНОСТИ КОММЕРЧЕСКОГО ИСПОЛЬЗОВАНИЯ ИЛИ ПРИГОДНОСТИ ДЛЯ КАКИХ-ЛИБО ЦЕЛЕЙ. В некоторых странах для определенных сделок подобные оговорки не допускаются, таким образом, это утверждение может не относиться к вам.

Данная информация может содержать технические неточности и типографские опечатки. Периодически в информацию вносятся изменения, они будут включены в новые издания этой публикации. IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в данной информации на сайты, не принадлежащие IBM, приводятся только для удобства и никоим образом не означают поддержки IBM этих сайтов. Материалы на таких веб-сайтах не являются составной частью материалов по данному продукту IBM, и вся ответственность за пользование такими веб-сайтами лежит на вас.

IBM может использовать или распространять информацию так, как сочтет нужным, без каких-либо обязательств с ее стороны.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Corporation
B1WA LKG1
550 King Street
Littleton, MA 01460-1250
U.S.A.

Такая информация может быть предоставлена на определенных условиях (в некоторых случаях к таким условиям может относиться оплата).

Лицензированная программа, описанная в данном документе, и все прилагаемые к ней материалы предоставляются IBM на условиях IBM Customer Agreement (Соглашения IBM с заказчиком), Международного соглашения о лицензиях на программы IBM или эквивалентного соглашения.

Приведенные данные о производительности измерены в контролируемой среде. Таким образом, результаты, полученные в других операционных средах, могут существенно отличаться от них. Некоторые показатели измерены получены в системах разработки и нет никаких гарантий, что в общедоступных системах эти показатели будут теми же. Более того, некоторые результаты могут быть получены путем экстраполяции. Реальные результаты могут отличаться от них. Пользователи должны проверить данные для своих конкретных сред.

Информация о продуктах других фирм получена от поставщиков этих продуктов, из их опубликованных объявлений или из других общедоступных источников. IBM не проверяла эти продукты и не может подтвердить точность измерений, совместимость или прочие утверждения о продуктах других фирм. Вопросы о возможностях продуктов других фирм следует направлять поставщикам этих продуктов.

Все утверждения о будущих направлениях разработок или намерениях фирмы IBM могут быть пересмотрены или отменены без дополнительного объявления, и отражают исключительно предполагаемые цели фирмы.

Все указанные здесь цены IBM являются текущими рекомендуемыми ценами на продукты IBM, и они могут измениться безо всякого уведомления. Дилерские цены могут отличаться от них.

Эта информация содержит примеры данных и отчетов, иллюстрирующие типичные деловые операции. Чтобы эти примеры были правдоподобны, в них включены имена лиц, названия компаний и товаров. Все эти имена и названия вымышлены и любое их сходство с реальными именами и адресами полностью случайно.

ЛИЦЕНЗИЯ НА ПЕРЕПЕЧАТКУ:

Эта информация содержит примеры прикладных программ на языках программирования, иллюстрирующих приемы программирования для различных операционных платформ. Разрешается копировать, изменять и распространять эти примеры программ в любой форме без оплаты IBM для целей разработки, использования, сбыта или распространения прикладных программ, соответствующих интерфейсу прикладного программирования операционных платформ, для которых эти примера программ написаны. Эти примеры не были всесторонне проверены во всех возможных условиях. Поэтому IBM не может гарантировать их надежность, пригодность и функционирование. Пробные программы предоставляются по принципу 'как есть', без какой-либо гарантии. IBM не несет ответственности ни за какой ущерб, вызванный использованием пробных программ.

Если вы просматриваете эту информацию на экране, фотографии и цветные иллюстрации могут быть не видны.

Товарные знаки

IBM, логотип IBM и ibm.com - товарные знаки или зарегистрированные товарные знаки корпорации International Business Machines во многих юрисдикциях мира. Прочие названия продуктов и услуг могут быть товарными знаками IBM или других компаний. Текущий список товарных знаков IBM находится в веб на странице "Copyright and trademark information" (Информация об авторских правах и товарных знаках) по адресу: www.ibm.com/legal/copytrade.shtml.

Замечания относительно политики конфиденциальности и положений об использовании

В программных продуктах IBM, включая программу как служебное решение ("Предложения относительно программ"), могут использоваться элементы cookie или другие технологии для сбора информации об использовании продукта, чтобы помочь улучшить опыт работы конечного пользователя, настроить взаимодействия с конечным пользователем или для других целей. Элемент cookie - это фрагмент данных, которые веб-сайт может отправить в ваш браузер и которые затем могут храниться на вашем компьютере в виде тега, идентифицирующего ваш компьютер. Во многих случаях никакой личной информации эти компоненты cookie не собирают. Если используемое вами Предложение относительно программ позволяет вам собирать личную информацию через компоненты cookie и аналогичные технологии, мы информируем вас ниже о соответствующих особенностях.

В зависимости от внедренных конфигураций данное Предложение относительно программ может использовать сеансы и хранимые компоненты cookie, которые собирают имя каждого пользователя и другую личную информацию для управления сеансами, усовершенствованной работы пользователей или других целей, касающихся отслеживания использования, или функциональных целей. Эти компоненты cookie можно отключить, но при их отключении также будут отключены функции, для поддержки которых они предназначены.

Различные законодательства регулируют сбор личной информации через компоненты cookies и аналогичные технологии. Если конфигурации, внедренные для этого Предложения относительно программ, обеспечивают вам, как заказчику, возможность собирать личную информацию от конечных пользователей через cookies и другие технологии, вы должны обратиться за местной юридической рекомендацией о том, существуют ли какие-либо законы, применимые к такому сбору данных, включая все требования относительно предоставления замечаний и согласований в тех случаях, где это применимо.

IBM требует, чтобы Клиенты (1) обеспечивали четкую и явную связь с терминами веб-сайта Заказчика относительно использования (например, политики конфиденциальности), включая связь со сбором и практикой использования данных IBM и Клиентом, (2) сообщали о том, что элементы cookie и явные элементы gif/веб-маяки помещались на компьютер посетителя компанией IBM от имени Клиента вместе с пояснением цели такой технологии, и (3) в той степени, в которой это требуется законом, получали согласие от посетителей веб-сайта перед помещением элементов cookie и явных элементов gif/веб-маяков Клиентом или компанией IBM от имени Клиента на устройства посетителя веб-сайта.

Более подробную информацию об использовании для этих целей различных технологий, включая компоненты cookie, смотрите в документе IBM Online Privacy Statement (Заявление об электронной конфиденциальности) по адресу: http://www.ibm.com/privacy/details/us/en, в разделе "Cookies, Web Beacons and Other Technologies" (Элементы cookie, веб-маяки и другие технологии).

Напечатано в Дании