Versão 9 Release 1.2
Maio de 2016

IBM InteractGuia do Administrador

Nota
Antes de usar estas informações e o produto suportado por elas, leia as informações em "Avisos" na página 345.

Esta edição aplica-se à versão 9, liberação 1, modificação 2 do IBM Interact e a todas as liberações e modificações subsequentes, até que seja indicado de outra forma em novas edições.

Índice

Apêndice C. Propriedades de configuração do ambiente de design do Interact 269

Apêndice D. Personalização de oferta em tempo real no lado do cliente . . . 293

Apêndice E. Integração do Interact e do Digital Recommendations 323

Capítulo 1. Administre o IBM Interact

Ao administrar o Interact você configura e mantém usuários e funções, origens de dados e recursos do produto opcionais. Além disso, você monitora e mantém os ambientes de design e de tempo de execução. Há interfaces de programação de aplicativos (APIs) específicas ao produto disponíveis para uso.

A administração do Interact consiste em várias tarefas. Essas tarefas podem incluir, mas não estão limitados a:

- Manter usuários e funções
- Manter origens de dados
- Configurar os recursos opcionais de entrega de oferta do Interact
- Monitorar e manter o desempenho do ambiente de tempo de execução

Antes de começar a administração do Interact, há alguns conceitos principais sobre como o Interact funciona com os quais você deve se familiarizar para facilitar as tarefas. As seções a seguir descrevem as tarefas administrativas associados ao Interact.

A segunda parte do guia de administração descreve as APIs disponíveis com o Interact:

- API do Interact
- API ExternalCallout
- API de aprendizado

Conceitos Chave do Interact

IBM® Interact é um mecanismo interativo que destina ofertas de marketing personalizadas para vários públicos.

Esta seção descreve alguns dos conceitos chave que você deve entender antes de trabalhar com o Interact.

Níveis de Público

Um nível de público é uma coleção de identificadores que podem ser previstos por uma campanha. É possível definir níveis de público para destinar o conjunto correto de públicos para sua campanha.

Por exemplo, um conjunto de campanhas pode usar os níveis de público "Família", "Cliente em potencial", "Cliente" e "Conta". Cada um desses níveis representa uma determinada visualização dos dados de marketing disponíveis para uma campanha.

Níveis de público são tipicamente organizados de forma hierárquica. Usando os exemplos acima:

- Doméstico está no topo da hierarquia, e cada doméstico pode conter diversos clientes e um ou mais prospectivos.
- Cliente é o próximo nível na hierarquia e cada cliente pode ter diversas contas.
- Conta está na parte inferior da hierarquia.

Outros exemplos mais complexos de hierarquias de público existem em ambientes business-to-business, em que níveis de público podem existir para negócios, empresas, divisões, grupos, indivíduos, contas, etc.

Estes níveis de público podem ter diferentes relacionamentos entre si, por exemplo, um para um, muitos para um ou muitos para muitos. Ao definir níveis de público, você permite que esses conceitos sejam representados dentro do Campaign para que os usuários possam gerenciar os relacionamentos entre esses públicos diferentes para propósitos de segmentação. Por exemplo, apesar da possibilidade de existirem diversos clientes em potencial por residência, você pode desejar limitar distribuições a um cliente em potencial por residência.

Ambiente de Design

Use o ambiente de design para configurar vários componentes do Interact e implementá-los no ambiente de tempo de execução.

O ambiente de design é onde você completa a maior parte de sua configuração do Interact. No ambiente de design, você define eventos, pontos de interação, segmentos inteligentes e regras de tratamento. Após configurar estes componentes, implemente-os no ambiente de tempo de execução.

O ambiente de design é instalado com o aplicativo da web do Campaign.

Eventos

Um evento é uma ação que é executada por um visitante e que aciona uma ação no ambiente de tempo de execução. Exemplos de um evento podem ser: colocar um visitante em um segmento, apresentar uma oferta ou registrar dados.

Os eventos são criados primeiro em um canal interativo e, em seguida, acionados por uma chamada para a API do Interact usando o método `postEvent`. Um evento pode levar a uma ou mais das ações a seguir que são definidas no ambiente de design do Interact:

- **Acionar Ressegmentação**. O ambiente de tempo de execução executa todos os fluxogramas interativos para o nível de público atual que está associado ao canal interativo novamente, usando os dados atuais na sessão do visitante.

  Ao projetar a interação, a menos que você especifique um fluxograma específico, uma ação de ressegmentação executará todos os fluxogramas interativos associados a esse canal interativo com o nível de público atual novamente, e as solicitações de ofertas aguardarão até que todos os fluxogramas sejam concluídos. A ressegmentação excessiva em uma visita única pode afetar o desempenho do ponto de contato de maneira visível para o cliente.

  Colocar o cliente em novos segmentos após novos dados significativos está incluído ao objeto de sessão do tempo de execução, como dados novos de solicitações do API Interact (como alterar o público) ou ações do cliente (como incluir itens novos a uma lista de desejos ou carrinho de compras).
- **Contato de Oferta de Log**. O ambiente de tempo de execução sinaliza as ofertas recomendadas para o serviço de banco de dados para efetuar o log das ofertas ao histórico de contato.

  O ponto de contato deve fornecer os códigos de tratamento para as ofertas para as quais registrar contatos. Se for necessário limitar o número de solicitações entre o ponto de contato e o servidor de tempo de execução, será possível registrar o contato de oferta na mesma chamada em que você solicita ofertas, sem fornecer um código de tratamento.

Se o ponto de contato não retornar os códigos de tratamento para as ofertas que o Interact apresenta para o visitante, o ambiente de tempo de execução efetua o log na última lista de ofertas recomendadas.

- **Aceitação de Oferta de Log**. O ambiente de tempo de execução sinaliza a oferta selecionada para o serviço de banco de dados para efetuar o log do histórico de resposta.
- **Rejeição de Oferta de Log**. O ambiente de tempo de execução sinaliza a oferta selecionada para o serviço de banco de dados para efetuar o log do histórico de resposta.
- **Acionar Expressão do Usuário**. Uma *ação de expressão* é uma ação que pode ser definida usando macros do Interact, incluindo funções, variáveis e operadores, incluindo EXTERNALCALLOUT. É possível designar o valor de retorno da expressão a qualquer atributo de perfil.

  Quando você clica no ícone de edição próximo ao Acionar Expressão do Usuário,o diálogo de edição Expressão do Usuário padrão é exibido, e é possível usar este diálogo para especificar um nível de público, nome do campo opcional ao qual designar os resultados e a definição da expressão por si mesma.
- **Acionar Eventos**. É possível usar a função Evento Acionador para inserir um nome de evento que deseja ser acionado por esta ação. Se você inserir um evento que já está definido, este evento é acionado quando esta ação é executada. Se o nome do evento que você inseriu não existe, esta ação causa a criação deste evento com a ação específica.

Também é possível usar eventos para acionar ações que são definidas pelo método `postEvent`, incluindo dados de criação do log em uma tabela, incluindo dados para aprendizado ou acionando fluxogramas individuais.

Os eventos podem ser organizados em categorias para sua conveniência no ambiente de design. As categorias não possuem propósito funcional no ambiente de tempo de execução.

Canais interativos

Use canais interativos no Interact para coordenar todos os objetos, dados e recursos do servidor que estão envolvidos no marketing interativo.

Um canal interativo é uma representação no Campaign de um ponto de contato em que o método da interface é um diálogo interativo. Esta representação de software é usada para coordenar todos os objetos, dados e recursos do servidor que estão envolvidos no marketing interativo.

Um canal interativo é uma ferramenta usada para definir pontos de interação e eventos. Também é possível acessar relatórios para um canal interativo na guia Análise desse canal interativo.

Os canais interativos também contêm designações de servidor de tempo de execução de produção e temporariedade. É possível criar vários canais interativos para organizar seus eventos e pontos de interação se você tiver apenas um conjunto de servidores de runtime de produção e de temporariedade ou para dividir seus eventos e pontos de interação por sistema voltado ao cliente.

Fluxogramas interativos

Use fluxogramas interativos para dividir seus clientes em segmentos e designar um perfil para um segmento.

Um fluxograma interativo está relacionado mas é um pouco diferente de um fluxograma interativo do Campaign. Os fluxogramas interativos executam a mesma função principal que fluxogramas em lote: dividindo seus clientes em grupos conhecidos como segmentos. Para fluxogramas interativos, no entanto, os grupos são segmentos inteligentes. O Interact usa esses fluxogramas interativos para designar um perfil a um segmento quando o evento comportacional ou evento de sistema indicar que uma ressegmentação do visitante é necessária.

Os fluxogramas interativos contêm um subconjunto dos processos de fluxograma em lote e alguns processos específicos do fluxograma interativo.

Nota: Os fluxogramas interativos podem ser criados apenas em uma sessão do Campaign.

Pontos de Interação

Um ponto de interação é um local em seu ponto de contato em que deseja apresentar uma oferta.

Os pontos de interação contêm preenchimento padrão em casos em que o ambiente de tempo de execução não tem outro conteúdo elegível a apresentar. Os pontos de interação podem ser organizados em zonas.

Ofertas

Uma oferta representa uma única mensagem de marketing, que pode ser entregue de várias maneiras.

No Campaign, você cria ofertas que podem ser usadas em uma ou mais campanhas.

Ofertas são reutilizáveis:

- Em campanhas diferentes
- Em momentos diferentes
- Para grupos de pessoas diferentes (células)
- Como "versões" diferentes variando os campos parametrizados da oferta

Você designa ofertas para pontos de interação nos pontos de contato que são apresentados aos visitantes.

Perfis

Um perfil é o conjunto de dados do cliente que é usado pelo ambiente de tempo de execução. Estes dados podem ser um subconjunto dos dados do cliente disponíveis em seu banco de dados de clientes, dados que são coletados em tempo real ou uma combinação dos dois.

Os dados do cliente são usados para os propósitos a seguir:

- Para designar um cliente a um ou mais segmentos inteligentes em cenários de interação em tempo real.

  Você precisa de um conjunto de dados do perfil para cada nível de público pelo qual deseja segmentar. Por exemplo, se você estiver segmentando por local, poderá incluir apenas o código de endereçamento postal do cliente a partir de todas as informações de endereço que você possui.
- Personalizar ofertas

- Como atributos para rastrear para aprendizado

  Por exemplo, é possível configurar o Interact para monitorar o estado civil de um visitante e quantos visitantes de cada status aceitam uma oferta específica. Assim, o ambiente de tempo de execução pode usar essas informações para refinar a seleção de ofertas.

Esses dados são somente de leitura para o ambiente de tempo de execução.

Ambiente de Tempo de Execução

O ambiente de tempo de execução se conecta a seu ponto de contato e executa interações. O ambiente de tempo de execução pode consistir em um ou muitos servidores de runtime que estão conectados a um ponto de contato.

O ambiente de tempo de execução usa as informações implementadas a partir do ambiente de design em combinação com a API do Interact para apresentar ofertas para seu ponto de contato.

Sessões de Tempo de Execução

Uma sessão de tempo de execução existe no servidor de runtime para cada visitante de seu ponto de contato. Esta sessão mantém todos os dados para o visitante que o ambiente de tempo de execução usa para designar visitantes a segmentos e ofertas recomendas.

Você cria uma sessão de tempo de execução quando usa a chamada `startSession`.

Pontos de Contato

Um ponto de contato é um aplicativo ou local em que é possível interagir com um cliente. Um ponto de contato pode ser um canal em que o cliente inicia o contato (uma interação de "entrada") ou em que você entra em contato com o cliente (uma interação de "saída").

Exemplos comuns são websites e aplicativos de central de atendimento. Usando a API do Interact, você pode integrar o Interact com seus pontos de contato para apresentar ofertas aos clientes com base em sua ação no ponto de contato. Os pontos de contato também são chamados de sistemas voltados ao cliente (CFS).

Regras de Tratamento

As regras de tratamento designam um oferta para um segmento inteligente. Essas designações são ainda mais restritas pela zona definida customizada que você associa à oferta na regra de tratamento.

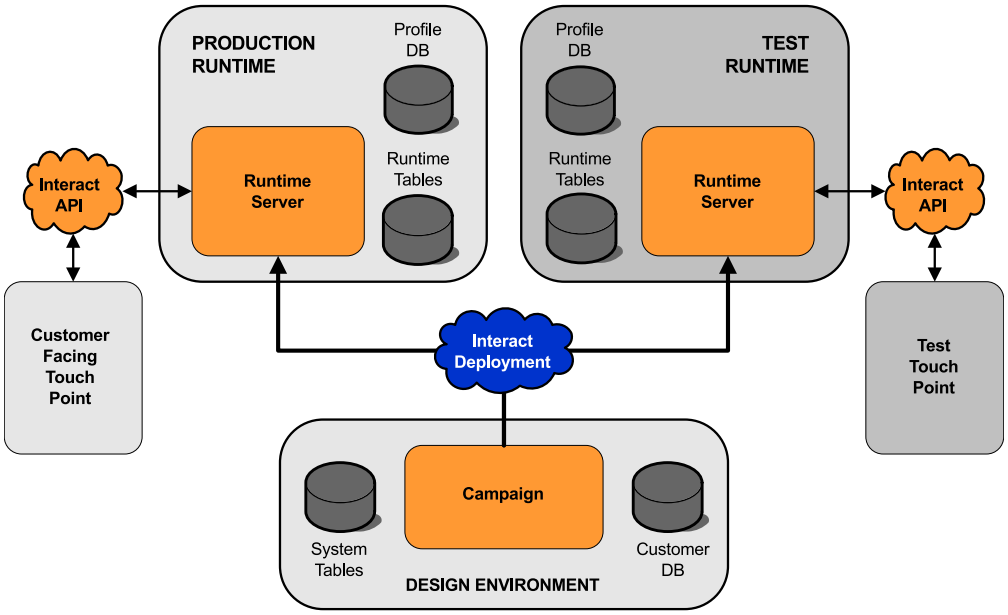
Por exemplo, você possui um conjunto de ofertas ao qual designa um segmento inteligente na zona de "login", mas um conjunto diferente de ofertas para o mesmo segmento na zona "após compra". As regras de tratamento são definidas em uma guia de estratégia de interação de uma campanha.

Cada regra de tratamento também possui uma pontuação de marketing. Se um cliente for designado a mais de um segmento e, dessa forma, mais de uma oferta for aplicável, as pontuações de marketing ajudam a definir quais ofertas o Interact sugere. As ofertas sugeridas pelo ambiente de tempo de execução podem ser influenciadas por um módulo de aprendizado, uma lista de supressão de oferta e designações de ofertas globais e individuais.

Arquitetura do Interact

O ambiente do Interact consiste em no mínimo dois componentes principais, ambiente de design e o ambiente de tempo de execução. Também é possível haver servidores de runtime de teste opcionais.

A figura a seguir mostra a visão geral da arquitetura de alto nível.



O ambiente de design é onde você executa a maioria de sua configuração do Interact. O ambiente de design é instalado com o aplicativo da web do Campaign e faz referência às tabelas do sistema Campaign e aos bancos de dados do cliente. É possível usar o ambiente de design para definir os pontos de interação e eventos usados com a API.

Após projetar e configurar a maneira que você deseja que o ambiente de tempo de execução manipule as interações do cliente, implemente esses dados em um grupo de servidores de temporariedade para teste ou em um grupo de servidores de runtime de produção para interação do cliente em tempo real.

A API do Interact fornece a conexão entre o ponto de contato e o servidor de runtime. Os objetos (pontos de interação e eventos) criados no ambiente de design são referenciados com a API do Interact e usados para solicitar informações do servidor de runtime.

Considerações de rede do Interact

Uma instalação de produção do Interact abrange pelo menos duas máquinas. Em um ambiente de produção de volume alto, com vários servidores de runtime do Interact e bancos de dados distribuídos, a instalação pode abranger dezenas de máquinas.

Para obter um melhor desempenho, existem vários requisitos de topologia de rede a serem considerados.

- Se a implementação da API do Interact iniciar e terminar sessões na mesma chamada, por exemplo:

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

  Não será necessário ativar a persistência de sessão (sessões permanentes) entre o balanceador de carga e os servidores de runtime do Interact. É possível configurar o gerenciamento de sessões dos servidores de runtime do Interact para o tipo de cache local.
- Se a implementação da API usar várias chamadas para iniciar e terminar sessões do Interact, por exemplo:

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

  e você estiver usando um balanceador de carga para os servidores de runtime do Interact, você deverá ativar algum tipo de persistência para o balanceador de carga (também conhecido como sessões persistentes). Se isso não for possível ou se você não estiver usando um balanceador de carga, configure o gerenciamento de sessões dos servidores do Interact para um `cacheType` distribuído. Se você estiver usando um cache distribuído, todos os servidores de runtime do Interact deverão ser capazes de comunicar-se por multicast. Pode ser necessário ajustar a rede para que a comunicação entre os servidores do Interact que usam o mesmo endereço IP e porta multicast não degrade o desempenho do sistema. Um balanceador de carga com sessões persistentes tem melhor desempenho do que o uso de um cache distribuído.
- Se houver vários grupos de servidores que usem um `cacheType` distribuído, cada um deverá usar uma porta multicast exclusiva. É melhor usar uma porta e um endereço multicast exclusivos para cada grupo de servidores.
- Mantenha o ambiente de tempo de execução (servidores do Interact, o Marketing Platform, os balanceadores de carga e o ponto de contato) no mesmo local geográfico para obter o melhor desempenho. O tempo de design e o tempo de execução podem estar em diferentes locais geográficos, mas isso causará lentidão da implementação.
- Tenha uma conexão de rede rápida (pelo menos 1 Gb) entre o grupo de servidores de produção do Interact e seu ponto de contato associado.
- O tempo de design requer acesso HTTP ou HTTPS ao tempo de execução para concluir as tarefas de implementação. Todos os firewalls ou outros aplicativos de rede devem ser configurados para permitir a implementação. Pode ser necessário ampliar as durações de tempo limite de HTTP entre o ambiente de design e os ambientes de tempo de execução, no caso de implementações grandes.
- O módulo de histórico de contatos e respostas requer acesso ao banco de dados de tempo de design (tabelas de sistema do Campaign) e acesso às tabelas de tempo de execução do Interact do banco de dados de tempo de execução). Você deve configurar os bancos de dados e a rede da maneira apropriada para que esta transferência de dados ocorra.

Em uma instalação de teste ou temporária, é possível instalar o tempo de design e o tempo de execução do Interact na mesma máquina. Este cenário não é recomendado para um ambiente de produção.

Efetuando login no IBM EMM

Use este procedimento para efetuar login no IBM EMM.

Antes de Iniciar

É necessário o seguinte.

- Uma conexão de intranet (rede) para acessar seu servidor do IBM EMM.
- Um navegador suportado instalado em seu computador.
- O nome de usuário e a senha para conectar-se ao IBM EMM.
- A URL para acessar o IBM EMM em sua rede.

A URL é:

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

em que

`host` é a máquina em que o Marketing Platform está instalado.

`domain.com` é o domínio no qual a máquina host reside

`port` é o número da porta no qual o servidor de aplicativos do Marketing Platform está atendendo.

Nota: O procedimento a seguir assume que você está efetuando login com uma conta que possui acesso de Administrador ao Marketing Platform.

Procedimento

Acesse a URL do IBM EMM usando seu navegador.

- Se o IBM EMM estiver configurado para integração com o Windows Active Directory ou com uma plataforma de controle de acesso à web, e você estiver conectado a esse sistema, verá a página de painel padrão. Seu login foi concluído.
- Se você vir a tela de login, efetue login usando as credenciais do administrador padrão. Em um ambiente de partição única, use `asm_admin` com `password` como a senha. Em um ambiente com várias partições, use `platform_admin` com `password` como a senha.

  Um prompt solicita que você altere a senha. É possível inserir a senha existente, mas para uma boa segurança, você deve escolher uma nova senha.
- Se o IBM EMM estiver configurado para usar SSL, poderá ser solicitado que você aceite um certificado de segurança digital na primeira vez que se conectar. Clique em **Sim** para aceitar o certificado.

Se seu login for bem-sucedido, o IBM EMM exibirá a página de painel padrão.

Resultados

Com as permissões padrão designadas às contas do administrador do Marketing Platform, é possível administrar contas do usuário e a segurança usando as opções listadas no menu **Configurações**. Para executar as tarefas de administração de nível mais alto para painéis do IBM EMM, você deve efetuar login como **platform_admin**.

Capítulo 2. Configurando usuários do IBM Interact

O Interact requer a configuração de dois conjuntos de usuários, os usuários do ambiente de tempo de execução e os usuários do ambiente de design.

- Os **usuários do tempo de execução** são criados no Marketing Platform configurado para trabalhar com os servidores de runtime.
- Os **usuários do tempo de design** são os usuários do Campaign. Configure a segurança para os vários membros da equipe de design para o Campaign.

Configurando o usuário do ambiente de tempo de execução

Depois de instalar o Interact, você deve configurar pelo menos um usuário do Interact, o usuário do ambiente de tempo de execução. Os usuários do tempo de execução são criados no Marketing Platform.

Sobre Esta Tarefa

O usuário do ambiente de tempo de execução fornece acesso às tabelas de tempo de execução. O usuário do ambiente de tempo de execução é o nome de usuário e a senha usados para implementar os canais interativos. O servidor de runtime usa a autenticação JDBC do servidor de aplicativos da web para as credenciais do banco de dados. Não é necessário incluir nenhuma origem de dados do ambiente de tempo de execução no usuário do ambiente de tempo de execução.

Ao criar usuários de tempo de execução:

- Se houver instâncias do Marketing Platform separadas para cada servidor de runtime, você deverá criar o mesmo usuário e a mesma senha em cada uma. Todos os servidores de runtime que pertencerem ao mesmo grupo de servidores deverão compartilhar as credenciais do usuário.
- Se você usar o utilitário de carregamento de banco de dados, deverá definir as tabelas de tempo de execução como uma origem de dados com credenciais de login para o ambiente de tempo de execução em suas propriedades de configuração em `Interact > general > systemTablesDataSource`.
- Se você ativar a segurança para o monitoramento JMX com o protocolo JMXMP, poderá ser necessário um usuário separado para a segurança do monitoramento JMX.

Consulte a documentação do Marketing Platform para obter as etapas para criar os usuários de tempo de execução.

Configurando usuários do ambiente de design

Os usuários do ambiente de design são usuários do Campaign. Configura os usuários do ambiente de design da mesma maneira que você configura as permissões de função do Campaign.

Sobre Esta Tarefa

Alguns usuários do ambiente de design também requerem algumas permissões do Campaign como Macros customizadas.

Ao criar usuários do ambiente de design:

- Se houver usuários do Campaign com permissão para editar fluxogramas interativos, forneça a eles acesso à origem de dados das tabelas de execução de teste.
- Se o Interact estiver instalado e configurado, as seguintes opções extras estarão disponíveis para a Política global padrão e novas políticas.
-

Categoria	Permissões
Campanhas	• Visualizar Estratégias de Interação de Campanha - Possibilidade de visualizar, mas não de editar, as guias de estratégia de interação em uma campanha. • Editar estratégias de interação de campanha - Capacidade de fazer mudanças nas guias Estratégia de interação, incluindo regras de tratamento. • Excluir Estratégias de Interação de Campanha - Possibilidade de remover as guias de estratégia de interação das campanhas. A exclusão de uma guia Estratégia de interação será restrita se a estratégia de interação tiver sido incluída em uma implementação do canal interativo. • Incluir Estratégias de Interação de Campanha - Possibilidade de criar novas guias de estratégia de interação em uma campanha. • Iniciar Implementações de Estratégia de Interação de Campanha - Possibilidade de marcar uma guia de estratégia de interação para implementação ou remoção de implementação.
Canais interativos	• Implementar canais interativos - Capacidade de implementar um canal interativo em ambientes de tempo de execução do Interact. • Editar canais interativos - Capacidade de fazer mudanças na guia Resumo de canais interativos. • Excluir Canais Interativos - Possibilidade de excluir canais interativos. A exclusão de canais interativos será restrita se o canal interativo tiver sido implementado. • Visualizar Canais Interativos - Possibilidade de ver, mas não editar, os canais interativos. • Incluir Canais Interativos - Possibilidade de incluir novos canais interativos. • Visualizar Relatórios de Canais Interativos - Possibilidade de ver a guia de análise do canal interativo. • Incluir Objetos-filhos do Canal Interativo - Possibilidade de incluir pontos de interação, zonas, eventos e categorias.
Sessões	• Visualizar Fluxogramas Interativos - Possibilidade de ver um fluxograma interativo em uma sessão. • Incluir Interativos - Possibilidade de incluir novos fluxogramas interativos em uma sessão. • Editar fluxogramas interativos - Capacidade de fazer mudanças nos fluxogramas interativos. • Excluir Fluxogramas Interativos - Possibilidade de excluir fluxogramas interativos. A exclusão de fluxogramas interativos será restrita se o canal interativo ao qual esse fluxograma interativo está designado tiver sido implementado. • Copiar Fluxogramas Interativos - Possibilidade de copiar fluxogramas interativos. • Execução de Teste de Fluxogramas Interativos - Possibilidade de iniciar uma execução de teste de um fluxograma interativo. • Revisar Fluxogramas Interativos - Possibilidade de ver um fluxograma interativo e abrir processos para visualizar configurações, mas não de fazer mudanças. • Implementar Fluxogramas Interativos - Possibilidade de marcar fluxogramas interativos para implementação ou remoção de implementação.

Exemplo de permissões do ambiente de design

Este exemplo lista as permissões que são concedidas a duas funções diferentes, uma para usuários que criam fluxogramas interativos e outra para usuários que definem estratégias de interação.

Função do fluxograma interativo

Esta tabela mostra as permissões que são fornecidas à função de fluxograma interativo:

Categoria	Permissão
Macro customizada	A função de usuário possui essas permissões: • Incluir macros customizadas • Editar macros customizadas • Usar macros customizadas
Campo derivado	A função de usuário possui essas permissões: • Incluir campo derivado • Editar campos derivados • Usar campos derivados
Modelo de fluxograma	A função de usuário possui essas permissões: • Colar modelos
Modelo de segmento	A função de usuário possui essas permissões: • Incluir segmentos • Editar segmentos
Sessão	A função de usuário possui essas permissões: • Visualizar resumo de sessão • Visualizar fluxogramas interativos • Incluir fluxogramas interativos • Editar fluxogramas interativos • Copiar fluxogramas interativos • Testar execução de fluxogramas interativos • Implementar fluxogramas interativos

Função de estratégia de interação

Esta tabela mostra as permissões fornecidas à função de estratégia de interação:

Categoria	Permissão
Campanha	A função de usuário possui essas permissões: • Visualizar resumo da campanha • Gerenciar células de destino de campanha • Visualizar estratégias de interação de campanha • Editar estratégias de interação de campanha • Incluir estratégias de interação de campanha • Iniciar implementações de estratégia de interação de campanha
Oferta	A função de usuário possui essas permissões: • Visualizar resumo da oferta

Categoria	Permissão
Modelo de segmento	A função de usuário possui essas permissões: • Visualizar resumo de segmento
Sessão	A função de usuário possui essas permissões: • Revisar fluxogramas interativos

Capítulo 3. Gerenciando origens de dados do Interact

O Interact requer várias origens de dados para funcionar corretamente. Algumas origens de dados contêm as informações que o Interact requer para funcionar, outras origens de dados contêm seus dados.

As seções a seguir descrevem as origens de dados do Interact, incluindo as informações necessárias para configurá-las corretamente e algumas sugestões para mantê-las.

Origens de dados do Interact

O Interact requer vários conjuntos de dados para que funcione. Os conjuntos de dados são armazenados e recuperados a partir de origens de dados e as origens de dados configuradas dependem dos recursos do Interact que estão sendo ativados.

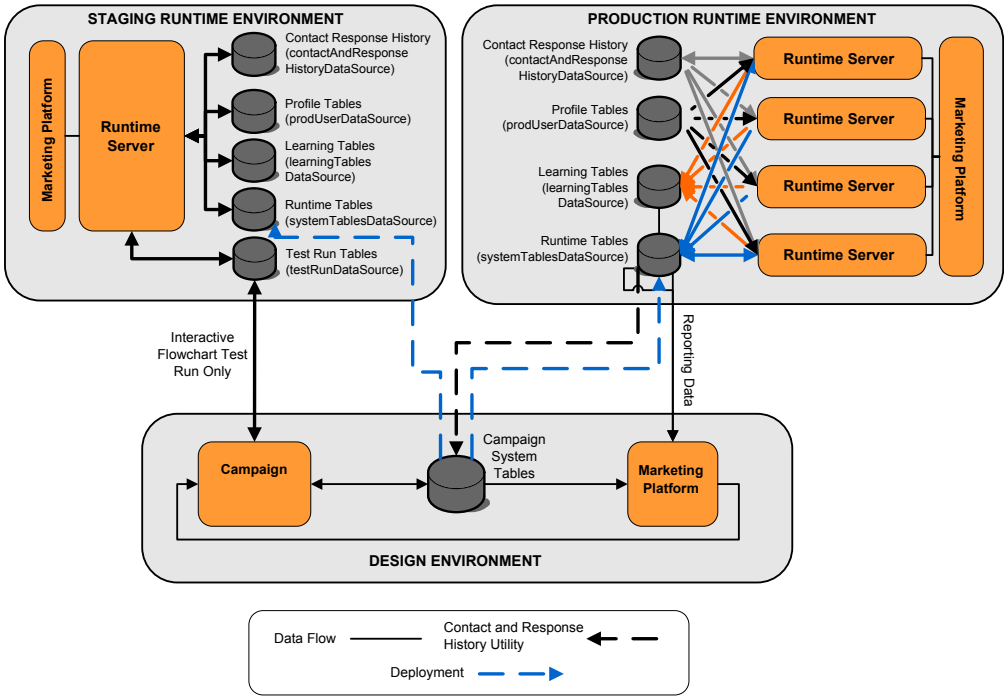
- **Tabelas do sistema do Campaign**. Além de todos os dados do Campaign, as tabelas de sistema do Campaign contêm dados para os componentes do Interact que são criados no ambiente de design, como regras de tratamento e canais interativos. O ambiente de design e as tabelas de sistema do Campaign usam os mesmos banco de dados e esquema físico.
- **Tabelas de tempo de execução**(`systemTablesDataSource`). Essa origem de dados contém os dados de implementação do ambiente de design, as tabelas de migração de histórico de contatos e respostas e as estatísticas de tempo de execução.
- **Tabelas de perfil** (`prodUserDataSource`). Essa origem de dados contém todos os dados do cliente, além de informações reunidas em tempo real que são necessárias para que os fluxogramas interativos coloquem os visitantes em segmentos inteligentes de forma adequada. Se você estiver se baseando totalmente em dados em tempo real, as tabelas de perfil não serão necessárias. Se estiver usando tabelas de perfil, você deverá ter pelo menos uma tabela de perfis por nível de público usado pelo canal interativo.

  As tabelas de perfil também podem conter as tabelas que são usadas para aumentar a entrega de oferta, incluindo tabelas de supressão de oferta, substituição de pontuação e designação de oferta global e individual.
- **Tabelas de execução de teste** (`testRunDataSource`). Essa origem de dados contém uma amostra de todos os dados necessários para que os fluxogramas interativos coloquem os visitantes em segmentos inteligentes, incluindo dados que imitam o que está sendo reunido em tempo real, durante uma interação. Essas tabelas somente são necessárias para o grupo de servidores designado como o grupo de servidores de execução de teste para o ambiente de design.
- **Tabelas de aprendizado** (`learningTablesDataSource`). Esta origem de dados contém todos os dados reunidos pelo utilitário de aprendizado integrado. Essas tabelas podem incluir uma tabela que define atributos dinâmicos. Se você não estiver usando o aprendizado ou estiver usando um utilitário de aprendizado externo que tenha criado, as tabelas de aprendizado não serão necessárias.
- **Histórico de contatos e respostas para resposta de sessão cruzada** (`contactAndResponseHistoryDataSource`). Essa origem de dados contém as tabelas de histórico de contatos do Campaign ou uma cópia delas. Se você não estiver usando o recurso de resposta de sessão cruzada, não será necessário configurar essas tabelas de histórico de contatos.

Bancos de dados e aplicativos

As origens de dados criadas para serem usadas pelo Interact também podem ser usadas para trocar ou compartilhar dados com outros aplicativos IBM EMM.

O diagrama a seguir mostra as origens de dados do Interact e como elas relacionam-se com aplicativos IBM EMM.



- O Campaign e o grupo de servidores de execução de teste acessam as tabelas de execução de teste.
- As tabelas de execução de teste somente são usadas para execuções de teste de fluxograma interativo.
- Quando você estiver usando um servidor de runtime para testar uma implementação, incluindo a API do Interact, o servidor de runtime usará as tabelas de perfis para obter dados.
- Se você configurar o módulo de histórico de contatos e respostas, o módulo usará o processo extrair, transformar e carregar (ETL) de plano de fundo para mover os dados a partir das tabelas de migração de tempo de execução para as tabelas de histórico de contatos e respostas do Campaign.
- A função de relatório consulta os dados das tabelas de aprendizado, das tabelas de tempo de execução e das tabelas de sistema do Campaign para exibir relatórios no Campaign.

Você deve configurar os ambientes de tempo de execução de teste para usar um conjunto diferente de tabelas do que os ambientes de tempo de execução de produção. Com tabelas separadas entre migração e produção, é possível manter os resultados dos testes separados dos resultados reais. Observe que o módulo de histórico de contatos e respostas sempre insere dados nas tabelas de histórico de contatos e respostas reais do Campaign (o Campaign não possui tabelas de histórico de contatos e respostas de teste). Se houver tabelas de aprendizado separadas para o ambiente de tempo de execução de teste e você desejar ver os

resultados em relatórios, será necessária uma instância separada do IBM Cognos BI para executar os relatórios de aprendizado para o ambiente de teste.

Tabelas do sistema Campaign

Ao instalar o ambiente de design do Interact, também são criadas novas tabelas específicas do Interact nas tabelas de sistema do Campaign. As tabelas criadas dependem dos recursos do Interact que estão sendo ativados.

Se você ativar o módulo de histórico de contatos e respostas, o módulo copiará o histórico de contatos e respostas das tabelas de migração nas tabelas de tempo de execução para as tabelas de histórico de contatos e respostas nas tabelas de sistema do Campaign. As tabelas padrão são `UA_ContactHistory`, `UA_DtlContactHist` e `UA_ResponseHistory`, mas o módulo de histórico de contatos e respostas usa quaisquer tabelas que estejam mapeadas no Campaign para as tabelas de histórico de contatos e respostas.

Se você usar as tabelas de ofertas globais e as tabelas de substituição de pontuação para designar ofertas, poderá ser necessário preencher a tabela `UACI_ICBatchOffers` nas tabelas de sistema do Campaign se você estiver usando ofertas que não estejam contidas nas regras de tratamento do Canal interativo.

Tabelas de Tempo de Execução

Se houver mais de um nível de público, você deverá criar as tabelas de migração para os dados do histórico de contatos e respostas para cada nível de público.

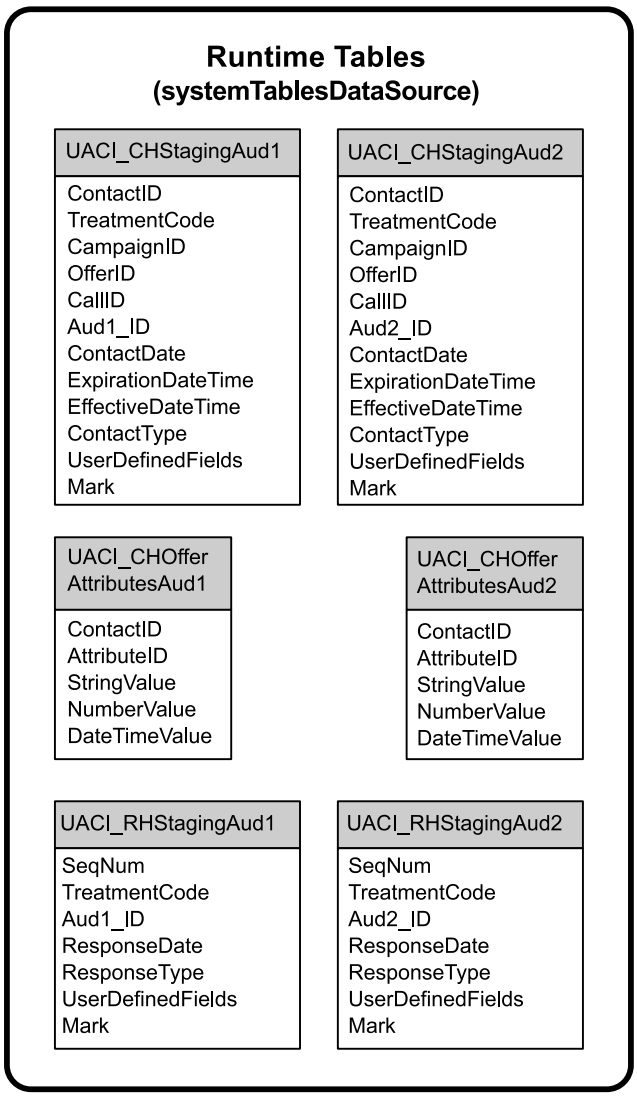
Os scripts SQL criam as seguintes tabelas para o nível de público padrão:

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

Você deve criar cópias dessas três tabelas para cada um dos níveis de público nas tabelas de tempo de execução.

Se as tabelas de históricos de contatos e respostas do Campaign tiverem campos definidos pelo usuário, você deverá criar os mesmos nomes e tipos de campo nas tabelas `UACI_CHStaging` e `UACI_RHStaging`. É possível preencher esses campos durante o tempo de execução criando pares nome-valor do mesmo nome nos dados da sessão. Por exemplo, as tabelas de histórico de contatos e respostas contêm o campo `catalogID`. Você deverá incluir o campo `catalogID` nas tabelas `UACI_CHStaging` e `UACI_RHStaging`. Posteriormente, a API do Interact preencherá esse campo definindo um parâmetro de evento como um par nome-valor chamado `catalogID`. Os dados da sessão podem ser fornecidos pela tabela de perfis, pelos dados temporais, pelo aprendizado ou pela API do Interact.

O diagrama a seguir mostra as tabelas de amostra dos públicos Aud1 e Aud2. Este diagrama não inclui todas as tabelas no banco de dados de tempo de execução.



Todos os campos nas tabelas são necessários. É possível modificar o `CustomerID` e o `UserDefinedFields` para corresponder às tabelas de histórico de contatos e respostas do Campaign.

Tabelas de Execução de Teste

As tabelas de execução de teste somente são usadas para execuções de teste de fluxogramas interativos. As execuções de teste de fluxogramas interativos devem testar a lógica da segmentação. Somente é necessário configurar o banco de dados de execução de teste para a instalação do Interact. As tabelas de execução de teste não precisam estar em um banco de dados independente. Por exemplo, é possível usar as tabelas de dados do cliente para o Campaign.

O usuário do banco de dados associado às tabelas de execução de teste devem ter privilégios CREATE para incluir as tabelas de resultados de execução de teste.

O banco de dados de execução de teste deve conter todas as tabelas mapeadas no canal interativo.

Essas tabelas devem conter dados para executar os cenários que você deseja testar nos fluxogramas interativos. Por exemplo, se os fluxogramas interativos tiverem lógica para classificar as pessoas em segmentos com base na opção selecionada em um sistema de correio de voz, deverá haver pelo menos uma linha para cada seleção possível. Se estiver criando uma interação que funcione com um formulário no website, você deverá incluir linhas que representem os dados que estejam faltando ou mal formados, por exemplo, use `name@domaincom` como o valor de um endereço de email.

Cada tabela de execução de teste deve conter pelo menos uma lista de IDs para o nível de público apropriado e uma coluna representando os dados em tempo real que você espera usar. Como as execuções de teste não possuem acesso aos dados em tempo real, você deverá fornecer dados de amostra para cada parte dos dados em tempo real esperados. Por exemplo, se você desejar usar dados que possam ser coletados em tempo real, por exemplo, o nome da última página da web visitada, armazenado no atributo `lastPageVisited` ou o número de itens em um carrinho de compras, armazenado no atributo `shoppingCartItemCount`, você deverá criar colunas com os mesmos nomes e preenchê-las com dados de amostra. Isto permite executar teste das ramificações da lógica do fluxograma que são comportamentais ou contextuais por natureza.

As execuções de teste de fluxogramas interativos não são otimizadas para trabalhar com grandes conjuntos de dados. É possível limitar o número de linhas usadas para a execução de teste no processo Interação. No entanto, isso sempre resulta na seleção do primeiro conjunto de linhas. Para assegurar-se de que diferentes conjuntos de linhas sejam selecionados, use diferentes visualizações das tabelas de execução de teste.

Para testar o desempenho de rendimento dos fluxogramas interativos no tempo de execução, você deverá criar um ambiente de execução de teste, incluindo uma tabela de perfis para o ambiente de teste.

Na prática, poderão ser necessários três conjuntos de tabelas para teste, uma tabela de execução de teste para execuções de teste de fluxogramas interativos, tabelas de perfis de teste para o grupo de servidores de teste e um conjunto de tabelas de perfis de produção.

Substituindo os tipos de dados padrão usados para tabelas criadas dinamicamente

O ambiente de tempo de execução do Interact cria dinamicamente as tabelas em dois cenários: durante uma execução de teste de um fluxograma e durante a execução de um processo de Captura instantânea que grava em uma tabela que ainda não existe. Para criar essas tabelas, o Interact baseia-se em tipos de dados codificados permanentemente para cada tipo de banco de dados suportado.

É possível substituir os tipos de dados padrão por meio da criação de uma tabela de tipos de dados alternativos, denominada `uaci_column_types`, na `testRunDataSource` ou `prodUserDataSource.` Essa tabela adicional permite que o Interact acomode casos raros que não sejam abrangidos pelos tipos de dados codificados permanentemente.

Quando a tabela `uaci_column_types` estiver definida, o Interact usará os metadados para as colunas como os tipos de dados a serem usados para qualquer geração de

tabela. Se a tabela uaci_column_types não estiver definida ou se houver quaisquer exceções encontradas durante a tentativa de ler a tabela, os tipos de dados padrão serão usados.

Na inicialização, o sistema de tempo de execução primeiro verifica o testRunDataSource para localizar a tabela uaci_column_types. Se a tabela uaci_column_types não existir no testRunDataSource ou se o prodUserDataSource for de um tipo de banco de dados diferente, o Interact verificará se a tabela está no prodUserDataSource.

Substituindo os tipos de dados padrão

Use este procedimento para substituir os tipos de dados padrão para tabelas criadas dinamicamente.

Sobre Esta Tarefa

Você deverá reiniciar o servidor de runtime sempre que alterar a tabela uaci_column_types. Planeje as mudanças de modo que a reinicialização do servidor tenha um efeito mínimo sobre as operações.

Procedimento

1. Crie uma tabela no TestRunDataSource ou ProdUserDataSource com as seguintes propriedades:

   Nome da tabela: uaci_column_types

   Nomes de coluna:

   - uaci_float
   - uaci_number
   - uaci_datetime
   - uaci_string

   Use o tipo de dados apropriado suportado pelo banco de dados para definir cada coluna.
2. Reinicie o servidor de runtime para permitir que o Interact reconheça a nova tabela.

Tipos de dados padrão para tabelas criadas dinamicamente

Para cada banco de dados suportado que o sistema de Runtime do Interact usar, haverá tipos de dados codificados permanentemente usados por padrão para colunas flutuantes, de número, de data/hora e de sequência.

Tabela 1. Tipos de dados padrão para tabelas criadas dinamicamente

Banco de dados	Tipos de dados padrão
DB2	• valor flutuante • bigint • registro de data e hora • varchar
Informix	• valor flutuante • int8 • DATETIME YEAR TO FRACTION • char2
Oracle	• valor flutuante • número(19) • registro de data e hora • varchar2

Tabela 1. Tipos de dados padrão para tabelas criadas dinamicamente (continuação)

Banco de dados	Tipos de dados padrão
SQL Server	• valor flutuante • bigint • data/hora • nvarchar

Banco de dados do perfil

Os conteúdos do banco de dados do perfil dependem inteiramente dos dados necessários para configurar os fluxogramas interativos e da API do Interact. O Interact requer ou recomenda que cada banco de dados contenha determinadas tabelas ou dados.

O banco de dados do perfil deve conter o seguinte:

- Todas as tabelas mapeadas no canal interativo.

  Essas tabelas devem conter todos os dados necessários para executar os fluxogramas interativos em produção. Essas tabelas devem ser simples, aperfeiçoadas e indexadas corretamente. Como há um custo de desempenho para acessar dados dimensionais, você deverá usar um esquema desnormalizado, sempre que possível. No mínimo, você deverá indexar a tabela de perfis nos campos de ID do nível de público. Se houver outros campos recuperados a partir de tabelas dimensionais, eles deverão ser indexados de maneira apropriada para reduzir o tempo de busca do banco de dados. Os IDs de público para as tabelas de perfil devem corresponder aos IDs de Público definidos no Campaign.
- Se você configurar a propriedade de configuração `enableScoreOverrideLookup` como true, você deverá incluir uma tabela de substituição de pontuação para pelo menos um nível de público. Defina os nomes da tabela de substituição de pontuação com a propriedade `scoreOverrideTable`.

  A tabela de substituição de pontuação pode conter pareamentos individuais de cliente para oferta. É possível criar uma tabela de substituição de pontuação de amostra, `UACI_ScoreOverride`, executando o script SQL `aci_usertab` com relação ao banco de dados do perfil. Você também deve indexar essa tabela na coluna ID de público.

  Se você configurar a propriedade `enableScoreOverrideLookup` como false, não será necessário incluir uma tabela de substituição de pontuação.
- Se você configurar a propriedade de configuração `enableDefaultOfferLookup` como true, você deverá incluir a tabela de ofertas globais (`UACI_DefaultOffers`). É possível criar a tabela de ofertas globais executando o script SQL `aci_usertab` com relação ao banco de dados do perfil.

  A tabela de ofertas globais pode conter pareamentos de público para oferta.
- Se você configurar a propriedade `enableOfferSuppressionLookup` como true, você deverá incluir uma tabela de supressão de ofertas para pelo menos um nível de público. Defina os nomes da tabela de supressão de ofertas com a propriedade `offerSuppressionTable`.

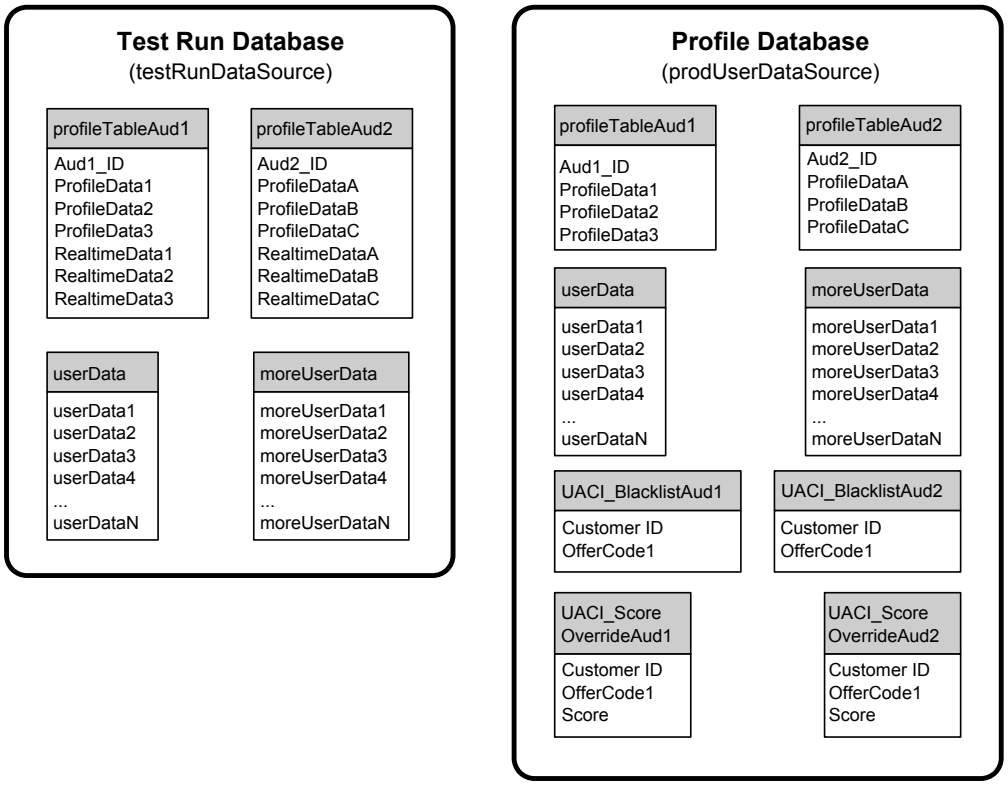
  A tabela de supressão de ofertas pode conter uma linha para cada oferta suprimida por um membro do público, embora não seja necessária uma entrada para cada membro do público. É possível criar uma tabela de supressão de ofertas de amostra `UACI_BlackList` executando o script SQL `aci_usertab` com relação ao banco de dados do perfil.

  Se você configurar a propriedade `enableOfferSuppressionLookup` como false, não será necessário incluir uma tabela de supressão de ofertas.

Uma grande quantia de dados em qualquer uma destas tabelas pode degradar o desempenho. Para obter melhores resultados, coloque índices apropriados nas colunas de nível de público para as tabelas usadas no tempo de execução que possuem grandes quantias de dados.

Todas as propriedades de configuração mencionadas acima estão na categoria **Interact > perfil** ou **Interact > perfil > níveis de público > *AudienceLevel***. O script SQL `aci_usertab` está localizado no diretório `ddl` no diretório de instalação do ambiente de tempo de execução.

O diagrama a seguir mostra as tabelas de exemplo para a execução de teste e os bancos de dados de perfil para o níveis de público Aud1 e Aud2.



Tabelas de Aprendizado

Se estiver usando o aprendizado integrado do Interact, você deverá configurar as tabelas de aprendizado. Essas tabelas contêm todos os dados com os quais o recurso de aprendizado integrado aprende.

Se estiver usando atributos de aprendizado dinâmico, você deverá preencher a tabela `UACI_AttributeList`.

O aprendizado envolve gravar em tabelas de migração intermediárias e agregar informações das tabelas de migração para as tabelas de aprendizado. As propriedades de configuração `insertRawStatsIntervalInMinutes` e `aggregateStatsIntervalInMinutes` na categoria `Interact > offerserving > configuração de aprendizado integrado` determinam com que frequência as tabelas de aprendizado são preenchidas.

O atributo insertRawStatsIntervalInMinutes determina com que frequência as informações de aceitação e contato para cada cliente e oferta são movidas da memória para as tabelas de migração UACI_OfferStatsTX e UACI_OfferTxAll. As informações armazenadas nas tabelas de migração de dados são agregadas e movidas para as tabelas UACI_OfferStats e UACI_OfferStatsAll em intervalos regulares determinados pela propriedade de configuração aggregateStatsIntervalInMinutes.

O aprendizado integrado do Interact usa esses dados para calcular as pontuações finais para ofertas.

Histórico de contatos para rastreamento de resposta de sessão cruzada

Se você ativar o recurso de resposta de sessão cruzada, o ambiente de tempo de execução precisará de acesso somente leitura para as tabelas do histórico de contatos do Campaign. É possível configurar o ambiente de tempo de execução para visualizar as tabelas de sistema do Campaign ou criar uma cópia das tabelas de históricos de contatos do Campaign. Se você criar uma cópia das tabelas, você deverá gerenciar o processo de manter a cópia atualizada. O módulo de histórico de contatos e respostas não atualizará a cópia das tabelas de históricos de contatos.

Você deverá executar o script SQL aci_crhtab com relação a essas tabelas de históricos de contatos para incluir as tabelas necessárias para o recurso de rastreamento de resposta de sessão cruzada.

Executando scripts do banco de dados para ativar os recursos do Interact

Para usar os recursos opcionais disponíveis no Interact, execute scripts de banco de dados com relação ao banco de dados para criar tabelas ou atualizar tabelas existentes.

Sua instalação do Interact, tanto do ambiente de tempo de design quando do ambiente de tempo de execução, inclui scripts **ddl** de recurso. Os scripts **ddl** incluem colunas necessárias em suas tabelas.

Para ativar qualquer um dos recursos opcionais, execute o script adequado com relação ao banco de dados ou à tabela indicada.

dbType é o tipo de banco de dados, como sqlsvr para Microsoft SQL Server, ora para Oracle ou db2 para IBM DB2.

Utilize a tabela a seguir para executar os scripts de banco de dados com relação ao banco de dados para criar tabelas ou atualizar tabelas existentes:

Tabela 2. Scripts de banco de dados

Nome do Recurso	Script de Recurso	Executar Com Relação	Alterar
Ofertas globais, supressão de ofertas substituição de pontuação	**aci_usrtab_***dbType***.sql** em *Interact_Home*\ddl\ acifeatures\ (Diretório de instalação do ambiente de tempo de execução)	Seu banco de dados de perfil (userProdDataSource)	Cria as tabelas UACI_DefaultOffers, UACI_BlackList e UACI_ScoreOverride.

Tabela 2. Scripts de banco de dados (continuação)

Nome do Recurso	Script de Recurso	Executar Com Relação	Alterar
Scoring	**aci_scoringfeature_***dbType***.sql** em *Interact_Home*\ddl\acifeatures\ (Diretório de instalação do ambiente de tempo de execução)	Tabelas de substituição de pontuação no seu banco de dados de perfil (userProdDataSource)	Inclui as colunas LikelihoodScore e AdjExploreScore.
Aprendizado	**aci_lrnfeature_***dbType***.sql** em *Interact_Home*\interactDT\ddl\acifeatures\ (Diretório de instalação do ambiente de tempo de design)	O banco de dados Campaign que contém as tabelas de históricos dos seus contatos	Inclui as colunas RTSelectionMethod, RTLearningMode e RTLearningModelID na tabela UA_DtlContactHist. Também inclui as colunas RTLearningMode e RTLearningModelID na tabela UA_ResponseHistory. O script também é requerido pelos recursos de relatório fornecidos pelo Pacote de Relatórios opcionalInteract.

Sobre o rastreamento de histórico de contatos e respostas

É possível configurar o ambiente de tempo de execução para registrar o histórico de contatos e respostas nas tabelas de histórico de contatos e respostas do Campaign. Os servidor de runtime armazenam o histórico de contatos e respostas em tabelas temporárias. O módulo de histórico de contatos e respostas copia esses dados das tabelas de migração para as tabelas de histórico de contatos e respostas do Campaign.

O módulo de histórico de contatos e respostas somente funcionará se você configurar as propriedades `interactInstalled` e `contactAndResponseHistTracking > isEnabled` na página Configuração do ambiente de design como `sim`.

Se você estiver usando o módulo de rastreamento de resposta de sessão cruzada, o módulo de histórico de contatos e respostas será uma entidade separada.

Tipos de contatos e respostas

É possível registrar um tipo de contato e dois tipos de resposta com o Interact. Também é possível registrar tipos de resposta mais customizados com o método `postEvent`.

Propriedades da tabela contactAndResponseHistTracking

Esta tabela lista as propriedades localizadas na categoria `contactAndResponseHistTracking`:

Evento	Tipo de contato/resposta	Propriedade de configuração
Registrar contato de oferta	Contato	`contacted`
Registrar aceitação de oferta	Resposta	`accept`
Registrar rejeição de oferta	Resposta	`reject`

Propriedades da tabela UA_UsrResponseType

Assegure-se de que a coluna CountsAsResponse da tabela UA_UsrResponseType nas tabelas de sistema do Campaign esteja configurada corretamente. Todos esses tipos de resposta devem existir na tabela UA_UsrResponseType.

Para que seja uma entrada válida na tabela UA_UsrResponseType, você deverá definir um valor para todas as colunas na tabela, incluindo CountsAsResponse. Os valores válidos para CountsAsResponse são:

- 0 - nenhuma resposta
- 1 - uma response
- 2 - uma rejeição
-

Essas respostas são usadas para o relatório.

Tipos de resposta adicionais

No Interact, é possível usar o método postEvent na API do Interact para acionar um evento que registre uma ação "aceitar" ou "rejeitar" para uma oferta. Também é possível aumentar o sistema para permitir que a chamada de postEvent registre tipos de resposta adicionais como Explorar, Considerar, Confirmar ou Cumprir.

Todos esses tipos de resposta devem existir na tabela UA_UsrResponseType nas tabelas de sistema do Campaign. Usando parâmetros de eventos específicos para o método postEvent, é possível registrar tipos de resposta adicionais e definir se uma aceitação deverá ser incluída no aprendizado.

Para registrar tipos de resposta adicionais, você deve incluir os seguintes parâmetros de eventos:

- **UACIResponseTypeCode** - uma sequência que representa um código do tipo de resposta. O valor deve ser uma entrada válida na tabela UA_UsrResponseType.

  Para que seja uma entrada válida no UA_UsrResponseType você deverá definir todas as colunas na tabela, incluindo CountsAsResponse. Os valores válidos para CountsAsResponse são 0, 1 ou 2. 0 indica que não há resposta, 1 indica uma resposta e 2 indica uma rejeição. Essas respostas são usadas para o relatório.
- **UACILogToLearning** - Um número com o valor 1 ou 0. 1 indica que Interact deve criar o log do evento como um aceitação para o sistema de aprendizado ou ativar a supressão de oferta em uma sessão. 0 indica que Interact não deve criar o log do evento no sistema de aprendizado ou ativar a supressão de oferta em uma sessão. Esse parâmetro permite criar vários métodos postEvent que registrem tipos de resposta diferentes sem influenciar o aprendizado. Se você não definir UACILogToLearning, o Interact assumirá o valor padrão 0.

Se um responseTypeCode for fornecido durante a postagem de um evento de aceitação, a oferta não será suprimida na aceitação. Independentemente do valor de ResponseTypeCode (por exemplo 0, 1, 2), se logToLearningAsAccept for 0, a oferta nunca deverá ser suprimida. Para suprimir a oferta o postEvent não deverá especificar o parâmetro UACIResponseTypeCode. Se o parâmetro UACIResponseTypeCode for fornecido, o valor de UACILogToLearning deverá ser 1 se você desejar que a oferta seja suprimida.

É possível criar diversos eventos com a ação Registrar aceitação de oferta, um para cada tipo de resposta que desejar registrar ou um único evento com a ação Registrar aceitação de oferta usado para cada chamada de `postEvent` usada para registrar tipos de resposta separados.

Por exemplo, crie um evento com a ação Log de Aceitação de Oferta para cada tipo de resposta. Você define as respostas customizadas a seguir na tabela `UA_UsrResponseType` [como Nome (código)]: Explorar (EXP), Considerar (CON) e Confirmar (CMT). Em seguida, cria três eventos e os nomeia como LogAccept_Explore, LogAccept_Consider e LogAccept_Commit. Os três eventos são exatamente iguais (têm a ação Registrar aceitação oferta), mas os nomes são diferentes para que a pessoa que trabalha com a API possa distinguí-los.

Ou é possível criar um único evento com a ação Registrar aceitação de oferta usada para todos os tipos de resposta customizada. Por exemplo, nomeie-o como LogCustomResponse.

Ao trabalhar com a API do Interact, não existe diferença funcional entre os eventos, mas as convenções de nomenclatura podem tornar o código mais claro. Além disso, se você der um nome separado a cada resposta customizada, o relatório de Resumo de Atividades de Evento do Canal exibirá informações mais precisas.

Primeiro, configure todos os pares nome-valor

```
//Define name value pairs for the UACIResponseTypeCode
// Response type Explore
NameValuePair responseTypeEXP = new NameValuePairImpl();
responseTypeEXP.setName("UACIResponseTypeCode");
responseTypeEXP.setValueAsString("EXP");
responseTypeEXP.setValueDataType(NameValuePair.DATA_TYPE_STRING);

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

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

//Define name value pairs for UACILOGTOLEARNING
//Does not log to learning
NameValuePair noLogToLearning = new NameValuePairImpl();
noLogToLearning.setName("UACILOGTOLEARNING");
noLogToLearning.setValueAsString("0");
noLogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

//Logs to learning
NameValuePair LogToLearning = new NameValuePairImpl();
LogToLearning.setName("UACILogToLearning");
LogToLearning.setValueAsString("1");
LogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

This first example shows using the individual events.

```
//EXAMPLE 1: This set of postEvent calls use the individually named events
//PostEvent with an Explore response
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogAccept_Explore, postEventParameters);
```

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

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

This second example shows using just one event.

```
//EXAMPLE 2: This set of postEvent calls use the single event
//PostEvent with an Explore response
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogCustomResponse, postEventParameters);

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

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

Ambos os exemplos executam exatamente as mesmas ações, no entanto, uma versão pode ser mais fácil de ler do que a outra.

Tabelas de migração do ambiente de tempo de execução para mapeamento de tabelas de históricos do Campaign

As tabelas de migração de histórico de contato do Interact são mapeadas para tabelas de históricos do Campaign. Você deve ter uma das tabelas de migração do ambiente de tempo de execução para cada nível de público.

Mapeamento de tabela de migração de histórico de contato UACI_CHStaging

Esta tabela mostra como a tabela de migração do ambiente de tempo de execução `UACI_CHStaging` é mapeada para a tabela de histórico de contato do Campaign. Os nomes de tabelas que são mostrados são as tabelas de amostra criadas para o público padrão nas tabelas de tempo de execução e nas tabelas do sistema do Campaign.

Tabela 3. Histórico de contato

UACI_CHStaging **Nome da coluna da tabela de migração do histórico de contato do Interact**	**Tabela de histórico de contato do Campaign**	**Nome da coluna da tabela**
ContactID	N/D	N/D
TreatmentCode	UA_Treatment	TreatmentCode
CampaignID	UA_Treatment	CampaignID
OfferID	UA_Treatment	OfferID
CellID	UA_Treatment	CellID
CustomerID	UA_DtlContactHist	CustomerID
ContactDate	UA_DtlContactHist	ContactDateTime
ExpirationDateTime	UA_Treatment	ExpirationDateTime
EffectiveDateTime	UA_Treatment	EffectiveDateTime
ContactType	UA_DtlContactHist	ContactStatusID
UserDefinedFields	UA_DtlContactHist	UserDefinedFields

ContactID é uma chave para aossicar a tabela UACI_CHOfferAtrib à tabela UACI_CHStaging. A coluna userDefinedFields pode conter quaisquer dados que você escolher.

Mapeamento de tabela de migração de histórico de contato UACI_CHOfferAttrib

Esta tabela mostra como a tabela de migração do ambiente de tempo de execução UACI_CHOfferAttrib é mapeada para a tabela de histórico de contato do Campaign. Os nomes de tabelas que são mostrados são as tabelas de amostra criadas para o público padrão nas tabelas de tempo de execução e nas tabelas do sistema do Campaign.

Tabela 4. Atributos de oferta

UACI_CHOfferAttrib **Nome da coluna da tabela de migração do histórico de contato do Interact**	**Tabela de histórico de contato do Campaign**	**Nome da coluna da tabela**
ContactID	N/D	N/D
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

Mapeamento de tabela de migração de dados de histórico de resposta de contato UACI_RHStaging

Esta tabela mostra como a tabela de migração do ambiente de tempo de execução UACI_RHStaging é mapeada para a tabela de histórico de respostas do Campaign. Os nomes de tabelas que são mostrados são as tabelas de amostra criadas para o público padrão nas tabelas de tempo de execução e nas tabelas do sistema do Campaign.

Tabela 5. Histórico de respostas

UACI_RHStaging **Nome da coluna da tabela de migração de dados de histórico de resposta do Interact**	**Tabela de histórico de resposta do Campaign**	**Nome da coluna da tabela**
SeqNum	N/D	N/D
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime
ResponseType	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

SeqNum é uma chave que é usada pelo módulo de histórico de respostas e contato para identificar dados, mas não é registrado nas tabelas de resposta do Campaign. A coluna userDefinedFields pode conter quaisquer dados que você escolher.

Colunas adicionais em tabelas de migração

Se você incluir colunas em tabelas de migração, o módulo de histórico de respostas e contato as gravará nas tabelas UA_DtlContactHist ou UA_ResponseHistory nas colunas do mesmo nome.

Por exemplo, se você incluir a coluna linkFrom em sua tabela UACI_CHStaging, o módulo de histórico de respostas e contato copiará esses dados na coluna linkFrom na coluna UA_DtlContactHist.

Colunas adicionais nas tabelas de histórico de resposta e contato do Campaign

Se você tiver colunas adicionais em suas tabelas de histórico de resposta e contato do Campaign e as colunas correspondentes às tabelas de migração antes de executar o módulo de histórico de respostas e contato.

Você preenche colunas adicionais nas tabelas de migração criando colunas com os mesmos nomes que seus pares nome-valor em seus dados de sessão de tempo de execução.

Por exemplo, você cria pares nome-valor `NumberItemsInWishList` e `NumberItemsInShoppingCart` e os inclui em sua tabela `UACI_RHStaging`. Quando um evento Aceitação de oferta de log ou Rejeição de oferta de log ocorrer, o ambiente de tempo de execução preencherá estes campos. O ambiente de tempo de execução preenche a tabela `UACI_CHStaging` quando um evento Contato de oferta de log ocorre.

Usar tabelas para incluir uma pontuação para uma oferta

É possível usar os campos definidos pelo usuário para incluir a pontuação que é usada para apresentar uma oferta. Inclua uma coluna que é chamada `FinalScore` para ambas as tabelas `UACI_CHStaging` nas tabelas de tempo de execução e `UA_DtlContactHist` nas tabelas de sistema Campaign. O Interact automaticamente preenche a coluna `FinalScore` com a pontuação final usada para a oferta se você estiver usando o aprendizado integrado.

Se você estiver construindo um módulo de aprendizado customizado, é possível usar o método `setActualValueUsed` da interface `ITreatment` e o método `logEvent` da interface `ILearning`.

Se você não estiver usando o aprendizado, inclua uma coluna que é chamada `Score` para ambas as tabelas `UACI_CHStaging` nas tabelas de tempo de execução e `UA_DtlContactHist` nas tabelas de sistema Campaign. O Interact automaticamente preenche a coluna `Pontuação` com a pontuação usada para a oferta.

Criar novas tabelas de históricos no Campaign e tabelas de migração de dados no Interact

Se estiver usando um nível de público diferente do `Cliente`, você terá que criar novas tabelas de históricos no Campaign e novas tabelas de migração de dados no Interact.

Por exemplo, o script da amostra abaixo é usado no banco de dados de tempo de design do IBM DB2 para criar tabelas de históricos no Campaign para um nível de público do tipo `Conta`.

```
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;
```

O script da amostra abaixo é usado no banco de dados de tempo de execução do IBM DB2 para criar tabelas de migração de dados históricos no Interact para um nível de público do tipo Conta.

```
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);
```

Configurando o monitoramento JMX para o módulo de histórico de contatos e respostas

Use este procedimento para configurar o monitoramento JMX para o módulo de histórico de contatos e respostas. Os protocolos JMXMP e RMI são suportados. A configuração do monitoramento JMX não ativa a segurança para o módulo de histórico de contatos e respostas. É possível usar o Marketing Platform para o ambiente de design para configurar o monitoramento JMX.

Sobre Esta Tarefa

Para usar a ferramenta de monitoramento JMX para o módulo de histórico de contatos e respostas, o endereço padrão usado para:

- O protocolo JMXMP é `service:jmx:jmxmp://`*`CampaignServer`*`:`*`port`*`/campaign`.
- O protocolo RMI é `service:jmx:rmi:///jndi/rmi://`*`CampaignServer`*`:`*`port`*`/campaign`.

Ao visualizar os dados na ferramenta de monitoramento JMX, os atributos de resultados são organizados primeiro por partição e, em seguida, por nível de público.

Procedimento

No Marketing Platform para o ambiente de design, edite as seguintes propriedades de configuração na categoria `Campanha > monitoramento`.

Propriedade de configuração	Configuração
`monitorEnabledForInteract`	**Verdadeiro**
port	O número da porta para o serviço JMX.
protocol	O protocolo a ser usado: • **JMXMP** • **RMI** A segurança não será ativada para o módulo de histórico de contatos e respostas, mesmo se o protocolo JMXMP for selecionado.

Sobre o rastreamento de resposta de sessão cruzada

Os visitantes podem nem sempre concluir uma transação em uma única visita para seu ponto de contato. Um cliente pode incluir um item em seu carrinho de compras em seu website e não concluir a compra até dois dias depois. Manter a sessão de tempo de execução ativa indefinidamente não é viável. É possível ativar o rastreamento de resposta de sessão cruzada para controlar uma apresentação de oferta em uma sessão e correspondê-la a uma resposta em outra sessão.

O rastreamento de resposta de sessão cruzada do Interact pode corresponder em códigos de tratamento ou códigos de oferta por padrão. Também é possível configurá-lo para corresponder qualquer código customizado de sua opção. A resposta de sessão cruzada corresponde aos dados disponíveis. Por exemplo, seu website inclui uma oferta com um código promocional gerado no momento da exibição para um bom desconto para uma semana. Um usuário pode incluir itens em seu carrinho de compras, mas não concluir a compra até três dias depois. Ao usar a chamada `postEvent` para registrar um evento de aceitação, é possível incluir apenas o código promocional. Como o tempo de execução não pode localizar um tratamento ou código de oferta para corresponder na sessão atual, o tempo de execução posiciona um evento de aceitação com as informações disponíveis em uma tabela de migração de resposta de sessão cruzada (XSessResponse). O serviço CrossSessionResponse periodicamente lê a tabela XSessResponse e tenta corresponder os registros aos dados do histórico de contatos disponíveis. O serviço CrossSessionResponse corresponde o código promocional ao histórico de contato e coleta todos os dados necessários para registrar uma resposta adequada. O serviço CrossSessionResponse grava a resposta nas tabelas de migração de resposta e, se o aprendizado estiver ativado, as tabelas de aprendizado. O módulo de histórico de respostas e contato grava a resposta nas tabela de histórico de respostas e contato do Campaign. O processamento bem-sucedido da resposta de sessão cruzada depende dos registros de históricos de contato originais que foram migrados para o banco de dados do Campaign pelo ETL de histórico de contato.

Configuração de origem de dados de rastreamento de resposta de sessão cruzada

O rastreamento de resposta de sessão cruzada do Interact corresponde aos dados da sessão a partir do ambiente de tempo de execução com o histórico de respostas e contato do Campaign. Por padrão, o rastreamento de resposta de sessão cruzada corresponde no código de tratamento ou código de oferta. É possível configurar o ambiente de tempo de execução para corresponder em um código alternativo customizado.

- Se você escolher corresponder em um código alternativo, você deve definir o código alternativo na tabela UACI_TrackingType nas tabelas de tempo de execução do Interact.
- O ambiente de tempo de execução deve ter acesso às tabelas de histórico de contato do Campaign. Isso pode ser ao configurar o ambiente de tempo de execução para ter acesso às tabelas de histórico de contato do Campaign ou criando uma cópia das tabelas de histórico de contato no ambiente de tempo de execução.

  Esse acesso é somente leitura e é separado do utilitário do histórico de respostas e contato.

  Se você criar uma cópia das tabelas, é sua responsabilidade assegurar que os dados na cópia do histórico de contato estejam corretos. É possível configurar a duração que o serviço CrossSessionResponse retém respostas não correspondidas para corresponder a com que frequência você atualiza os dados na cópia das tabelas de histórico de contato usando a propriedade purgeOrphanResponseThresholdInMinutes. Se você estiver usando o módulo de histórico de respostas e contato, deverá coordenar as atualizações de ETL para assegurar que possui os dados mais atuais.

Configurando tabelas de histórico de resposta e contato para rastreamento de resposta de sessão cruzada

Se você criar uma cópia das tabelas de histórico de contato ou usar as tabelas reais nas tabelas do sistema do Campaign, você deve executar as etapas a seguir para configurar as tabelas de histórico de resposta e contato.

Antes de Iniciar

As tabelas de histórico de resposta e contato devem ser mapeadas adequadamente no Campaign antes de executar estas etapas.

Procedimento

1. Execute o script SQL aci_lrnfeature no diretório interactDT/ddl/acifeatures no diretório de instalação do ambiente de design do Interact em relação às tabelas UA_DtlContactHist e UA_ResponseHistory em suas tabelas do sistema Campaign.

   Isso inclui a coluna RTSelectionMethod nas tabelas UA_DtlContactHist e UA_ResponseHistory. Execute o script aci_lrnfeature com relação a essas tabelas para cada um de seus níveis de público. Edite o script conforme necessário para trabalhar com a tabela correta para cada um dos seus níveis de público.
2. Se você desejar copiar as tabelas de histórico de contato no ambiente de tempo de execução, faça-o agora.

   Se estiver criando uma cópia das tabelas de histórico de contato do Campaign acessíveis pelo ambiente de tempo de execução para o suporte de rastreamento de resposta de sessão cruzada, use as diretrizes a seguir:

   - O rastreamento de resposta de sessão cruzada requer acesso somente leitura a essas tabelas.
   - O rastreamento de resposta de sessão cruzada requer as tabelas a seguir do histórico de contato do Campaign.
     - UA_DtlContactHist (para cada nível de público)
     - UA_Treatment

Você deve atualizar os dados nestas tabelas regularmente para assegurar um rastreamento de resposta preciso.

3. Execute o script SQL `aci_crhtab` no diretório `ddl` no diretório de instalação do ambiente de tempo de execução do Interact com relação à origem de dados do histórico de resposta e contato.

   Esse script cria as tabelas `UACI_XsessResponse` e `UACI_CRHTAB_Ver`.

4. Crie uma versão da tabela `UACI_XsessResponse` para cada nível de público.

Resultados

Para melhorar o desempenho do rastreamento de resposta de sessão cruzada, você pode desejar limitar a quantia de dados do histórico de contatos, seja pela forma como copia os dados do histórico de contatos ou configurando uma visualização nas tabelas de histórico de contato do Campaign. Por exemplo, se você tiver uma prática de negócios que nenhuma oferta é válida por mais de 30 dias, você deve limitar os dados do histórico de contatos para os últimos 30 dias. Para modificar o número de dias de dados do histórico de contatos para manter, abra a propriedade de configuração **Campaign | partitions | partition*n* | Interact | contactAndResponseHistTracking** e configure o valor de **daysBackInHistoryToLookupContact**.

Você não verá resultados do rastreamento de resposta de sessão cruzada até que o módulo de histórico de respostas e contato seja executado. Por exemplo, o padrão `processSleepIntervalInMinutes` é 60 minutos. Portanto, pode levar pelo menos uma hora antes de as respostas de sessão cruzada aparecerem em seu histórico de respostas do Campaign.

Tabela UACI_TrackingType

A tabela `UACI_TrackingType` é parte das tabelas de ambiente de tempo de execução. Essa tabela define os códigos de rastreamento usados com rastreamento de resposta de sessão cruzada. O código de rastreamento define que método o ambiente de tempo de execução usa para corresponder a oferta atual em uma sessão de tempo de execução com o histórico de respostas e contato.

Coluna	Tipo	Descrição
`TrackingCodeType`	int	Um número que representa o tipo de código de rastreamento. Esse número é referenciado pelos comandos SQL usados para corresponder informações dos dados da sessão para as tabelas de histórico de respostas e contatos.
`Nome`	varchar(64)	O nome para o tipo de código de rastreamento. É passado para os dados da sessão usando o parâmetro reservado `UACI_TrackingCodeType` com o método `postEvent`.
`Descrição`	varchar(512)	Uma breve descrição do tipo de código de rastreamento. Este campo é opcional.

Por padrão, o ambiente de tempo de execução possui dois tipos de código de rastreamento definidos, conforme mostrado na tabela a seguir. Para qualquer código alternativo, você deve definir um `TrackingCodeType` exclusivo.

TrackingCodeType	Nome	Descrição
1	Código de tratamento	Código de tratamento gerado por UACI
2	Código de oferta	Código de oferta de campanha UAC

UACI_XSessResponse

A tabela `UACI_XSessResponse` é parte das tabelas de ambiente de tempo de execução. Essa tabela é usada para rastreamento de resposta de sessão cruzada.

Uma instância desta tabela para cada nível de público deve existir na origem de dados de histórico de respostas e contato disponível para rastreamento de resposta de sessão cruzada do Interact.

Coluna	Tipo	Descrição
SeqNumber	bigint	Identificador para a linha de dados. O serviço CrossSessionResponse processa todos os registros na ordem SeqNumber.
ICID	bigint	ID de canal interativo
AudienceID	bigint	O ID de público para este nível de público. O nome desta coluna deve corresponder ao ID de público definido no Campaign. A tabela de amostra contém a coluna CustomerID.
TrackingCode	varchar(64)	O valor que é transmitido pelo parâmetro UACIOfferTrackingCode do método postEvent.
TrackingCodeType	int	A representação numérica do código de rastreamento. O valor deve ser uma entrada válida na tabela UACI_TrackingType.
OfferID	bigint	O ID da oferta conforme definido no Campaign.
ResponseType	int	O tipo de resposta para este registro. O valor deve ser uma entrada válida na tabela UA_UsrResponseType.
ResponseTypeCode	varchar(64)	O código do tipo de resposta para este registro. O valor deve ser uma entrada válida na tabela UA_UsrResponseType.
ResponseDate	datetime	A data da resposta.
Mark	bigint	O valor deste campo identifica o estado do registro. • 1 - Em processo • 2 - Bem-sucedido • NULL - Tentar novamente • -1 - O registro está no banco de dados há mais de purgeOrphanResponseThresholdInMinutes minutos. Como parte da manutenção do administrador de banco de dados desta tabela, é possível verificar este campo em busca de registros que não estejam sendo correspondidos, isto é, todos os registros com valor de -1. Todos os registros com valor 2 são removidos automaticamente pelo serviço CrossSessionResponse.
UsrDefinedFields	char(18)	Quaisquer campos customizados que você desejar incluir quando estiver correspondendo respostas de oferta ao histórico de respostas e contato. Por exemplo, se você desejar corresponder em um código promocional, inclua um campo definido pelo usuário de código promocional.

Ativando o rastreamento de resposta de sessão cruzada

Use este procedimento para ativar o rastreamento de resposta de sessão cruzada.

Antes de Iniciar

Você deve configurar o módulo de histórico de respostas e contato para aproveitar ao máximo o rastreamento de resposta de sessão cruzada.

Para usar o rastreamento de resposta de sessão cruzada, você deve configurar o ambiente de tempo de execução para ter acesso de leitura às tabelas de histórico de resposta e contato do Campaign. É possível ler a partir de uma das tabelas reais de histórico de resposta e contato do Campaign no ambiente de tempo de execução ou uma cópia das tabelas nas origens de dados do ambiente de tempo de execução. Configurar o ambiente de tempo de execução para ter acesso de leitura à tabela de histórico de respostas e contato é separado de qualquer configuração de módulo de histórico de respostas e contato.

Se você estiver correspondendo algo que não o código de tratamento ou código de oferta, deve incluí-lo na tabela UACI_TrackingType.

Procedimento

1. Crie as tabelas XSessResponse nas tabelas de histórico de resposta e contato acessíveis ao ambiente de tempo de execução.

2. Defina as propriedades na categoria contactAndResponseHistoryDataSource para o ambiente de tempo de execução.
3. Defina a propriedade crossSessionResponseTable para cada nível de público.
4. Crie uma categoria OverridePerAudience para cada nível de público.

Correspondência de oferta de resposta de sessão cruzada

Por padrão, o rastreamento de resposta de sessão cruzada corresponde códigos de tratamento ou códigos de oferta. O serviço crossSessionResponse usa comandos SQL para corresponder códigos de tratamento, códigos de oferta ou um código customizado de dados da sessão às tabelas de histórico de resposta e contato do Campaign. É possível editar esses comandos SQL para corresponder quaisquer customizações que fizer para seus códigos de rastreamento, códigos de oferta ou códigos customizados.

Correspondendo por código de tratamento

A SQL para corresponder por código de tratamento deve retornar todas as colunas na tabela XSessResponse para este nível de público mais uma coluna chamada OfferIDMatch. O valor na coluna OfferIDMatch deve ser o offerId que acompanha o código de tratamento no registro XSessResponse.

A seguir, uma amostra do comando SQL gerado padrão que corresponde os códigos de tratamento. O Interact gera a SQL para usar os nomes de tabela corretos para o nível de público. Essa SQL é usada se a propriedade Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byTreatmentCode > SQL estiver configurada como **Usar SQL gerado do sistema**.

```
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
```

Os valores UACI_XsessResponse, UA_DtlContactHist, CustomerID e UA_ContactHistory são definidos por suas configurações no Interact. Por exemplo, UACI_XsessResponse é definido pela propriedade de configuração Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable.

Se você tiver customizado suas tabelas de histórico de resposta e contato, pode ser necessário revisar esta SQL para trabalhar com suas tabelas. Você define substituições de SQL na propriedade Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byTreatmentCode > OverrideSQL. Se você fornecer alguma SQL de substituição, você também deverá alterar a propriedade SQL para **Override SQL**.

Correspondendo por código de oferta

A SQL para corresponder por código de oferta deve retornar todas as colunas na tabela XSessResponse para este nível de público mais uma coluna chamada

TreatmentCodeMatch. O valor na coluna TreatmentCodeMatch é o Código de Tratamento que acompanha o ID de Oferta (e Código de Oferta) no registro XSessResponse.

A seguir, uma amostra do comando SQL gerado padrão que corresponde a códigos de oferta. O Interact gera a SQL para usar os nomes de tabela corretos para o nível de público. Esta SQL é usada se a propriedade Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byOfferCode > SQL estiver configurada como **Usar SQL gerado do sistema**.

```
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
```

Os valores UACI_XsessResponse, UA_DtlContactHist, CustomerID e UA_ContactHistory são definidos por suas configurações no Interact. Por exemplo, UACI_XsessResponse é definido pela propriedade de configuração Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable.

Se você tiver customizado suas tabelas de histórico de resposta e contato, pode ser necessário revisar esta SQL para trabalhar com suas tabelas. Você define substituições de SQL na propriedade Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byOfferCode > OverrideSQL. Se você fornecer alguma SQL de substituição, você também deverá alterar a propriedade SQL para **Override SQL**.

Correspondendo por código alternativo

É possível definir um comando SQL para corresponder por algum código alternativo de sua opção. Por exemplo, você pode ter códigos promocionais ou códigos do produto separados dos códigos de oferta ou tratamento.

Você deve definir esse código alternativo na tabela UACI_TrackingType nas tabelas de ambiente de tempo de execução do Interact.

Você deve fornecer uma SQL ou um procedimento armazenado na propriedade Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byAlternateCode > OverrideSQL que retorna todas as colunas na tabela XSessResponse para este nível de público mais as colunas TreatmentCodeMatch e OfferIDMatch. É possível como opção retornar offerCode no lugar de OfferIDMatch (na forma de offerCode1, offerCode2, ... offerCode*N* para N códigos de oferta da peça). Os valores na coluna TreatmentCodeMatch e OfferIDMatch (ou colunas de código de oferta) devem corresponder a TrackingCode no registro XSessResponse.

Por exemplo, o pseudocódigo SQL a seguir corresponde na coluna AlternateCode na tabela 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>
```

Em que *<x>* é o código de rastreamento definido na tabela UACI_TrackingType.

Usando um utilitário de carregamento de banco de dados com o ambiente de tempo de execução

Por padrão, o ambiente de tempo de execução grava dados de histórico de respostas e contatos de dados da sessão em tabelas de migração. Em um sistema de produção muito ativo, no entanto, a quantia de memória necessária para armazenar em cache todos os dados antes de o tempo de execução poder gravá-los nas tabelas de migração pode ser proibitiva. É possível configurar o tempo de execução para usar um utilitário de carregamento de banco de dados para melhorar o desempenho.

Ao ativar um utilitário de carregamento de banco de dados, em vez de manter todo o histórico de respostas e contatos na memória antes de gravar nas tabelas de migração, o tempo de execução grava os dados em um arquivo temporário. Você define o local do diretório que contém os arquivos temporários com a propriedade externalLoaderStagingDirectory. Esse diretório contém vários subdiretórios. O primeiro subdiretório é o diretório de instâncias de tempo de execução, que contém os diretórios contactHist e respHist. Os diretórios contactHist e respHist

contêm subdiretórios denominados exclusivamente no formato de *audienceLevelName.uniqueID.currentState*, que contêm arquivos temporários.

Estado Atual	Descrição
CACHE	Conteúdo do diretório que está sendo gravado atualmente em um arquivo.
READY	Conteúdo do diretório pronto para ser processado.
RUN	Conteúdo do diretório que está sendo gravado atualmente no banco de dados.
PROCESSED	Conteúdo do diretório que foi gravado no banco de dados.
ERROR	Ocorreu um erro ao gravar o conteúdo do diretório no banco de dados.
ATTN	Conteúdo da atenção de necessidade do diretório. Isto é, pode ser necessário executar algumas etapas para concluir a gravação do conteúdo deste diretório no banco de dados.
RERUN	Conteúdo do diretório pronto para ser gravado no banco de dados. Você deve renomear um diretório de `ATTN` ou `ERROR` para `RERUN` após ter corrigido o problema.

É possível definir o diretório de instâncias de tempo de execução definindo a propriedade JVM interact.runtime.instance.name no script de inicialização do servidor de aplicativos. Por exemplo, você poderia incluir `-Dinteract.runtime.instance.name=instance2` em seu script de inicialização do servidor de aplicativos da web. Se não configurado, o nome padrão será `DefaultInteractRuntimeInstance`.

O diretório `samples` contém arquivos de amostra para auxiliá-lo na gravação de seus próprios arquivos de controle do utilitário de carregamento de banco de dados.

Ativando um utilitário de carregamento de banco de dados com o ambiente de tempo de execução

Use este procedimento para ativar um utilitário de carregamento de banco de dados com o ambiente de tempo de execução.

Antes de Iniciar

Você deverá definir os comandos ou arquivos de controle para o utilitário de carregamento de banco de dados antes de configurar o ambiente de tempo de execução para usá-los. Esses arquivos devem existir no mesmo local em todos os servidores de runtime no mesmo grupo de servidores.

O Interact fornece comandos de amostra e arquivos de controle no diretório `loaderService` na instalação do servidor de runtime do Interact.

Procedimento

1. Confirme se o usuário do ambiente de tempo de execução possui credenciais de login para a origem de dados das tabelas de tempo de execução que está definida em `Interact > general > systemTablesDataSource` nas propriedades de configuração.
2. Defina as propriedades de configuração `Interact > geral > systemTablesDataSource > loaderProperties`.
3. Defina a propriedade `Interact > serviços >externalLoaderStagingDirectory`.
4. Revise as propriedades de configuração `Interact > services > responseHist > fileCache`, caso seja necessário.
5. Revise as propriedades de configuração `Interact > services > contactHist > fileCache`, caso seja necessário.
6. Reinicie o servidor de runtime.

Processo ETL de padrão de evento

Para processar grandes quantias de dados de padrão de evento do IBM Interact e para disponibilizar esses dados para propósitos de consultas e relatórios, é possível instalar um processo extrair, transformar e carregar (ETL) independente em qualquer servidor suportado para obter um desempenho ideal.

No Interact, todos os dados de padrão de evento para um determinado AudienceID são armazenados como uma única coleção nas tabelas de banco de dados de tempo de execução. As informações de AudienceID e de estado padrão são armazenadas como um objeto binário grande (BLOB). Para executar quaisquer consultas SQL ou relatórios com base nos padrões de evento, esse novo processo ETL será necessário para dividir o objeto em tabelas dentro de um banco de dados de destino. Para isso, o processo ETL independente obtém os dados de padrão de evento a partir das tabelas de banco de dados de runtime do Interact, processa-os no planejamento especificado e armazena-os no banco de dados de destino disponível para consultas SQL ou relatórios adicionais.

Além de mover e transformar os dados de padrão de evento para o banco de dados de destino, o processo ETL independente também sincroniza os dados no banco de dados de destino com as informações mais atuais no banco de dados de tempo de execução do Interact. Por exemplo, se você excluir um padrão de evento no tempo de execução do Interact, os dados processados desse padrão de evento serão removidos do banco de dados de destino na próxima vez em que o processo ETL for executado. As informações do estado de padrão de evento também serão mantidas atualizadas. Portanto, as informações armazenadas sobre os padrões de evento no banco de dados de destino serão somente de dados atuais, não informações históricas.

Executando o processo ETL independente

Quando o processo ETL independente é ativado em um servidor, ele é executado continuamente no plano de fundo até que seja interrompido. O processo segue as instruções nas propriedades de configuração do Marketing Platform para determinar a frequência, as conexões de banco de dados e outros detalhes durante sua operação.

Antes de Iniciar

Antes de executar o processo ETL independente, assegure-se de concluir as tarefas a seguir:

- Deve-se ter as permissões de uma função de usuário administrador do Interact.
- Deve-se ter instalado o processo em um servidor e configurado os arquivos no servidor e no Marketing Platform corretamente para sua configuração.

Nota:

Se estiver executando o processo ETL no Microsoft Windows em um idioma diferente do inglês dos Estados Unidos, use `chcp` no prompt de comandos para configurar a página de códigos para o idioma em uso. Por exemplo, você poderá usar qualquer um dos códigos a seguir: ja_jp=932, zh_cn=936, ko_kr=949, ru_ru=1251 e para de_de, fr_fr, it_it, es_es, pt_br, use 1252. Para assegurar a exibição de caracteres adequada, use o comando `chcp` no prompt de comandos do Windows antes de ativar o processo ETL.

Sobre Esta Tarefa

Após instalar e configurar o processo ETL independente, você estará pronto para ativar o processo.

Procedimento

1. Abra um prompt de comandos no servidor em que o processo ETL está instalado.
2. Navegue para o diretório `<Interact_home>/PatternStateETL/bin` que contém os arquivos executáveis para o processo ETL.
3. Execute o arquivo `command.bat` (no Microsoft Windows) ou o arquivo `command.sh` (nos sistemas operacionais como o UNIX) com os parâmetros a seguir:
   - `-u <username>`. Esse valor deve ser um usuário válido do Marketing Platform e você deve configurar esse usuário com acesso às origens de dados **TargetDS** e **RuntimeDS** que o processo ETL usará.
   - `-p <password>`. Substitua o *<password>* pela senha correspondente ao usuário especificado. Se a senha para esse usuário estiver em branco, especifique duas aspas duplas (como em `-p ""`). A senha é opcional ao executar o arquivo de comandos. Se você omitir a senha com o comando, será solicitado que ela seja inserida quando o comando for executado.
   - `-c <profileName>`. Substitua o *<profileName>* pelo nome exato especificado no Marketing Platform na configuração **Interact | PatternStateETL** que foi criada.

     O nome inserido aqui deve corresponder ao valor especificado no campo **Nome da nova categoria** quando você criou a configuração.
   - `start`. O comando start é necessário para iniciar o processo.

   O comando completo para iniciar o processo possui o seguinte formato:

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

Resultados

O processo ETL independente é executado e continua em execução no plano de fundo até que o processo seja parado ou até que o servidor seja reiniciado.

Nota:

Na primeira vez que você executar o processo, os dados padrão de evento acumulados podem levar um tempo considerável para serem executados. As vezes subsequentes em que o processo for executado funcionarão somente com o conjunto mais recente de dados de padrão de evento e levão menos tempo para serem concluídas.

Esteja ciente de que também é possível fornecer o argumento `help` para o arquivo `command.bat` ou `command.sh` para ver todas as opções disponíveis, como no exemplo a seguir:

```
command.bat help
```

Parando o processo ETL independente

Quando o processo ETL independente é ativado em um servidor, ele é executado continuamente no plano de fundo até que seja interrompido.

Sobre Esta Tarefa

Procedimento

1. Abra um prompt de comandos no servidor em que o processo ETL está instalado.
2. Navegue para o diretório `<Interact_home>/PatternStateETL/bin` que contém os arquivos executáveis para o processo ETL.
3. Execute o arquivo `command.bat` (no Microsoft Windows) ou o arquivo `command.sh` (nos sistemas operacionais como o UNIX) com os parâmetros a seguir:
   - `-u <username>`. Esse valor deve ser um usuário válido do Marketing Platform e você deve ter configurado esse usuário com acesso às origens de dados **TargetDS** e **RuntimeDS** que o processo ETL usará.
   - `-p <password>`. Substitua o <*password*> pela senha correspondente ao usuário especificado. Se a senha para esse usuário estiver em branco, especifique duas aspas duplas (como em `-p ""`). A senha é opcional ao executar o arquivo de comandos. Se você omitir a senha com o comando, será solicitado que ela seja inserida quando o comando for executado.
   - `-c <profileName>`. Substitua o <*profileName*> pelo nome exato especificado no Marketing Platform na configuração **Interact | PatternStateETL** que foi criada.

     O nome inserido aqui deve corresponder ao valor especificado no campo **Nome da nova categoria** quando você criou a configuração.
   - `stop`. O comando parar é necessário para parar o processo. Se você usar esse comando, todas as operações ETL em andamento serão concluídas antes que o processo seja encerrado.

     Para encerrar o processo ETL sem aguardar a conclusão das operações em andamento, use `forcestop` em vez de `stop`.

   O comando completo para iniciar o processo possui o seguinte formato:

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

Resultados

O processo ETL independente é parado.

Capítulo 4. Entrega de oferta

É possível configurar o Interact de muitas formas para aprimorar como ele seleciona as ofertas para apresentar. As seções a seguir descrevem estes recursos opcionais em detalhes.

Elegibilidade da oferta

O propósito do Interact é apresentar ofertas elegíveis. O Interact simplesmente apresenta a mais ideal entre as ofertas elegíveis, com base no visitante, canal e situação.

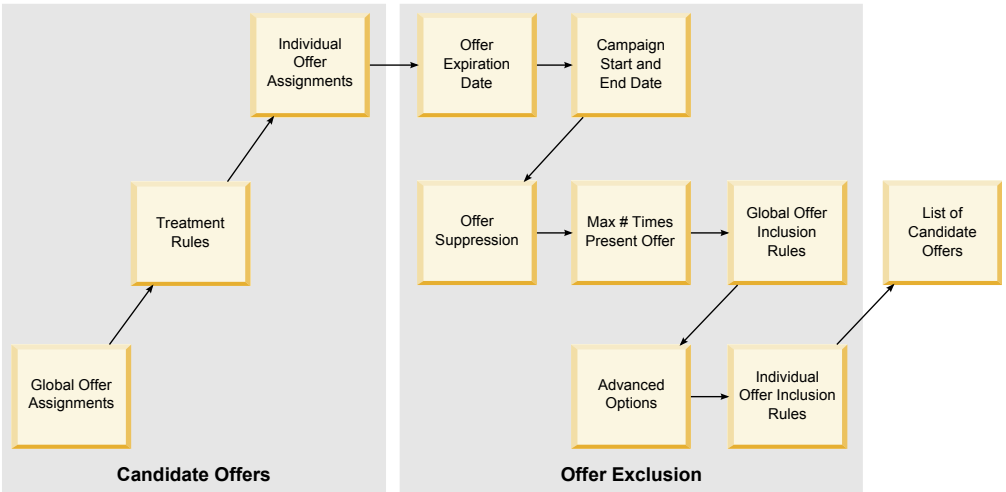
As regras de tratamento são apenas o início de como o Interact determina que ofertas são elegíveis para um cliente. O Interact possui vários recursos opcionais que podem ser implementados para aprimorar como o ambiente de tempo de execução determina que ofertas apresentar. Nenhum desses recursos garante que uma oferta seja apresentada a um cliente. Esses recursos influenciam a probabilidade de uma oferta ser elegível a ser apresentada a um cliente. É possível usar quantos recursos forem necessários para implementar a melhor solução para seu ambiente.

Há três principais áreas em que é possível influenciar a elegibilidade da oferta: gerar a lista de ofertas de candidatos, determnar a pontuação de marketing e aprender.

Gerando uma lista de ofertas candidatas

A geração de uma lista de ofertas candidatas possui dois principais estágios. O primeiro estágio é gerar uma lista de todas as possíveis ofertas para as quais o cliente pode estar elegível. O segundo estágio é filtrar qualquer oferta para a qual o cliente não está mais elegível. Há vários lugares em ambos os estágios em que é possível influenciar a geração da lista de ofertas candidatas.

Este diagrama mostra os estágios da geração de lista de ofertas candidatas. As setas mostram a ordem de precedência. Por exemplo, se uma oferta passar no filtro **Nº máximo de vezes para apresentar uma oferta**, mas falhar no filtro **Regras de inclusão de oferta global**, o ambiente de tempo de execução excluirá a oferta.

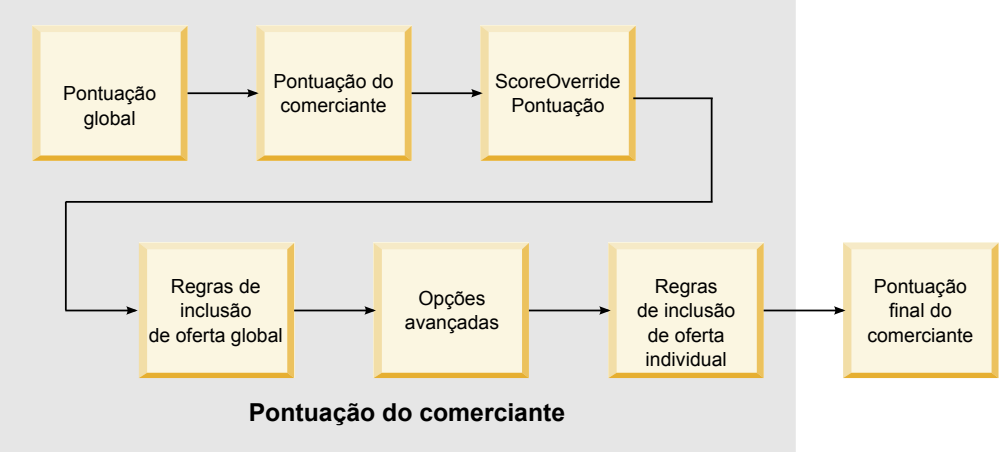


- **Designações de oferta global** - É possível definir ofertas globais por nível de público usando a tabela de ofertas globais.
- **Regras de tratamento** - O método básico para definir ofertas por segmento pelo ponto de interação usando a guia Estratégia de interação.
- **Designações de oferta individual** - É possível definir designações de oferta específicas por cliente usando a tabela de substituição de pontuação.
- **Data de expiração de oferta** - Ao criar uma oferta no Campaign, é possível definir uma data de expiração. Se a data de expiração para uma oferta tiver passado, o ambiente de tempo de execução excluirá a oferta.
- **Data de início e encerramento da campanha** - Ao criar uma campanha no Campaign, é possível definir uma data de início e encerramento para a campanha. Se a data de início para a campanha não tiver ocorrido ou a data de encerramento para a campanha tiver passado, o ambiente de tempo de execução excluirá a oferta.
- **Supressão da oferta** - É possível definir a supressão da oferta para membros do público específicos usando a tabela de supressão de ofertas.
- **Nº máximo de vezes para apresentar uma oferta** - Ao definir um canal interativo, você define o número máximo de vezes para apresentar uma oferta a um cliente por sessão. Se a oferta já tiver sido apresentada este número de vezes, o ambiente de tempo de execução excluirá a oferta.
- **Regras de inclusão de oferta global** - É possível definir uma expressão booleana para filtrar ofertas em um nível de público usando a tabela de ofertas globais. Se o resultado for false, o ambiente de tempo de execução excluirá a oferta.
- **Opções avançadas** - É possível usar a opção avançada **Considerar esta regra elegível se a expressão a seguir for true** em uma regra de tratamento para filtrar ofertas em um nível de segmento. Se o resultado for false, o ambiente de tempo de execução excluirá a oferta.
- **Regras de inclusão de oferta individual** - É possível definir uma expressão booleana para filtrar ofertas em um nível do cliente usando a tabela de substituição de pontuação. Se o resultado for false, o ambiente de tempo de execução excluirá a oferta.

Calcular a pontuação de marketing

Há muitas formas de influenciar (usando um cálculo) ou substituir a pontuação de marketing.

Este diagrama mostra os diferentes estágios em que é possível influenciar ou substituir a pontuação de marketing.



As setas mostram a ordem de precedência. Por exemplo, se você definir uma expressão para determinar a pontuação de marketing nas Opções Avançadas para uma regra de tratamento e definir uma expressão na tabela de substituição de pontuação, a expressão na tabela de substituição de pontuação terá precedência.

- **Pontuação global** - É possível definir uma pontuação por nível de público usando a tabela de ofertas globais.
- **Pontuação do comerciante** - É possível definir uma pontuação por segmento usando a régua de controle em uma regra de tratamento.
- **Pontuação de Substituição de pontuação** - É possível definir uma pontuação por cliente usando a tabela de substituição de pontuação.
- **Regras de inclusão de oferta global** - É possível definir uma expressão que calcula uma pontuação por nível de público usando a tabela de ofertas globais.
- **Opções Avançadas** - É possível definir uma expressão que calcula uma pontuação por segmento usando a opção avançada **Use a expressão a seguir como pontuação de marketing** em uma regra de tratamento.
- **Regras de inclusão de oferta de substituição de pontuação** - É possível definir uma expressão que calcula uma pontuação por cliente usando a tabela de substituição de pontuação.

Influenciando o aprendizado

Se você estiver usando o módulo de aprendizado integrado do Interact, é possível influenciar a saída de aprendizado além das configurações de aprendizado padrão, como a lista de atributos de aprendizado ou o nível de confiança. É possível substituir componentes do algoritmo de aprendizado ao usar os componentes restantes.

É possível substituir o aprendizado usando as colunas `LikelihoodScore` e `AdjExploreScore` das ofertas padrão e tabelas de substituição de pontuação. É possível incluir essas colunas nas ofertas padrão e tabelas de substituição de pontuação usando o script de recurso `aci_scoringfeature`. Para usar essas substituições adequadamente, é necessário um entendimento completo do aprendizado integrado do Interact.

O módulo de aprendizado obtém a lista de ofertas de candidato e a pontuação de marketing por oferta de candidato e as usa nos cálculos finais. A lista de ofertas é usada com os atributos de aprendizado para calcular a probabilidade (probabilidade de aceitação) que o cliente aceitará a oferta. Usando essas probabilidades e o número histórico de apresentações para equilibrar entre a exploração e a explotação, o algoritmo de aprendizado determina o peso da oferta. Finalmente, o aprendizado integrado usa o peso da oferta, multiplica-o pela pontuação de marketing final e retorna uma pontuação final. As ofertas são classificadas por essa pontuação final.

Suprimir ofertas

É possível configurar o ambiente de tempo de execução para suprimir ofertas.

Há várias maneiras pelas quais o ambiente de tempo de execução suprime uma oferta:

- O elemento **Nº máximo de vezes para mostrar qualquer oferta durante uma única visita** de um canal interativo.

  Você define o **Número máximo de vezes para mostrar qualquer oferta durante uma única visita** ao criar ou editar um canal interativo.

- O uso de uma tabela de supressão de ofertas.

  Você cria uma tabela de supressão de ofertas em seu banco de dados do perfil.
- Ofertas cuja data de expiração passou.
- Ofertas de campanhas expiradas.
- Ofertas excluídas, porque não passam uma regra de tratamento de oferta (opção avançada de regra de tratamento).
- Ofertas já explicitamente aceitas ou rejeitadas em uma sessão do Interact. Se um cliente explicitamente aceitar ou rejeitar uma oferta, essa oferta será suprimida durante a sessão.

Ativando a supressão da oferta

Use este procedimento para ativar a supressão da oferta.

Sobre Esta Tarefa

É possível configurar o Interact para fazer referência a uma lista de ofertas suprimidas.

Procedimento

1. Crie um `offerSuppressionTable`, uma nova tabela para cada público que contiver o ID de público e o ID da oferta.
2. Configure a propriedade `enableOfferSuppressionLookup` para **true**.
3. Configure a propriedade `Interact > profile > offerSuppressionTable` com o nome da tabela de supressão de ofertas do público apropriado.

Tabela de supressão da oferta

A tabela de supressão de ofertas permite suprimir uma oferta para um ID de público específico. Por exemplo, se seu público for Cliente, é possível suprimir uma oferta para o cliente John Smith. Uma versão desta tabela para pelo menos um nível de público deve existir em seu banco de dados do perfil de produção. É possível criar uma tabela de supressão de ofertas de amostra, `UACI_BlackList` executando o script SQL `aci_usertab` em relação a seu banco de dados do perfil. O script SQL `aci_usertab` está localizado no diretório `ddl` em seu diretório de instalação do ambiente de tempo de execução.

Você deve definir os campos `AudienceID` e `OfferCode1` para cada linha. É possível incluir colunas adicionais se seu ID de público ou Código de oferta consistir em diversas colunas. Essas colunas devem corresponder aos nomes da coluna definidos no Campaign. Por exemplo, se você definir o público `Cliente` pelos campos `HHold_ID` e `MemberNum`, você deve incluir `HHold_ID` e `MemberNum` na tabela de supressão de ofertas.

Nome	Descrição
AudienceID	(Necessário) O nome desta coluna deve corresponder ao nome da coluna que define o ID de público no Campaign. Se seu ID de público consistir em diversas colunas, é possível incluí-los nesta tabela. Cada linha deve conter o ID de público ao qual você designa a oferta padrão, por exemplo, `customer1`.
OfferCode1	(Necessário) O código de oferta para a oferta que está substituindo. Se seus códigos de oferta forem feitos de diversos campos, é possível incluir colunas adicionais, por exemplo, `OfferCode2`, e assim por diante.

Ofertas globais e designações individuais

É possível configurar o ambiente de tempo de execução para designar ofertas específicas além das regras de tratamento configuradas na guia Estratégia de interação. é possível definir ofertas globais para qualquer membro de um nível de público e designações individuais para membros do público específicos. Por exemplo, é possível definir uma oferta global para que toda a família veja quando outras não estiverem disponíveis e criar uma designação de oferta individual para a família Smith específica.

É possível restringir as ofertas globais e as designações individuais por zona, célula e regras de inclusão de oferta. Ambas as ofertas globais e designações individuais são configuradas incluindo dados em tabelas específicas em seu banco de dados do perfil de produção.

Para ofertas globais e designações individuais funcionem adequadamente, todas as células referenciadas e códigos de oferta devem existir na implementação. Para assegurar que os dados necessários estejam disponíveis, você deve configurar os códigos de célula padrão e a tabela `UACI_ICBatchOffers`.

Definindo os códigos de célula padrão

Se você usar as ofertas padrão ou tabelas de substituição de pontuação para designações de ofertas globais ou individuais, você deverá definir os códigos de célula padrão. O `DefaultCellCode` é usado quando não há código de célula definido em uma determinada linha nas ofertas padrão ou tabelas de intervenção de pontuação. O relatório usa esse código de célula padrão.

Sobre Esta Tarefa

O `DefaultCellCode` deve corresponder ao formato do código de célula que está definido no Campaign. Esse código de célula é usado para todas as designações de oferta que aparecem no relatório.

Se você definir códigos de célula padrão exclusivos, será possível identificar facilmente ofertas designadas pelas ofertas padrão ou tabelas de substituição de pontuação.

Procedimento

Defina a propriedade `DefaultCellCode` para cada nível de público e tipo de tabela na categoria `IndividualTreatment`.

Definindo ofertas não usadas em uma regra de tratamento

Se você usar as ofertas padrão ou tabelas de substituição de pontuação, deve assegurar que todos os códigos de oferta existam na implementação. Se você souber que todas as ofertas que você usa nas ofertas padrão ou tabelas de substituição de pontuação são usadas em suas regras de tratamento, as ofertas existem na implementação. Entretanto, qualquer oferta que não for usada em uma regra de tratamento deverá ser definida na tabela `UACI_ICBatchOffers`.

Sobre Esta Tarefa

A tabela `UACI_ICBatchOffers` existe nas tabelas de sistema do Campaign.

Procedimento

Preencha a tabela UACI_ICBatchOffers com códigos de oferta que você usa na oferta padrão ou tabelas de substituição de pontuação. A tabela possui o formato a seguir:

Nome da coluna	Tipo	Descrição
ICName	varchar(64)	O nome do canal interativo ao qual a oferta está associada. Se você estiver usando a mesma oferta com dois canais interativos diferentes, você deve fornecer uma linha para cada canal interativo.
OfferCode1	varchar(64)	A primeira parte do código de oferta.
OfferCode2	varchar(64)	A segunda parte do código de oferta.
OfferCode3	varchar(64)	A terceira parte do código de oferta.
OfferCode4	varchar(64)	A quarta parte do código de oferta.
OfferCode5	varchar(64)	A quinta parte do código de oferta

Sobre a tabela de ofertas globais

A tabela de ofertas globais permite definir tratamentos no nível de público. Por exemplo, é possível definir uma oferta global para cada membro do público Família.

É possível definir configurações globais para os elementos a seguir da entrega de oferta do Interact.

- Designação de oferta global
- Pontuação do comerciante global, por um número ou uma expressão
- Expressão booleana para filtrar ofertas
- Probabilidade e peso de aprendizado, se você estiver usando o Aprendizado integrado do Interact
- Substituição de global learning

Designando ofertas globais

Use este procedimento para configurar o ambiente de tempo de execução para designar ofertas globais para um nível de público, além de tudo que está definido em regras de tratamento.

Procedimento

1. Crie uma tabela que é chamada UACI_DefaultOffers em seu banco de dados do perfil.

   Para criar a tabela UACI_DefaultOffers com as colunas corretas, use o arquivo ddl aci_usrtab.
2. Configure a propriedade enableDefaultOfferLookup para **true**.

Tabela de oferta global

A tabela de oferta global deve existir em seu banco de dados do perfil. É possível criar a tabela de oferta global, UACI_DefaultOffers executando o script SQL aci_usertab em seu banco de dados do perfil.

O script SQL aci_usertab está localizado no diretório ddl em seu diretório de instalação do ambiente de tempo de execução.

Você deve definir os campos AudienceLevel e OfferCode1 para cada linha. Os outros campos são opcionais para restringir suas designações de oferta adicionais ou influenciar o aprendizado integrado no nível de público.

Para um melhor desempenho, você deve criar um índice nesta tabela na coluna do nível de público.

Nome	Tipo	Descrição
AudienceLevel	varchar(64)	(Necessário) O nome do nível de público a que você designa a oferta padrão, por exemplo, `customer` ou `household`. Esse nome deve corresponder ao nível de público conforme definido no Campaign.
OfferCode1	varchar(64)	(Necessário) O código de oferta para a oferta padrão. Se seus códigos de oferta forem feitos de diversos campos, é possível incluir as colunas adicionais, por exemplo, `OfferCode2` e assim por diante. Se estiver incluindo esta oferta para fornecer uma designação de oferta global, você deve incluir essa oferta na tabela `UACI_ICBatchOffers`.
Pontuação	valor flutuante	Um número para definir a pontuação de marketing para esta designação de oferta.
OverrideTypeID	int	Se configurado como 1, se a oferta não existir na lista de candidatos de ofertas, inclua essa oferta na lista, bem como usando quaisquer dados de pontuação para a oferta. Em geral, use 1 para fornecer designações de oferta global. Se configurado para `0`, *null*, ou qualquer outro número que não 1, use quaisquer dados para a oferta somente se a oferta existir na lista de candidatos de ofertas. Na maioria dos casos, uma regra de tratamento ou designação individual substituirá esta configuração.
Predicado	varchar(4000)	É possível inserir expressões nesta coluna como para opções avançadas para regras de tratamento. É possível usar as mesmas variáveis e macros disponíveis a você ao gravar opções avançadas para regras de tratamento. O comportamento dessa coluna depende do valor na coluna `EnableStateID`. • Se `EnableStateID` for 2, esta coluna funcionará igual à opção **Considere esta regra elegível se a expressão a seguir for true** nas opções avançadas para regras de tratamento para restringir essa designação de oferta. Essa coluna deve conter uma expressão booleana e resolver para true para incluir essa oferta. Se você acidentalmente definir uma expressão que resolva para um número, qualquer número diferente de zero será considerado true e zero será considerado false. • Se `EnableStateID` for 3, esta coluna funcionará igual à opção **Use a expressão a seguir como a pontuação de marketing** nas opções avançadas para regras de tratamento para restringir essa oferta. Essa coluna deve conter uma expressão que resolva para um número. • Se `EnableStateID` for 1, Interact ignorará qualquer valor nessa coluna.
FinalScore	valor flutuante	Um número para substituir a pontuação final usada para ordenar a lista final de ofertas retornadas. Esta coluna é usada se você tiver ativado o módulo de aprendizado integrado. É possível implementar seu próprio aprendizado para usar essa coluna.
CellCode	varchar(64)	O código de célula para um segmento interativo ao qual deseja designar esta oferta padrão. Se seus códigos de célula forem feitos de diversos campos, é possível incluir as colunas adicionais. Você deve fornecer um código de célula se `OverrideTypeID` for 0 ou nulo. Se você não incluir um código de célula, o ambiente de tempo de execução ignora esta linha de dados. Se `OverrideTypeID` for 1, você não precisará fornecer um código de célula nesta coluna. Se você não fornecer um código de célula, o ambiente de tempo de execução usará o código de célula definido na propriedade `DefaultCellCode` para este nível de público e tabela para fins de relatório.
Zona	varchar(64)	O nome da zona à qual você deseja que esta designação de oferta se aplique. Se NULL, isto se aplica a todas as zonas.
EnableStateID	int	O valor nesta coluna define o comportamento da coluna `Predicate`. • **1** - não use a coluna `Predicate`. • **2** - use `Predicate` como um booleano para filtrar a oferta. Isto segue as mesmas regras que a opção avançada **Considere esta regra elegível se a expressão a seguir for true** em uma regra de tratamento. • **3** - use `Predicate` para definir a pontuação do comerciante. Isto segue as mesmas regras que a opção avançada **Use a expressão a seguir como a pontuação de marketing** em uma regra de tratamento. Qualquer linha em que esta coluna seja Null ou qualquer valor que não 2 ou 3 ignora a coluna `Predicate`.
LikelihoodScore	valor flutuante	Esta coluna é usada somente para influenciar o aprendizado integrado. É possível incluir essa coluna com a ddl `aci_scoringfeature`.
AdjExploreScore	valor flutuante	Esta coluna é usada somente para influenciar o aprendizado integrado. É possível incluir essa coluna com a ddl `aci_scoringfeature`.

Sobre a tabela de substituição de pontuação

A tabela de substituição de pontuação permite definir tratamentos em um ID de público ou nível individual. Por exemplo, se seu nível de público for Visitante, é possível criar substituições para visitantes específicos.

É possível definir substituições para os elementos a seguir da entrega de oferta do Interact.

- Designação de oferta individual
- Pontuação de comerciante individual, por um número ou por uma expressão
- Expressão booleana para filtrar ofertas
- Probabilidade de aprendizado e peso, se você estiver usando o aprendizado integrado
- Substituição de aprendizado individual

Configurando substituições de pontuação

É possível configurar o Interact para usar uma pontuação que é gerada a partir de um aplicativo de modelagem em vez da pontuação de marketing.

Procedimento

1. Crie uma tabela de substituição de pontuação para cada nível de público para o qual deseja fornecer substituições.

   Para criar uma tabela de substituição de pontuação de amostra com as colunas corretas, use o arquivo ddl `aci_usrtab`.
2. Configure a propriedade `enableScoreOverrideLookup` para **true**.
3. Configure a propriedade `scoreOverrideTable` para o nome da tabela de substituição de pontuação para cada nível de público para o qual deseja fornecer substituições.

   Você não precisa fornecer uma tabela de substituição de pontuação para cada nível de público.

Tabela de substituição de pontuação

A tabela de substituição de pontuação deve existir em seu banco de dados do perfil de produção. É possível criar uma tabela de substituição de pontuação, `UACI_ScoreOverride` executando o script SQL `aci_usertab` em seu banco de dados do perfil.

O script SQL `aci_usertab` está localizado no diretório `ddl` em seu diretório de instalação do ambiente de tempo de execução.

Você deve definir os campos *`AudienceID`*, `OfferCode1` e `Score` para cada linha. Os valores nos outros campos são opcionais para restringir suas designações de oferta individual adicionais ou fornecer informações de substituição de pontuação para o aprendizado integrado.

Nome	Tipo	Descrição
AudienceID	varchar(64)	(Necessário) O nome desta coluna deve corresponder ao nome da coluna que define o ID de público no Campaign. A tabela de amostra criada pelo arquivo ddl `aci_usertab` cria esta coluna como a coluna `CustomerID`. Se seu ID de público consistir em diversas colunas, é possível incluí-los nesta tabela. Cada linha deve conter o ID de público ao qual você designa a oferta individual, por exemplo, `customer1`. Para um melhor desempenho, você deve criar um índice nesta coluna.

Nome	Tipo	Descrição
OfferCode1	varchar(64)	(Necessário) O código de oferta para a oferta. Se seus códigos de oferta forem feitos de diversos campos, é possível incluir as colunas adicionais, por exemplo, `OfferCode2` e assim por diante. Se você estiver incluindo esta oferta para fornecer uma designação de oferta individual, você deve incluir essa oferta na tabela `UACI_ICBatchOffers`.
Pontuação	valor flutuante	Um número para definir a pontuação de marketing para esta designação de oferta.
OverrideTypeID	int	Se configurado como `0` ou *null* (ou qualquer número que não 1), use quaisquer dados para a oferta apenas se a oferta existir na lista de ofertas candidatas. Em geral, use `0` para fornecer substituições de pontuação. Você deve fornecer um código de célula Se configurado como `1`, se a oferta não existir na lista de candidatos de ofertas, inclua essa oferta na lista, bem como usando quaisquer dados de pontuação para a oferta. Em geral, use `1` para fornecer designações de oferta individual.
Predicado	varchar(4000)	É possível inserir expressões nesta coluna como para opções avançadas para regras de tratamento. É possível usar as mesmas variáveis e macros disponíveis a você ao gravar opções avançadas para regras de tratamento. O comportamento dessa coluna depende do valor na coluna `EnableStateID`. • Se `EnableStateID` for `2`, esta coluna funcionará igual à opção **Considere esta regra elegível se a expressão a seguir for true** nas opções avançadas para regras de tratamento para restringir essa designação de oferta. Essa coluna deve conter uma expressão booleana e resolver para true para incluir essa oferta. Se você acidentalmente definir uma expressão que resolva para um número, qualquer número diferente de zero será considerado true e zero será considerado false. • Se `EnableStateID` for `3`, esta coluna funcionará igual à opção **Use a expressão a seguir como a pontuação de marketing** nas opções avançadas para regras de tratamento para restringir essa oferta. Essa coluna deve conter uma expressão que resolva para um número. • Se `EnableStateID` for `1`, Interact ignorará qualquer valor nessa coluna.
FinalScore	valor flutuante	Um número para substituir a pontuação final usada para ordenar a lista final de ofertas retornadas. Esta coluna é usada se você tiver ativado o módulo de aprendizado integrado. É possível implementar seu próprio aprendizado para usar essa coluna.
CellCode	varchar(64)	O código de célula para um segmento interativo ao qual deseja designar esta oferta. Se seus códigos de célula forem feitos de diversos campos, é possível incluir as colunas adicionais. Você deve fornecer um código de célula se `OverrideTypeID` for `0` ou nulo. Se você não incluir um código de célula, o ambiente de tempo de execução ignora esta linha de dados. Se `OverrideTypeID` for `1`, você não precisará fornecer um código de célula nesta coluna. Se você não fornecer um código de célula, o ambiente de tempo de execução usará o código de célula definido na propriedade `DefaultCellCode` para este nível de público e tabela para fins de relatório.
Zona	varchar(64)	O nome da zona à qual você deseja que esta designação de oferta se aplique. Se NULL, isto se aplica a todas as zonas.
EnableStateID	int	O valor nesta coluna define o comportamento da coluna `Predicate`. • **1** - não use a coluna `Predicate`. • **2** - use `Predicate` como um booleano para filtrar a oferta. Isto segue as mesmas regras que a opção avançada **Considere esta regra elegível se a expressão a seguir for true** em uma regra de tratamento. • **3** - use `Predicate` para definir a pontuação do comerciante. Isto segue as mesmas regras que a opção avançada **Use a expressão a seguir como a pontuação de marketing** em uma regra de tratamento. Qualquer linha em que esta coluna seja Null ou qualquer valor que não 2 ou 3 ignora a coluna `Predicate`.
LikelihoodScore	valor flutuante	Esta coluna é usada somente para influenciar o aprendizado integrado. É possível incluir essa coluna com a ddl `aci_scoringfeature`.
AdjExploreScore	valor flutuante	Esta coluna é usada somente para influenciar o aprendizado integrado. É possível incluir essa coluna com a ddl `aci_scoringfeature`.

Visão geral de aprendizado integrado do Interact

Enquanto você faz tudo que pode para assegurar que propõe as ofertas corretas aos segmentos corretos, é sempre possível aprender algo das seleções reais de seus visitantes. O comportamento real de seus visitantes deve influenciar sua estratégia. É possível executar o histórico de respostas através de algumas ferramentas de modelagem para obter uma pontuação que pode ser incluída em seus fluxogramas interativos.

Entretanto, estes dados não são de tempo real.

O Interact fornece duas opções para que você aprenda com as ações do visitante em tempo real:

- Módulo de aprendizado integrado - o ambiente de tempo de execução possui um módulo de aprendizado baseado em Naive Bayesian. Esse módulo monitora os atributos do cliente de sua escolha e usa esses dados para ajudar a selecionar quais ofertas apresentar.
- API de aprendizado - o ambiente de tempo de execução também possui uma API de aprendizado para gravar seu próprio módulo de aprendizado.

Você não tem de usar o aprendizado. Por padrão, o aprendizado está desativado.

Módulo de aprendizado do Interact

O módulo de aprendizado do Interact monitora as respostas do visitante a ofertas e atributos de visitante.

Modos de módulos de aprendizado

O módulo de aprendizado possui dois modos gerais:

- Exploração - o módulo de aprendizado serve ofertas em ordem para que possa reunir dados de resposta suficientes para otimizar a estimativa que é usada durante o modo de exploração. Ofertas servidas durante a exploração não refletem necessariamente a opção ideal.
- Explotação - após dados suficientes serem coletados pela fase de exploração, o módulo de aprendizado usará as probabilidades para ajudar a selecionar as ofertas para apresentar.

O módulo de aprendizado usa duas propriedades para alternar entre o modo de exploração e o modo de explotação. As duas propriedades são:

- um nível de confiança que você configura com a propriedade `confidenceLevel`.
- uma probabilidade que o módulo de aprendizado apresenta uma oferta aleatória que você configura com a propriedade `percentRandomSelection`.

Propriedade de nível de confiança

Você configura o `confidenceLevel` para uma porcentagem que apresenta o nível de certeza (ou confiança) que o módulo de aprendizado deve estar antes de pontuações para uma oferta serem usadas na arbitragem. Em primeiro lugar, quando o módulo de aprendizado não possuir dados a partir dos quais trabalhar, o módulo de aprendizado dependerá inteiramente da pontuação de marketing. Após cada oferta ser apresentada quantas vezes definido pelo `minPresentCountThreshold`, o módulo de aprendizado entrará no modo de

exploração. Sem muitos dados para trabalhar, o módulo de aprendizado não terá confiança de que as porcentagens que calcula estão corretas. Portanto, ele permanece no modo de exploração.

O módulo de aprendizado designa pesos a cada oferta. Para calcular os pesos, o módulo de aprendizado usa uma fórmula que toma como entrada o nível de confiança, os dados de aceitação históricos e os dados da sessão atuais. A fórmula equilibra inerentemente entre exploração e explotação e retorna o peso apropriado.

Propriedade de seleção aleatória

Para assegurar que o sistema não seja inclinado às ofertas com melhor desempenho durante estágios iniciais, o Interact apresenta uma oferta aleatória ao percentual de `percentRandomSelection` do tempo. Esta porcentagem de oferta aleatória força o módulo de aprendizado para recomendar ofertas diferentes das mais bem-sucedidas para determinar se outras ofertas seriam mais bem-sucedidas se tivessem maior exposição. Por exemplo, se você configurar `percentRandomSelection` para 5, 5% do tempo que o módulo de aprendizado apresenta uma oferta aleatória e inclui os dados de resposta a seus cálculos.

É possível configurar a **% Aleatória** para especificar a mudança na qual a oferta retornada é selecionada aleatoriamente, sem considerar pontuações, para cada zona na guia Pontos de interação da janela Canal Interativo.

Como o módulo de aprendizado determina as ofertas

O módulo de aprendizado determina quais ofertas são apresentadas da forma a seguir.

1. Calcula a probabilidade de um visitante selecionar uma oferta.
2. Calcula o peso da oferta usando a probabilidade da etapa 1 e determina se deve estar no modo de exploração e explotação.
3. Calcula uma pontuação final para cada oferta usando a pontuação de marketing e o peso da oferta da etapa 2.
4. Classifica as ofertas pelas pontuações que são determinadas na etapa 3 e retorna o número solicitado das principais ofertas.

Por exemplo, o módulo de aprendizado determina que um visitante tem 30% de probabilidade de aceitar a oferta A e 70% de probabilidade de aceitar a oferta B e explorar essas informações. Nas regras de tratamento, a pontuação de marketing para a oferta A é 75 e 55 para oferta B. Entretanto, os cálculos na etapa 3 tornam a pontuação final para a oferta B mais alta que a oferta A, portanto, o ambiente de tempo de execução recomenda a oferta B.

Propriedades de fator de peso

O aprendizado também é baseado na propriedade `recencyWeightingFactor` e na propriedade `recencyWeightingPeriod`. Estas propriedades permitem que você inclua mais peso em dados mais recentes que os dados mais antigos. `recencyWeightingFactor` é a porcentagem de peso a dar aos dados recentes. `recencyWeightingPeriod` é a duração de tempo recente. Por exemplo, você configura `recencyWeightingFactor` para 0,30 e `recencyWeightingPeriod` para 24. Essas configurações significam que as 24 horas anteriores de dados são 30% de todos os dados considerados. Para uma semana de dados, todos os dados médios nos seis primeiros dias são 70% dos dados e o último dia é 30% dos dados.

Dados de tabela de migração gravados

Cada sessão grava os dados a seguir em uma tabela de migração de aprendizado:

- Contato de oferta
- Aceitação da Oferta
- Atributos de aprendizado

Em um intervalo configurável, um agregador lê os dados de uma tabela de migração, compila-os e os grava em uma tabela. O módulo de aprendizado lê esses dados agregados e os usa em cálculos.

Ativando o módulo de aprendizado

Todos os servidores de runtime têm um módulo de aprendizado integrado. Por padrão, esse módulo de aprendizado é desativado. Você ativa o módulo de aprendizado alterando uma propriedade de configuração.

Procedimento

No Marketing Platform para o ambiente de tempo de execução, edite as propriedades de configuração a seguir na categoria `Interact > offerserving`.

Propriedade de configuração	Configuração
`optimizationType`	**BuiltInLearning**

Atributos de aprendizado

O módulo de aprendizado aprende usando os atributos de visitante e dados de aceitação da oferta. É possível selecionar quais atributos de visitante você monitora. Esses atributos de visitante podem ser qualquer coisa em um perfil do cliente, incluindo algum parâmetro de evento que coletar em tempo real.

Atributos de tabelas dimensionais não são suportados no aprendizado.

Embora você possa configurar qualquer número de atributos para monitorar, o IBM recomenda configurar não mais do que dez atributos de aprendizado entre os atributos de aprendizado estático e dinâmico, bem como seguir estas diretrizes.

- Selecione atributos independentes.

  Não selecione atributos que são semelhantes. Por exemplo, se criar um atributo chamado HighValue e esse atributo for definido por um cálculo baseado em salário, não selecione HighValue e Salary. Atributos semelhantes não ajudam o algoritmo de aprendizado.
- Selecione atributos com valores discretos.

  Se um atributo tiver valores de intervalo, você deve selecionar um valor exato. Por exemplo, se você desejar usar salário como um atributo, você deve dar a cada intervalo de salário um valor específico, o intervalo 20.000-30.000 deve ser A, 30.001-40.000 deve ser B e assim por diante.
- Limite o número de atributos que você controla, para que não impeça o desempenho.

  O número de atributos que pode ser controlado depende de seus requisitos de desempenho e sua instalação do Interact. Se você puder, use outra ferramenta de modelagem (como o PredictiveInsight) para determinar os dez principais atributos de previsão. É possível configurar o módulo de aprendizado para remover automaticamente os atributos que não são de previsão, mas que também possuem um custo de desempenho.

É possível gerenciar o desempenho definindo tanto o número de atributos que você monitora e o número de valores por atributo que você monitora. A propriedade `maxAttributeNames` define o número máximo de atributos de visitante que você controla. A propriedade `maxAttributeValues` define o número máximo de valores que você controla por atributo. Todos os outros valores serão designados a uma categoria definida pelo valor da propriedade `otherAttributeValue`. Entretanto, o mecanismo de aprendizado só controla os primeiros valores que encontra. Por exemplo, você está controlando a cor dos olhos do atributo de visitante. Você só está interessado nos valores azul, marrom e verde, portanto, configure `maxAttributeValues` para 3. Entretanto, os três primeiros visitantes têm os valores azul, marrom e avelã. Isso significa que todos os visitantes com olhos verdes são designados a `otherAttributeValue`.

Também é possível usar atributos de aprendizado dinâmico que permitem definir seus critérios de aprendizado mais especificamente. Os atributos de aprendizado dinâmico permitem aprender com a combinação de dois atributos como uma única entrada. Por exemplo, considere as informações de perfil a seguir.

ID do visitante	Tipo de cartão	Saldo de cartão
1	Cartão de ouro	$1.000
2	Cartão de ouro	$9.000
3	Cartão de bronze	$1.000
4	Cartão de bronze	$9.000

Se você usar atributos de aprendizado padrão, será possível apenas aprender com o tipo de cartão e saldo individualmente. Os visitantes 1 e 2 serão agrupados juntos mesmo com base no Tipo de cartão e os visitantes 2 e 4 agrupados com base no Saldo do cartão. Isto pode não ser um preditor preciso do comportamento de aceitação da oferta. Se os portadores de Cartão Ouro tenderem a ter saldos maiores, o comportamento do Visitante 2 poderá ser radicalmente diferente do Visitante 4, que defasaria os atributos de aprendizado padrão. Entretanto, se você usar atributos de aprendizado dinâmico, cada um destes visitantes aprenderá individualmente e as previsões serão mais precisas.

Se você usar atributos de aprendizado dinâmico e o visitante tiver dois valores válidos para um atributo, o módulo de aprendizado selecionará o primeiro valor que encontrar.

Se você configurar a propriedade `enablePruning` para `yes`, o módulo de aprendizado determinará de forma algorítmica quais atributos não serão preditivos e não considerará mais esses atributos ao calcular pesos. Por exemplo, se você estiver rastreando um atributo que representa cor de cabelo e o módulo de aprendizado determinar que não há padrão para aceitar uma oferta baseada na cor de cabelo do visitante, o módulo de aprendizado não mais considerará o atributo cor de cabelo. Os atributos são reavaliados toda vez que o processo de agregação de aprendizado é executado (definido pela propriedade `aggregateStatsIntervalInMinutes`). Os atributos de aprendizado dinâmico também são removidos.

Definindo um atributo de aprendizado

Use este procedimento para definir um atributo de aprendizado.

Sobre Esta Tarefa

É possível configurar até o número `maxAttributeNames` de atributos do visitante.

O *(learningAttributes)* é um modelo para criar novos atributos de aprendizado padrão. Você deve inserir um novo nome para cada atributo. Não é possível criar duas categorias com o mesmo nome

Procedimento

No Marketing Platform para o ambiente de design, edite as propriedades de configuração a seguir na categoria `Campaign > partitions > partition`*n* `> Interact > learning`.

Propriedade de configuração	Configuração
`attributeName`	O `attributeName` deve corresponder ao nome de um par nome-valor nos dados do perfil. Este nome não faz distinção entre maiúsculas e minúsculas.

Defina atributos de aprendizado dinâmico

Para definir atributos de aprendizado dinâmico, você deve preencher a tabela `UACI_AttributeList` na origem de dados Aprendizado.

Todas as colunas nesta tabela têm o tipo de varchar(64).

Coluna	Descrição
`AttributeName`	O nome do atributo dinâmico mediante o qual você deseja aprender. Este valor deve ser um valor real possível no `AttributeNameCol`.
`AttributeNameCol`	O nome da coluna completo (estrutura hierárquica, a partir da tabela de perfis) em que `AttributeName` pode ser localizado. Este nome da coluna não precisa ser um atributo de aprendizado padrão.
`AttributeValueCol`	O nome da coluna completo (estrutura hierárquica, a partir da tabela de perfis) em que o valor associado para o `AttributeName` pode ser localizado.

Por exemplo, considere a tabela de perfis a seguir e sua tabela de dimensões associada.

Tabela 6. MyProfileTable

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

Tabela 7. MyDimensionTable

KeyField	CardType	CardBalance
Key1	Cartão de ouro	1000
Key2	Cartão de ouro	9000
Key3	Cartão de bronze	1000
Key4	Cartão de bronze	9000

A seguir, uma tabela de amostra `UACI_AttributeList` que corresponde no tipo de cartão e saldo.

Tabela 8. UACI_AttributeList

AttributeName	AttributeNameCol	AttributeValueCol
Cartão de ouro	MyProfileTable.MyDimensionTable. CardType	MyProfileTable.MyDimensionTable. CardBalance
Cartão de bronze	MyProfileTable.MyDimensionTable. CardType	MyProfileTable.MyDimensionTable. CardBalance

Configurando o ambiente de tempo de execução para reconhecer módulos de aprendizado externo

É possível usar a API de aprendizado Java™ para gravar seu próprio módulo de aprendizado. Você deve configurar o ambiente de tempo de execução para reconhecer seu utilitário de aprendizado no Marketing Platform.

Sobre Esta Tarefa

Você deve reiniciar o servidor de runtime do Interact para que estas mudanças entrem em vigor.

Procedimento

1. No Marketing Platform para o ambiente de tempo de execução, edite as propriedades de configuração a seguir na categoria `Interact > offerserving`. As propriedades de configuração para a API de otimizador de aprendizado existem na categoria `Interact > offerserving > External Learning Config`.

Propriedade de configuração	Configuração
`optimizationType`	**ExternalLearning**
`externalLearningClass`	nome de classe para o aprendizado externo
`externalLearningClassPath`	O caminho para a classe ou arquivos JAR no servidor de runtime para o aprendizado externo. Se você estiver usando um grupo de servidores e todos os servidores de runtime fizerem referência à mesma instância do Marketing Platform, cada servidor deverá ter uma cópia da classe ou arquivos JAR no mesmo local.

2. Reinicie o servidor de runtime do Interact para que estas mudanças entrem em vigor.

Capítulo 5. Entendendo a API do Interact

O Interact entrega ofertas dinamicamente em uma ampla variedade de pontos de contato. Por exemplo, é possível configurar o ambiente de tempo de execução e o ponto de contato para enviar mensagens aos funcionários da central de atendimento informando-os sobre as melhores perspectivas de venda cruzada ou de venda de produto complementar para um cliente que tenha feito uma chamada com um tipo específico de consulta de serviço. Também é possível configurar o ambiente de tempo de execução e o ponto de contato para fornecer ofertas customizadas a um cliente (visitante) que tenha acessado uma área específica do website.

A interface de programação de aplicativos (API) do Interact permite configurar o ponto de contato e um servidor de runtime para trabalharem em conjunto para entregar as melhores ofertas possíveis. Usando a API, o ponto de contato pode solicitar informações do servidor de runtime para designar o visitante a um grupo (um segmento) e apresentar ofertas com base nesse segmento. Também é possível registrar dados para análise posterior para refinar as estratégias de apresentação de oferta.

A API do Interact também permite a comunicação do cliente do usuário final com o servidor por meio de JavaScript.

Para fornecer a maior flexibilidade possível, integrando o Interact aos seus ambientes, o IBM fornece um serviço da web que pode acessado usando a API do Interact.

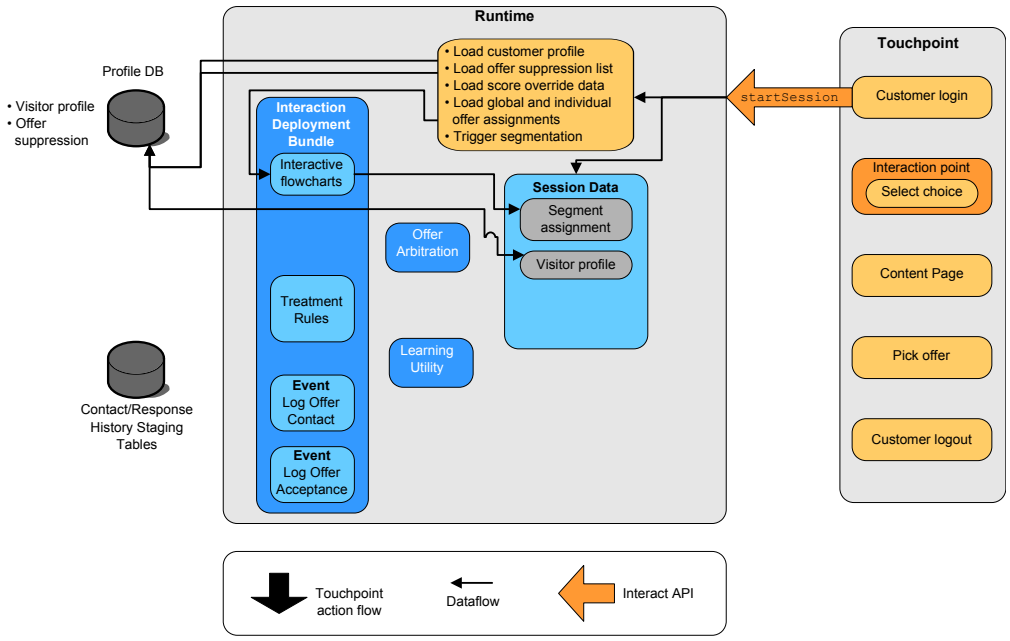
Fluxo de dados da API do Interact

Este exemplo mostra como funciona a API entre o ponto de contato e o ambiente de tempo de execução. O visitante somente usa quatro ações - efetuar login, navegar para a página que exibe ofertas, selecionar uma oferta e efetuar logout. É possível projetar a integração com o grau de complicação necessário, dentro dos limites dos requisitos de desempenho.

Este diagrama mostra uma implementação simples da API do Interact.

Um visitante efetua login em um website e navega até uma página que exibe ofertas. O visitante seleciona uma oferta e efetua logout. Embora a interação seja simples, ocorrerem vários eventos no ponto de contato e no servidor de runtime:

1. Início de uma sessão
2. Navegação para uma página
3. Seleção de uma oferta
4. Fechamento da sessão



Início da sessão

Quando o visitante efetua login, um `startSession` é acionado.

O método `startSession` executa quatro ações:

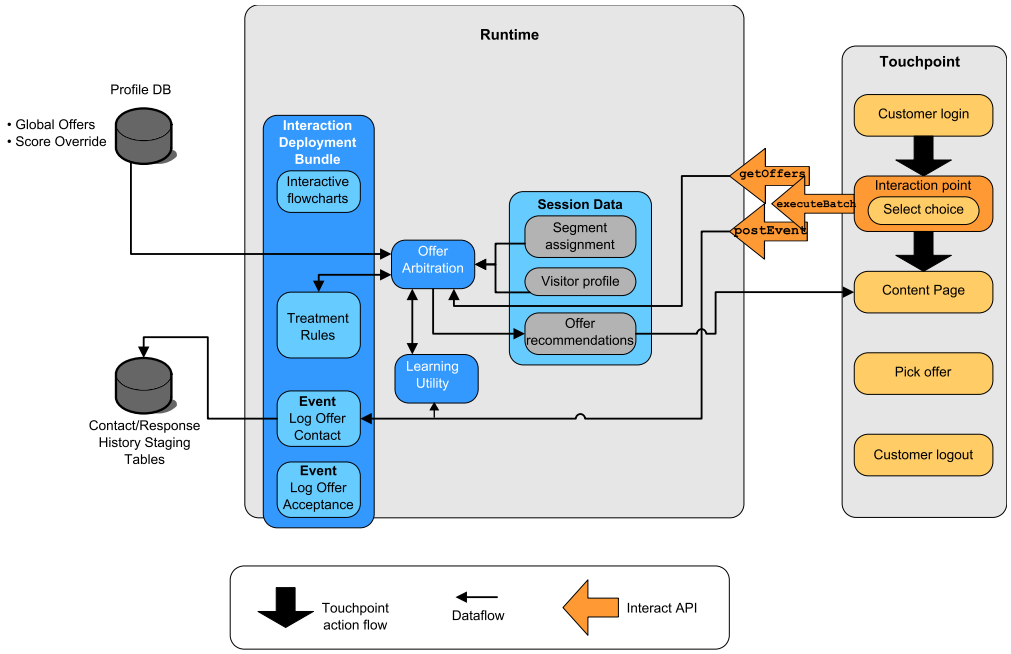
1. Cria uma nova sessão de tempo de execução
2. Envia uma solicitação para carregar os dados do perfil do cliente na sessão
3. Envia uma solicitação para usar os dados do perfil e iniciar um fluxograma interativo para colocar o cliente em segmentos. A execução desse fluxograma é assíncrona.
4. O servidor de runtime carrega quaisquer informações de supressão de oferta e de tratamento de oferta individual e global na sessão. Os dados da sessão são mantidos na memória durante a sessão.

Navegação para uma página

O visitante navega no site até atingir um ponto de interação predefinido. Na figura, o segundo ponto de interação (Selecionar opção) é colocado onde o visitante clica em um link que apresenta um conjunto de ofertas. O gerenciador de ponto de contato configurou o link para acionar um método `executeBatch` para selecionar uma oferta.

Seleção de uma oferta

Este diagrama mostra a chamada de API que aciona o método `executeBatch`.

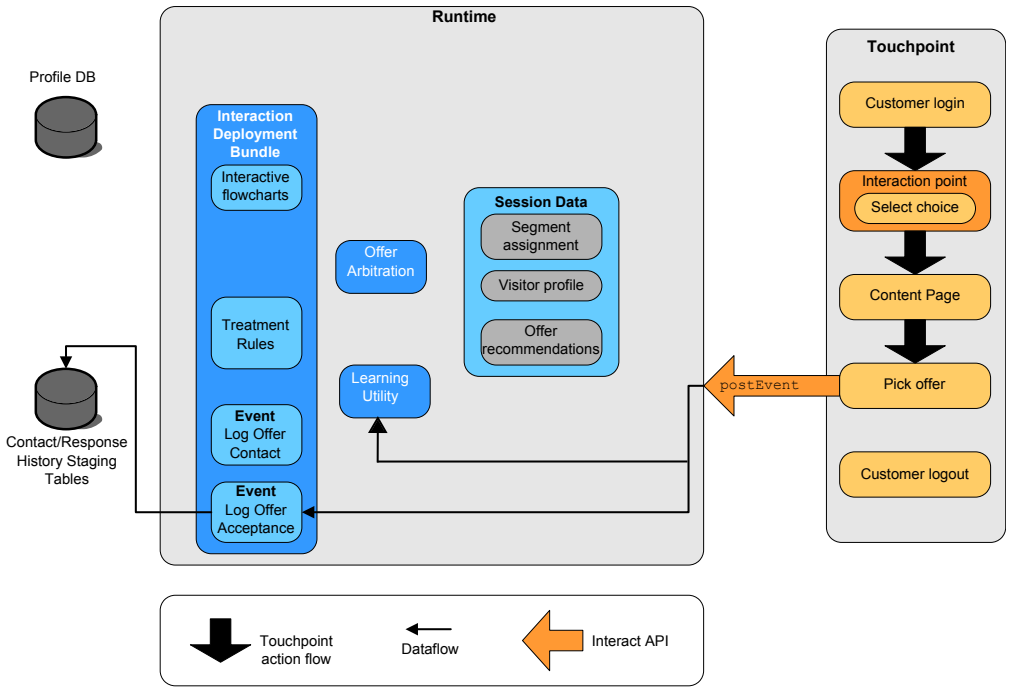


O método `executeBatch` permite chamar mais de um método em uma única chamada para o servidor de runtime. Este executeBatch específico chama dois outros métodos, `getOffers` e `postEvent`. O método `getOffers` solicita uma lista de ofertas. O servidor de runtime usa os dados de segmentação, a lista de supressão de oferta, as regras de tratamento e o módulo de aprendizado para propor um conjunto de ofertas. O servidor de runtime retorna um conjunto de ofertas que são exibidas na página de conteúdo.

O método `postEvent` aciona um dos eventos que são definidos no ambiente de design. Nesse caso específico, o evento envia uma solicitação para registrar as ofertas que são apresentadas ao histórico de contato.

O visitante seleciona uma das ofertas (Escolher oferta).

Este diagrama mostra o método `postEvent`.

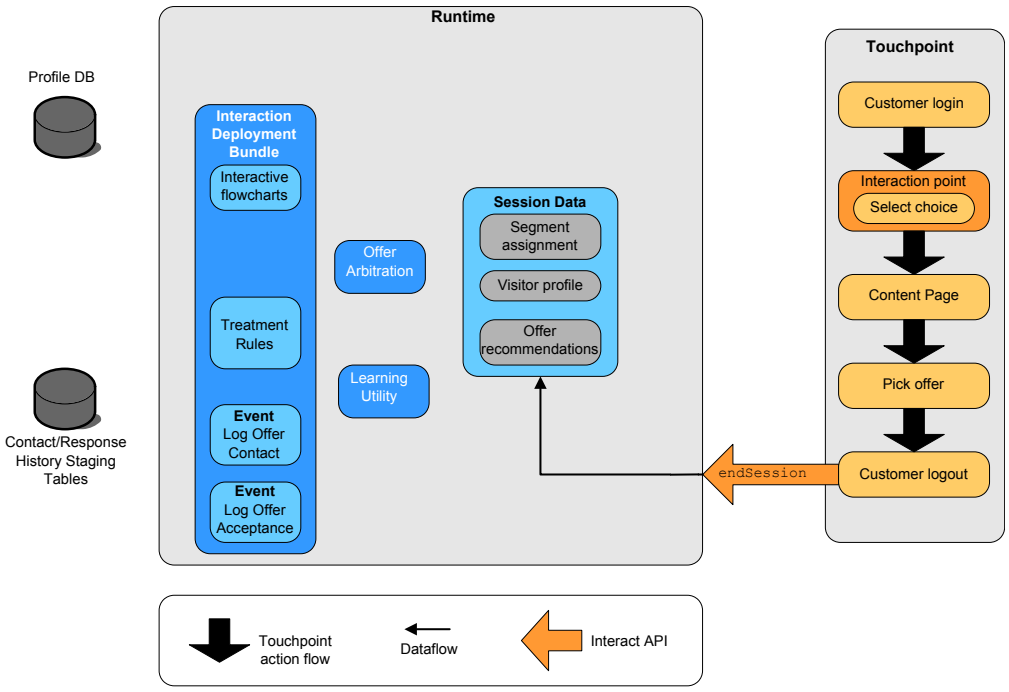


O controle de interface com o usuário associado à seleção da oferta é configurado para enviar um outro método `postEvent`. Esse evento envia uma solicitação para registrar a aceitação da oferta no histórico de respostas.

Fechamento da sessão

Depois que o visitante seleciona a oferta, ele conclui suas ações no website e efetua logout. O comando para efetuar logout é vinculado ao método `endSession`.

Este diagrama mostra o método `endSession`.



O método `endSession` fecha a sessão. Se o visitante esquecer de efetuar logout, haverá um tempo limite da sessão configurável para assegurar que todas as sessões realmente terminem. Se você desejar manter qualquer um dos dados transmitidos para a sessão, como informações incluídas nos parâmetros nos métodos `startSession` ou `setAudience`, trabalhe com a pessoa que cria fluxogramas interativos. A pessoa que cria um fluxograma interativo pode usar o processo de Captura instantânea para gravar esses dados em um banco de dados antes que a sessão termine e que os dados sejam perdidos. Em seguida, será possível usar o método `postEvent` para chamar o fluxograma interativo contendo o processo de Captura instantânea.

Exemplo de planejamento de interação simples

Neste exemplo, você está projetando uma interação para o website de uma empresa de telefone celular. Você cria três diferentes ofertas, configura a criação de log para as ofertas, designa códigos de tratamento para a oferta e mostra uma série de imagens que vinculam à oferta.

Processo de design

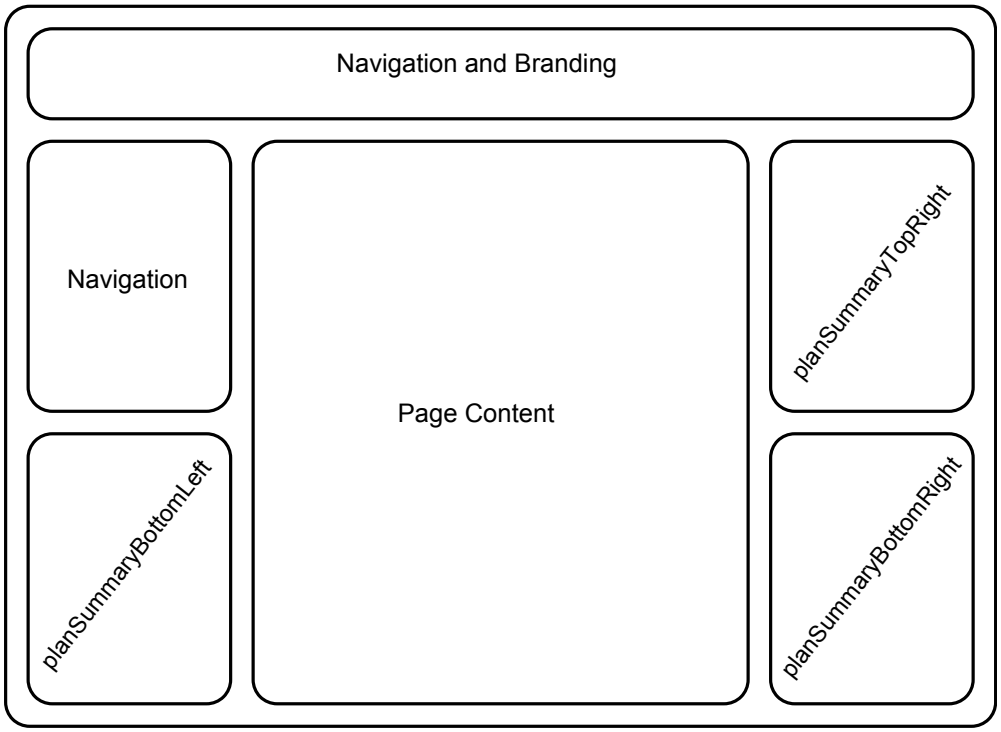
Para projetar uma interação para esse cliente:

1. Identifique os requisitos da página de resumo do cliente
2. Crie pontos de interação para os requisitos de oferta
3. Configure a criação de log para as ofertas
4. Crie códigos de tratamento
5. Vincule uma série de imagens rotativas para a oferta

Esse exemplo é básico e não mostra a melhor maneira de gravar a integração. Por exemplo, nenhum desses exemplos inclui uma verificação de erro que use a classe Resposta.

Identifique os requisitos para a página de resumo do plano de telefone celular

O diagrama a seguir mostra o layout da página de resumo do plano de telefone celular.



Defina os itens a seguir para atender aos requisitos da página de resumo do plano de telefone celular:

Requisito	Implementação
Uma oferta a ser exibida em uma zona dedicada a ofertas sobre upgrades A área na página que exibe o upgrade da oferta deve ser definida. Além disso, depois que o Interact escolhe uma oferta a ser exibida, informações devem ser registrados.	• Ponto de interação: `ip_planSummaryBottomRight` • Evento: `evt_logOffer`
Duas ofertas para upgrades de telefone Cada área na página que exibe os upgrades de telefone deve ser definida.	• Ponto de interação: `ip_planSummaryTopRight` • Ponto de interação:`ip_planSummaryBottomLeft`
Para análise, é necessário registrar quais ofertas são aceitas e quais ofertas são rejeitadas.	• Evento: `evt_offerAccept` • Evento: `evt_offerReject`
Você também sabe que deve transmitir o código de tratamento de uma oferta sempre que registrar um contato, uma aceitação ou rejeição de oferta.	`NameValuePair`
Exiba três imagens rotativas na página. Vincule as imagens às ofertas.	

Crie pontos de interação

Agora é possível solicitar que o usuário do ambiente de design crie os pontos de interação e eventos enquanto você começa a codificar a integração com o ponto de contato.

Para cada ponto de interação que exibir uma oferta, primeiro será necessário obter uma oferta e, em seguida, extrair as informações necessárias para exibir a oferta. Por exemplo, solicite uma oferta para a área inferior direita da página da web (`planSummaryBottomRight`)

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

Essa chamada de resposta retorna um objeto de resposta que inclui uma resposta `OfferList`. No entanto, a página da web não pode usar um objeto `OfferList`. É necessário um arquivo de imagem para a oferta, que você sabe que é um dos atributos de oferta (`offerImg`). É necessário extrair o atributo de oferta necessário do `OfferList`.

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

Configure a criação de log

Agora que a oferta está sendo exibida, isso pode ser registrado como um contato.

```
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)
```

Em vez de chamar cada um desses métodos individualmente, é possível usar o método `executeBatch`, conforme é mostrado no exemplo a seguir para a parte `planSummaryBottomLeft` da página da web.

```
Command getOffersCommand = new CommandImpl();
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(ip_planSummaryBottomLeft);
getOffersCommand.setNumberRequested(1);

Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEvent(evt_logOffer);

/** Build command array */
Command[] commands =
{
 getOffersCommand,
 postEventCommand
};
```

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

Não é necessário definir o UACIOfferTrackingCode neste exemplo. O servidor de runtime do Interact registrará automaticamente a última lista de tratamentos recomendados como contatos se o UACIOfferTrackingCode não for fornecido.

Crie códigos de tratamento

Se necessário, crie um NameValuePair para conter o código de tratamento, como no exemplo a seguir.

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

Vincule imagens a ofertas

Para a segunda área na página que exibe um upgrade de telefone, você gravou algo para alterar a imagem exibida a cada 30 segundos. Você decide girar entre três imagens e usar o seguinte para recuperar o conjunto de ofertas a ser armazenado em cache para ser usado no código para girar as imagens.

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

Você deve gravar a busca de código do cliente a partir do cache local e registrar para o contato uma única vez para cada oferta após a exibição de sua imagem. Para registrar o contato, o parâmetro UACITrackingCode deverá ser postado como antes. Cada oferta possui um código de rastreio diferente.

```
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);
  }
 }
}
```

Para cada oferta, se a oferta for clicada, serão registradas as ofertas aceitas e as ofertas rejeitadas. (Nesse cenário, as ofertas não selecionadas explicitamente serão consideradas rejeitadas.) A seguir está um exemplo que demonstra quando a oferta `ip_planSummaryTopRight` é selecionada:

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

Na prática, seria melhor enviar essas três chamadas `postEvent` com o método `executeBatch`.

Projetando integração da API do Interact

A construção da integração da API do Interact com o ponto de contato requer um pouco de design antes que seja possível iniciar a implementação. É necessário trabalhar com a equipe de marketing para decidir onde no ponto de contato você deseja que o ambiente de tempo de execução entregue ofertas (definir os pontos de interação) e que outro tipo de rastreamento ou funcionalidade interativa deseja usar (definir os eventos).

Na fase de design, isso pode ser uma simples descrição. Por exemplo, para um website de telecomunicações, a página de resumo de plano do cliente deve exibir uma oferta relacionada a upgrade de plano e duas ofertas para upgrades de telefone.

Depois que a empresa decidir onde e como deseja interagir com clientes, será necessário usar o Interact para definir os detalhes. Um autor de fluxograma deverá projetar os fluxogramas interativos que serão usados quando os eventos de ressegmentação ocorrerem. É necessário decidir o número e os nomes de pontos de interação e eventos, bem como quais dados devem ser transmitidos para segmentação, postagem de evento e recuperação de oferta adequadas. O usuário do ambiente de design define os pontos de interação e eventos do Canal interativo. Em seguida, esses nomes poderão ser usados durante a codificação da integração com o ponto de contato no ambiente de tempo de execução. Você também deverá definir quais informações de métrica serão necessárias, para definir quando será necessário registrar contatos e respostas de oferta.

Pontos a serem considerados

Ao projetar uma interação, tenha em mente os efeitos que a ausência de oferta elegível, um servidor de runtime inatingível e a sincronização de processo causam na interação. Seja específico ao definir rejeições de oferta. Considere os recursos do produto opcionais que podem aprimorar a interação.

Ao projetar a interação:

Crie algum conteúdo de preenchimento padrão
Crie um conteúdo de preenchimento padrão, uma mensagem positiva da marca ou um conteúdo vazio, para cada ponto de interação em que as ofertas poderão ser apresentadas. Esse conteúdo de preenchimento é usado quando não há ofertas elegíveis a serem entregues ao visitante atual na situação atual. É possível designar esse conteúdo de preenchimento padrão como a sequência padrão para o ponto de interação.

Inclua um método alternativo de apresentação de conteúdo
Inclua algum método de apresentação de conteúdo para o caso de o ponto de contato não poder atingir o grupo de servidores de runtime por algum motivo inesperado.

Considere o tempo que os fluxogramas em execução gastos
Ao acionar eventos que ressegmentam o visitante, incluindo `postEvent` e `setAudience`, tenha em mente que a execução de fluxogramas demorará um certo período de tempo. O método `getOffers` aguarda até que a segmentação seja concluída para que o método `getOffers` seja executado. Uma frequência excessiva de ressegmentação pode prejudicar o desempenho da resposta da chamada de `getOffers`.

Decida o que uma "rejeição de oferta" significa
Vários relatórios, como o relatório Resumo de desempenho de oferta de canal, apresentam o número de vezes que uma oferta é rejeitada. Este relatório mostra o número de vezes que um `postEvent` acionou uma ação Registrar rejeição de oferta. É necessário determinar se a ação Registrar rejeição de oferta representa uma rejeição real, como um clique em um link chamado **Não, obrigado**. Ou se a ação Registrar rejeição de oferta representa uma oferta ignorada, como uma página que exiba três banners de propaganda diferentes e nenhum dele seja selecionado.

Decida quais recursos de seleção de oferta serão usados
Há diversos recursos opcionais que podem ser usados para aprimorar a seleção de oferta do Interact. Esses recursos incluem:

- Aprendizado
- Supressão de oferta
- Designações de ofertas individuais
- Outros elementos de entrega de oferta

É necessário determinar quantos, caso haja algum, desses recursos opcionais aprimorariam as interações.

Capítulo 6. Gerenciando a API do IBM Interact

Sempre que usar o método `startSession`, você cria uma sessão de tempo de execução do Interact no servidor de runtime. É possível usar as propriedades de configuração para gerenciar as sessões em um servidor de runtime.

Pode ser necessário definir essas configurações conforme você implementa a integração do Interact com o ponto de contato.

Essas propriedades de configuração estão na categoria `sessionManagement`.

Código de idioma e a API do Interact

É possível usar o Interact para pontos de contato que não estejam em inglês. O ponto de contato e todas as sequências na API usam o código do idioma definido para o usuário do ambiente de tempo de execução.

Somente é possível selecionar um código de idioma por grupo de servidores.

Por exemplo, no ambiente de tempo de execução, você cria dois usuários, asm_admin_en com o código do idioma do usuário configurado como inglês e asm_admin_fr com o código do idioma do usuário configurado como francês. Se o ponto de contato for projetado para pessoas que falam francês, defina a propriedade `asmUserForDefaultLocale` para o ambiente de execução como asm_admin_fr.

Sobre o monitoramento JMX

O Interact fornece o serviço de monitoramento Java Management Extensions (JMX) que pode ser acessado com qualquer aplicativo de monitoramento JMX. Esse monitoramento JMX permite monitorar e gerenciar os servidores de runtime.

Os atributos JMX fornecem muitas informações detalhadas sobre o servidor de runtime. Por exemplo, o atributo JMX `ErrorCount` fornece o número de mensagens de erro registradas desde a última reconfiguração ou o último início do sistema. Essas informações podem ser usadas para saber com que frequência há erros no sistema. Se você tiver codificado o website para somente chamar um término de sessão se alguém concluir uma transação, também será possível comparar o `startSessionCount` com o `endSessionCount` para ver quantas transações estão incompletas.

O Interact suporta os protocolos RMI e JMXMP, conforme definido por JSR 160. É possível conectar-se ao serviço de monitoramento JMX com qualquer cliente JMX compatível com JSR160.

Os fluxogramas interativos somente podem ser monitorados com o monitoramento JMX. As informações sobre os Fluxogramas interativos não aparecem no Monitoramento do Campaign.

Nota: Se estiver usando o IBM WebSphere com um gerenciador de nó, você deverá definir o argumento genérico da JVM para ativar o monitoramento JMX.

Configurando o Interact para usar o monitoramento JMX com o protocolo RMI

Use este procedimento para configurar o Interact para usar o monitoramento JMX com o protocolo RMI.

Sobre Esta Tarefa

O endereço padrão para monitoramento para o protocolo for RMI é `service:jmx:rmi:///jndi/rmi://RuntimeServer:port/interact`.

Procedimento

No Marketing Platform para o ambiente de tempo de execução, edite as propriedades de configuração a seguir na categoria `Interact > monitoramento` .

Propriedade de configuração	Configuração
`protocolo`	**RMI**
`porta`	O número da porta para o serviço JMX.
`enableSecurity`	**Falso** A implementação do Interact do protocolo RMI não suporta segurança.

Configurando o Interact para usar o monitoramento JMX com o protocolo JMXMP

Use este procedimento para configurar o Interact para usar o monitoramento JMX com o protocolo JMXMP.

Antes de Iniciar

O protocolo JMXMP requer duas bibliotecas extras na seguinte ordem no caminho de classe, `InteractJMX.jar` e `jmxremote_optional.jar`. Esses dois arquivos podem ser localizados no diretório `lib` de instalação do ambiente de tempo de execução.

Sobre Esta Tarefa

Se você ativar a segurança, o nome do usuário e a senha deverão corresponder a um usuário no Marketing Platform para o ambiente de tempo de execução. Não é possível usar uma senha vazia.

O endereço padrão de monitoramento para o protocolo JMXMP é `service:jmx:jmxmp://RuntimeServer:port`.

Procedimento

1. Verifique se as bibliotecas `InteractJMX.jar` e `jmxremote_optional.jar` estão no caminho de classe em ordem. Se elas não estiverem no caminho de classe, inclua-as no caminho de classe.
2. No Marketing Platform para o ambiente de tempo de execução, edite as propriedades de configuração a seguir na categoria `Interact > monitoramento` .

Propriedade de configuração	Configuração
`protocolo`	**JMXMP**
`porta`	o número da porta para o serviço JMX
`enableSecurity`	**False** para desativar a segurança ou **True** para ativar a segurança

Configurando o Interact para usar os scripts jconsole para o monitoramento JMX

Se você não tiver um aplicativo de monitoramento JMX separado, será possível usar o jconsole que é instalado com a JVM. É possível iniciar o jconsole com os scripts de inicialização no diretório `Interact/tools`.

Sobre Esta Tarefa

O script do jconsole usa o protocolo JMXMP para monitoramento, por padrão. As configurações padrão do `jconsole.bat` são:

A conexão 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%
```

A conexão 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
```

Procedimento

1. Abra o `Interact\tools\jconsole.bat` (Windows) ou o `Interact/tools/jconsole.sh` (UNIX) em um editor de texto.
2. Configure `INTERACT_LIB` com o caminho completo para o diretório *`InteractInstallationDirectory`*`/lib`.
3. Configure `HOST` com o nome do host do servidor de runtime que deseja monitorar.
4. Configure `PORT` com a porta que foi configurada para o JMX atender com a propriedade `Interact > monitoramento > porta`.
5. Opcional: Se você estiver usando o protocolo RMI para monitoramento, inclua um comentário antes da conexão JMXMP e remova o comentário antes da conexão RMI.

Atributos JMX

Há vários atributos disponíveis para o monitoramento JMX. Os atributos do ambiente de design incluem o monitoramento do ETL de histórico de respostas de contatos. Os atributos do ambiente de tempo de execução incluem exceções, vários atributos de fluxograma diferentes, código do idioma, criador de logs e estatísticas do conjunto de encadeamentos. Vários atributos de estatísticas de serviço também estão disponíveis. Todos os dados fornecidos pelo monitoramento JMX são fornecidos desde a última reconfiguração ou inicialização do sistema. Por exemplo, as contagens do número de itens ocorrem desde a última reconfiguração ou inicialização do sistema, não desde a instalação.

Atributos do Monitor do ETL de histórico de respostas de contatos

Os atributos do Monitor do ETL de histórico de respostas de contatos fazem parte do ambiente de design. Todos os atributos a seguir fazem parte do ambiente de tempo de execução.

Tabela 9. Monitor do ETL de histórico de respostas de contatos

Atributo	Descrição
AvgCHExecutionTime	O número médio de milissegundos que o módulo de histórico de contatos e respostas gasta para gravar na tabela de históricos de contatos. Essa média somente é calculada para as operações que são bem-sucedidas e para as quais há pelo menos um registro gravado na tabela de históricos de resposta.
AvgETLExecutionTime	O número médio de milissegundos que o módulo de histórico de contatos e respostas gasta para ler dados do ambiente de tempo de execução. A média inclui o tempo para operações bem-sucedidas e com falha.
AvgRHExecutionTime	O número médio de milissegundos que o módulo de histórico de contatos e respostas gasta para gravar na tabela de históricos de respostas. Essa média somente é calculada para as operações que são bem-sucedidas e para as quais há pelo menos um registro gravado na tabela de históricos de resposta.
ErrorCount	O número de mensagens de erro registradas desde a última reconfiguração ou o último início do sistema, caso haja.
HighWaterMarkCHExecutionTime	O número máximo de milissegundos que o módulo de histórico de contatos e respostas gastou para gravar na tabela de históricos de contatos. Esse valor somente é calculado para as operações que são bem-sucedidas e para as quais há pelo menos um registro gravado na tabela de históricos de resposta.
HighWaterMarkETLExecutionTime	O número máximo de milissegundos que o módulo de histórico de contatos e respostas gastou para ler dados do ambiente de tempo de execução. O cálculo inclui as operações bem-sucedidas e com falha.
HighWaterMarkRHExecutionTime	O número máximo de milissegundos que o módulo de histórico de contatos e respostas gastou para gravar na tabela de históricos de contatos. Esse valor somente é calculado para as operações que são bem-sucedidas e para as quais há pelo menos um registro gravado na tabela de históricos de respostas.
LastExecutionDuration	O número de milissegundos que o módulo de histórico de contatos e respostas gastou para executar a última cópia.
NumberOfExecutions	O número de vezes que o módulo de histórico de contatos e respostas foi executado desde a última inicialização.
LastExecutionStart	O horário em que a última execução do módulo de histórico de contatos e respostas foi iniciada.
LastExecutionSuccessful	Se for true, a última execução do módulo de histórico de contatos e respostas bem-sucedida. Se for false, terá ocorrido um erro.
NumberOfContactHistoryRecordsMarked	O número de registros de histórico de contatos na tabela `UACI_CHStaging` que estão sendo movido durante a execução atual do módulo de histórico de contatos e respostas. Esse valor somente será maior que zero se o módulo de histórico de contatos e respostas estiver em execução.
NumberOfResponseHistoryRecordsMarked	O número de registros de histórico de contatos na tabela `UACI_RHStaging` que estão sendo movido durante a execução atual do módulo de histórico de contatos e respostas. Esse valor somente será maior que zero se o módulo de histórico de contatos e respostas estiver em execução.

Atributos de exceção

Os atributos de exceção fazem parte do ambiente de tempo de execução.

Tabela 10. Exceções

Atributo	Descrição
errorCount	O número de mensagens de erro registradas desde a última reconfiguração ou o último início do sistema.
warningCount	O número de mensagens de aviso registradas desde a última reconfiguração ou o último início do sistema.

Atributos de estatísticas do mecanismo de fluxograma

Os atributos de estatísticas do mecanismo de fluxograma fazem parte do ambiente de tempo de execução.

Tabela 11. Estatísticas do mecanismo de fluxograma

Atributo	Descrição
activeProcessBoxThreads	A contagem ativa de encadeamentos do processo de fluxograma (compartilhados entre todas as execuções) que estão atualmente em execução.
activeSchedulerThreads	A contagem ativa de encadeamentos do planejador de fluxograma que estão atualmente em execução.
avgExecutionTimeMillis	Tempo médio de execução de fluxograma em milissegundos.
CurrentJobsInProcessBoxQueue	O número de tarefas que estão aguardando para serem executadas por encadeamentos do processo de fluxograma.
CurrentJobsInSchedulerQueue	O número de tarefas que estão aguardando para serem executadas por encadeamentos do planejador de fluxograma.
maximumProcessBoxThreads	O número máximo de encadeamentos do processo de fluxograma (compartilhados entre todas as execuções) que podem ser executados.
maximumSchedulerThreads	O número máximo de encadeamentos do planejador fluxograma (um encadeamento por execução) que podem ser executados.
numExecutionsCompleted	O número total de execuções de fluxograma que foram concluídas.
numExecutionsStarted	O número total de execuções de fluxograma iniciadas.

Atributos de fluxogramas específicos por canal interativo

Atributos de fluxogramas específicos por canal interativo fazem parte do ambiente de tempo de execução.

Tabela 12. Fluxogramas específicos por canal interativo

Atributo	Descrição
AvgExecutionTimeMillis	Tempo médio de execução em milissegundos para este fluxograma neste canal interativo.
HighWaterMarkForExecutionTime	Tempo máximo de execução em milissegundos para este fluxograma neste canal interativo.
LastCompletedExecutionTimeMillis	Tempo de execução em milissegundos para a última conclusão deste fluxograma neste canal interativo.
NumExecutionsCompleted	Número total de execuções que foram concluídas para este fluxograma neste canal interativo.
NumExecutionsStarted	Número total de execuções iniciadas para este fluxograma neste canal interativo.

Atributos de código de idioma

Os atributos de código de idioma fazem parte do ambiente de tempo de execução.

Tabela 13. Código de idioma

Atributo	Descrição
locale	Configuração de código de idioma para o cliente JMX.

Atributos de configuração do criador de logs

Os atributos de configuração do criador de logs fazem parte do ambiente de tempo de execução.

Tabela 14. Configurações do criador de logs

Atributo	Descrição
categoria	Altere a categoria de log no qual o nível de log pode ser manipulado.

Atributos de estatísticas do conjunto de encadeamentos de serviços

Os atributos de estatísticas do conjunto de encadeamentos de serviços fazem parte do ambiente de tempo de execução.

Tabela 15. Estatísticas do conjunto de encadeamentos de serviços

Atributo	Descrição
activeContactHistThreads	O número aproximado de encadeamentos que estão executando ativamente as tarefas para Histórico de contato e Histórico de respostas.
activeFlushCacheToDBThreads	O número aproximado de encadeamentos que estão executando ativamente as tarefas para limpar estatísticas em cache para o armazenamento de dados.
activeOtherStatsThreads	O número aproximado de encadeamentos que estão executando tarefas ativamente para Estatísticas elegíveis, Atividades de evento e Estatísticas padrão.
CurrentHighWaterMarkInContactHistQueue	O maior número de entradas enfileiradas a serem registradas pelo serviço que coleta os dados do histórico de contatos e respostas.
CurrentHighWaterMark InFlushCachetoDBQueue	O maior número de entradas enfileiradas para serem registradas pelo serviço que grava os dados no cache para as tabelas de banco de dados.
CurrentHighWaterMarkInOtherStatsQueue	O número máximo de entradas enfileiradas a serem registradas pelo serviço que coleta as estatísticas de elegibilidade de oferta, as estatísticas de uso de sequências padrão, as estatísticas de atividade de evento e o log customizado para dados da tabela.
currentMsgsInContactHistQueue	O número de tarefas na fila para o conjunto de encadeamentos usado para Histórico de contatos e o Histórico de respostas.
currentMsgsInFlushCacheToDBQueue	O número de tarefas na fila para o conjunto de encadeamentos usado para limpar as estatísticas em cache para o armazenamento de dados.
currentMsgsInOtherStatsQueue	O número de tarefas na fila para o conjunto de encadeamentos usado para Estatísticas elegíveis, Atividades de evento e Estatísticas padrão.
maximumContactHistThreads	O maior número de encadeamentos que já estiveram simultaneamente no conjunto usado para Histórico de contatos e o Histórico de respostas.
maximumFlushCacheToDBThreads	O maior número de encadeamentos que já estiveram simultaneamente no conjunto usado para limpar estatísticas em cache para o armazenamento de dados.
maximumOtherStatsThreads	O maior número de encadeamentos que já estiveram simultaneamente no conjunto usado para Estatísticas elegíveis, Atividades de evento e Estatísticas padrão.

Atributos de estatísticas de serviço

As Estatísticas de serviço consistem em um conjunto de atributos para cada serviço.

- ContactHistoryMemoryCacheStatistics - O serviço que coleta dados para as tabelas de migração do histórico de contatos.
- CustomLoggerStatistics - O serviço que coleta dados customizados a serem gravados para uma tabela (um evento que usa o parâmetro de evento `UACICustomLoggerTableName`).
- Estatísticas padrão - O serviço que coleta estatísticas relacionadas ao número de vezes que a sequência padrão para o ponto de interação é usada.
- Estatísticas de elegibilidade - O serviço que grava as estatísticas para ofertas elegíveis.
- Estatísticas de atividades de evento - O serviço que coleta as estatísticas de eventos, eventos do sistema, como `getOffer` ou `startSession` e eventos do usuário acionados por `postEvent`.
- Estatísticas de cache de memória do histórico de respostas - O serviço que grava nas tabelas de migração de histórico de respostas.

- Estatísticas de resposta de sessão cruzada - O serviço que coleta dados de rastreamento de resposta de sessão cruzada.

Tabela 16. Estatísticas de serviço

Atributo	Descrição
Count	O número de mensagens processadas.
ExecTimeInsideMutex	O período de tempo gasto no processamento de mensagens para este serviço, excluindo o tempo gasto aguardando outros encadeamentos, em milissegundos. Se houver uma grande diferença entre ExecTimeInsidMutex e ExecTimeMillis, poderá ser necessário alterar o tamanho do conjunto de encadeamentos para o serviço.
ExecTimeMillis	O período de tempo gasto no processamento de mensagens para este serviço, incluindo o tempo gasto aguardando outros encadeamentos, em milissegundos.
ExecTimeOfDBInsertOnly	O período de tempo, em milissegundos, gasto somente no processamento da parte de inserção em lote.
HighWaterMark	O número máximo de mensagens processadas para este serviço.
NumberOfDBInserts	O número total de inserções em lote executadas.
TotalRowsInserted	O número total de linhas inseridas no banco de dados.

Atributos de Estatísticas de serviço - utilitário de carregamento de banco de dados

Os Atributos de Estatísticas de serviço - utilitário de carregamento de banco de dados fazem parte do ambiente de tempo de execução.

Tabela 17. Estatísticas de serviço - utilitário de carregamento de banco de dados

Atributo	Descrição
ExecTimeOfWriteToCache	O período de tempo, em milissegundos, gasto para gravar no arquivo de cache, incluindo gravação em arquivos e obtenção da chave primária do banco de dados, quando necessário.
ExecTimeOfLoaderDBAccessOnly	O período de tempo, em milissegundos, gasto somente na execução da parte do carregador do banco de dados.
ExecTimeOfLoaderThreads	O período de tempo, em milissegundos, gasto pelos encadeamentos do carregador do banco de dados.
ExecTimeOfFlushCacheFiles	O período de tempo, em milissegundos, gasto limpando o cache e recriando novos.
ExecTimeOfRetrievePKDBAccess	O período de tempo, em milissegundos, gasto recuperando o acesso ao banco de dados de chave primária.
NumberOfDBLoaderRuns	O número total de execuções do carregador do banco de dados.
NumberOfLoaderStagingDirCreated	O número total de diretórios temporários criados.
NumberOfLoaderStagingDirRemoved	O número total de diretórios temporários removidos.
NumberOfLoaderStaging DirMovedToAttention	O número total de diretórios temporários renomeados para atenção.
NumberOfLoaderStagingDirMovedToError	O número total de diretórios temporários renomeados para erro.
NumberOfLoaderStagingDirRecovered	O número total de diretórios temporários recuperados, incluindo no tempo de inicialização, e executados novamente por encadeamentos secundários.
NumberOfTimesRetrievePKFromDB	O número total de vezes que a chave primária foi recuperada a partir do banco de dados.
NumberOfLoaderThreadsRuns	O número total de execuções dos encadeamentos do carregador do banco de dados.
NumberOfFlushCacheFiles	O número total de vezes que o cache de arquivos foi limpo.

Atributos de estatísticas de API

Os Atributos de estatísticas de API fazem parte do ambiente de tempo de execução.

Tabela 18. Estatísticas de API

Atributo	Descrição
endSessionCount	O número de chamadas de API `endSession` desde a última reconfiguração ou o último início do sistema.
endSessionDuration	Tempo decorrido para a última chamada de API `endSession`.

Tabela 18. Estatísticas de API (continuação)

Atributo	Descrição
executeBatchCount	O número de chamadas de API `executeBatch` desde a última reconfiguração ou o último início do sistema.
executeBatchDuration	Tempo decorrido para a última chamada de API `executeBatch`.
getOffersCount	O número de chamadas de API `getOffers` desde a última reconfiguração ou o último início do sistema.
getOffersDuration	Tempo decorrido para a última chamada de API `getOffer`.
getProfileCount	O número de chamadas de API `getProfile` desde a última reconfiguração ou o último início do sistema.
getProfileDuration	Tempo decorrido para a última chamada de API `getProfileDuration`.
getVersionCount	O número de chamadas de API `getVersion` desde a última reconfiguração ou o último início do sistema.
getVersionDuration	Tempo decorrido para a última chamada de API `getVersion`.
loadOfferSuppressionDuration	Tempo decorrido para a última chamada de API `loadOfferSuppression`.
LoadOffersBySQLCount	O número de chamadas de API `LoadOffersBySQL` desde a última reconfiguração ou o último início do sistema.
LoadOffersBySQLDuration	Tempo decorrido para a última chamada de API `LoadOffersBySQL`.
loadProfileDuration	Tempo decorrido para a última chamada de API `loadProfile`.
loadScoreOverrideDuration	Tempo decorrido para a última chamada de API `loadScoreOverride`.
postEventCount	O número de chamadas de API `postEvent` desde a última reconfiguração ou o último início do sistema.
postEventDuration	Tempo decorrido para a última chamada de API `postEvent`.
runSegmentationDuration	Tempo decorrido para a última chamada de API `runSegmentation`.
setAudienceCount	O número de chamadas de API `setAudience` desde a última reconfiguração ou o último início do sistema.
setAudienceDuration	Tempo decorrido para a última chamada de API `setAudience`.
setDebugCount	O número de chamadas de API `setDebug` desde a última reconfiguração ou o último início do sistema.
setDebugDuration	Tempo decorrido para a última chamada de API `setDebug`.
startSessionCount	O número de chamadas de API `startSession` desde a última reconfiguração ou o último início do sistema.
startSessionAverage	Tempo médio decorrido da última chamada API `startSession`.
ActiveSessionCount	O número de sessões que estão ativas no momento na instância de tempo de execução do Intercat. **Nota:** O ActiveSessionCount no bean gerenciado do JMX com.unicacorp.interact:type=api, group=Statistics não considera eventos com tempo limite atingido e, portanto, pode mostrar contagem de ativos incorreta.

Atributos de estatísticas de otimizador de aprendizado

Os Atributos de estatísticas de otimizador de aprendizado fazem parte do ambiente de tempo de execução.

Tabela 19. Estatísticas de otimizador de aprendizado

Atributo	Descrição
LearningOptimizerAcceptCalls	O número de eventos de aceitação transmitidos para o módulo de aprendizado.
LearningOptimizer AcceptTrackingDuration	O número total de milissegundos gastos com o registro dos eventos de aceitação no módulo de aprendizado.
LearningOptimizerContactCalls	O número de eventos de contato transmitidos para o módulo de aprendizado.
LearningOptimizer ContactTrackingDuration	O número total de milissegundos gastos com o registro dos eventos de contato no módulo de aprendizado.
LearningOptimizerLogOtherCalls	O número de eventos que não sejam de contato ou aceitação transmitidos para o módulo de aprendizado.
LearningOptimizer LogOtherTrackingDuration	O período, em milissegundos, gasto no registro de outros eventos (que não sejam de contato ou aceitação) no módulo de aprendizado.
LearningOptimizer NonRandomCalls	O número de vezes que a implementação de aprendizado configurada foi aplicada.
LearningOptimizer RandomCalls	O número de vezes que a implementação de aprendizado configurada foi aprovada e a seleção aleatória foi aplicada.
LearningOptimizer RecommendCalls	O número de solicitações de recomendação transmitidas para o módulo de aprendizado.

Tabela 19. Estatísticas de otimizador de aprendizado (continuação)

Atributo	Descrição
LearningOptimizer RecommendDuration	O número total de milissegundos gastos na lógica de recomendação do aprendizado.

Atributos de estatísticas de oferta padrão

Os atributos de estatísticas de oferta padrão fazem parte do ambiente de tempo de execução.

Tabela 20. Estatísticas de oferta padrão

Atributo	Descrição
LoadDefaultOffersDuration	Tempo decorrido no carregamento de ofertas padrão.
DefaultOffersCalls	O número de vezes que as ofertas padrão são carregadas.

Atributos dos Dispatchers de Mensagem Acionada

Atributos dos Dispatchers de Mensagem Acionada fazem parte do ambiente de tempo de execução.

Tabela 21. Dispatchers de Mensagem Acionada

Atributo	Descrição
NumRequested	O número total de ofertas que foram solicitadas para despacho usando o dispatcher.
NumDispatched	O número total de ofertas que este dispatcher despachou com sucesso.
AvgExecutionTime	O tempo médio, em milissegundos, que este dispatcher usa para despachar uma oferta. Apenas as ofertas que foram despachadas para os gateways com sucesso são contadas no cálculo.
CurrentQueueSize	O número de ofertas atualmente aguardando para serem despachadas.
GatewayInvocation	O número de ofertas e o tempo de despacho médio, em milissegundos, despachado para cada gateway por este dispatcher. O formato de seu valor é {`gateway name=[number of offers, average dispatching time]`}.

Atributos dos Gateways de Mensagens Acionadas

Os atributos dos Gateways de Mensagens Acionadas fazem parte do ambiente de tempo de execução.

Tabela 22. Gateways de Mensagens Acionadas

Atributo	Descrição
NumValidationRequested	O número total de ofertas que este gateway solicitou para validação.
NumValidated	O número total de ofertas que este gateway validou com sucesso.
AvgValidationTime	O tempo médio, em milissegundos, que este gateway usa para validar uma oferta. Apenas as ofertas que foram validadas com sucesso são contadas no cálculo.
NumDeliveryRequested	O número total de ofertas que este gateway solicitou para entrega.
NumDelivered	O número total de ofertas que este gateway entregou em sucesso.
AvgDeliveryTime	O tempo médio, em milissegundos, que este gateway usa para entregar uma oferta. Apenas as ofertas que foram entregues com sucesso são contadas no cálculo.

Atributos de Mensagens da Mensagem Acionada

Os atributos de Mensagens da Mensagem Acionada fazem parte do ambiente de tempo de execução.

Tabela 23. Mensagens da Mensagem Acionada

Atributo	Descrição
ProcessSuccessCount	O número total de vezes que esta mensagem acionada foi executada com sucesso.
AvgSuccessProcessTime	O tempo médio, em milissegundos, que esta mensagem acionada gasta para cada execução bem-sucedida.
ProcessErrorCount	O número total de vezes que esta mensagem acionada não foi executada com sucesso.
AvgErrorProcessTime	O tempo médio, em milissegundos, que esta mensagem acionada gasta para cada execução que não foi bem-sucedida.
SelectBranchCount	O número total de vezes que uma seleção de ramificação foi executada durante o processamento de mensagens acionadas.
AvgSelectBranchTime	O tempo médio, em milissegundos, que a seleção de ramificação usa durante o processamento de mensagens acionadas.
SelectOfferCount	O número total de vezes que a seleção de oferta foi executada durante o processamento de mensagens acionadas.
AvgSelectOfferTime	O tempo médio, em milissegundos, que a execução de seleção de oferta usa durante o processamento de mensagens acionadas.
SelectChannelCount	O número total de vezes que a seleção de canal foi executada durante o processamento de mensagens acionadas.
AvgSelectChannelTime	O tempo médio, em milissegundos, que a execução da seleção de canal usa durante o processamento de mensagens acionadas.
FlowchartWaitCount	O número total de vezes que esta mensagem acionada aguardou pela conclusão da segmentação.
AvgFlowchartWaitTime	O tempo médio, em milissegundos, que esta mensagem acionada aguardou pela conclusão da segmentação em cada execução.
WaitFlowchartTimeoutCount	O número total de vezes que esta mensagem acionada atingiu o tempo limite ao aguardar pela conclusão da segmentação.

Operações JMX

Há diversas operações disponíveis para o monitoramento JMX.

A tabela a seguir descreve as operações disponíveis para o monitoramento JMX.

Grupo	Atributo	Descrição
Configurações do criador de logs	activateDebug	Configure o nível de log para o arquivo de log que está definido em `Interact/conf/interact_log4j.properties` para depuração.
Configurações do criador de logs	activateError	Configure o nível de log para o arquivo de log que está definido em `Interact/conf/interact_log4j.properties` para erro.
Configurações do criador de logs	activateFatal	Configure o nível de log para o arquivo de log que está definido em `Interact/conf/interact_log4j.properties` para fatal.
Configurações do criador de logs	activateInfo	Configure o nível de log para o arquivo de log que está definido em `Interact/conf/interact_log4j.properties` para informação.
Configurações do criador de logs	activateTrace	Configure o nível de log para o arquivo de log que está definido em `Interact/conf/interact_log4j.properties` para rastreio.
Configurações do criador de logs	activateWarn	Configure o nível de log para o arquivo de log que está definido em `Interact/conf/interact_log4j.properties` para aviso.
Código de idioma	changeLocale	Altere o código de idioma do cliente JMX. Os códigos de idioma suportados são Interact `de`, `en`, `s` e `fr`.
ContactResponseHistory ETLMonitor	reset	Reconfigurar todos os contadores.
Estatísticas de oferta padrão	updatePollPeriod	Atualiza defaultOfferUpdatePollPeriod. Esse valor, em segundos, informa ao sistema quanto tempo esperar antes de atualizar as ofertas padrão no cache. Se for configurado como `-1`, o sistema lê o número de ofertas padrão somente na inicialização.

Capítulo 7. Classes e métodos para as APIs Java, SOAP e REST do IBM Interact

As seguintes seções listam os requisitos e outros detalhes que você deverá saber antes de trabalhar com a API do Interact.

Nota: Esta seção assume que você esteja familiarizado com o ponto de contato, a linguagem de programação Java e o trabalho com uma API baseada em Java.

A API do Interact possui um adaptador do cliente Java que usa a serialização do Java sobre HTTP. Além disso, o Interact fornece um WSDL para suportar clientes SOAP. O WSDL expõe o mesmo conjunto de funções que o adaptador do cliente Java, portanto as seções a seguir, exceto os exemplos, ainda se aplicam.

Nota: Várias ocorrências de qualquer parâmetro em uma única chamada de API não são suportadas.

Classes da API do Interact

A API do Interact é baseada na classe `InteractAPI`.

Há 6 interfaces de suporte.

- `AdvisoryMessage`
- `BatchResponse`
- `NameValuePair`
- `Oferta`
- `OfferList`
- `Resposta`

Essas interfaces possuem 3 classes concretas de suporte. As duas classes concretas a seguir precisam ser instanciadas e transmitidas como argumentos para os métodos da API do Interact:

- NameValuePairImpl
- CommandImpl

Uma terceira classe concreta, chamada `AdvisoryMessageCode` está disponível para fornecer as constantes usadas para distinguir os códigos de mensagem retornados do servidor, sempre que aplicável.

O restante desta seção descreve os métodos que compõem a API do Interact.

Pré-requisitos da serialização do Java sobre HTTP

O adaptador cliente Java usa a serialização do Java sobre HTTP.

Os pré-requisitos para usar o adaptador cliente Java para a serialização do Java sobre HTTP são:

1. Inclua os seguintes arquivos no `CLASSPATH`:
   `Interact_Home/lib/interact_client.jar`

2. Todos os objetos transmitidos e retornados entre o cliente e o servidor podem ser localizados no pacote `com.unicacorp.interact.api`. Para obter mais detalhes, consulte a API Javadoc do Interact instalada no servidor de runtime em `Interact_Home/docs/apiJavaDoc`. É possível visualizar o Javadoc abrindo o arquivo `index.html` nesse local com qualquer navegador da web.
3. Para obter uma instância da classe `InteractAPI`, chame o método estático `getInstance` com a URL do servidor de runtime do Interact.

Pré-requisitos SOAP

Antes que seja possível acessar o servidor de runtime com SOAP, devem ser executadas várias tarefas de pré-requisito para configurar o ambiente.

Importante: Os testes de desempenho mostram que o adaptador de serialização do Java é executado em uma taxa muito maior do que um cliente SOAP gerado. Para obter um desempenho melhor, use o adaptador de serialização do Java sempre que possível.

Para acessar o servidor de runtime com SOAP, você deve fazer o seguinte:

1. Converta o WSDL da API do Interact com o kit de ferramentas de SOAP de sua escolha.

   O WSDL da API do Interact é instalado com o Interact no diretório `Interact/conf`.

   Ao configurar o SOAP usado os arquivos XML WSDL, você deverá modificar as URLs com o nome do host e a porta do servidor de runtime.

   O texto do WSDL está disponível no final do Guia de Administração do Interact.
2. Instale e configure o servidor de runtime.

   O servidor de runtime deverá estar em execução para que a integração seja totalmente testada.
3. Verifique se você está usando a versão do SOAP correta.

   O Interact usa axis2 1.3 como a infraestrutura de SOAP no servidores de runtime do Interact. Para obter detalhes sobre quais versões do SOAP o axis2 1.3 suporta, consulte o seguinte website:

   Apache Axis2

   O Interact foi testado com os clientes axis2, XFire, JAX-WS-Ri, DotNet, SOAPUI e IBM RAD SOAP.

Pré-requisitos do REST

Um método para chamar a API do Interact é usando chamadas do formato JSON (JavaScript Object Notation) sobre HTTP, mencionado aqui como a API REST. A API REST tem a vantagem de ter melhor desempenho do que o SOAP, embora o adaptador de serialização do Java ainda seja o método mais rápido para chamadas de API do Interact.

Antes de iniciar o uso da API REST, esteja ciente do seguinte:

- A URL que suporta chamadas REST para a API do Interact é:

  `http://Interact_Runtime_Server:PORT/interact/servlet/RestServlet`, substituindo o nome do host ou endereço IP real do servidor de runtime do Interact e a porta na qual o Interact é implementado.
- Há duas classes do Interact específicas para a API REST: `RestClientConnector`, que funciona como um auxiliar para se conectar a uma instância de tempo de

execução do Interact por meio do REST com o formato JSON, e `RestFieldConstants`, que descreve o formato subjacente da mensagem JSON usada para solicitações e respostas da API.

- Um cliente REST de amostra é fornecido em `Interact _Home/samples/javaApi/InteractRestClient.java`. Embora o código de amostra seja um exemplo simples, ele deve fornecer um bom ponto de início para demonstrar como a API REST é usada.
- Para obter uma descrição completa das classes da API REST juntamente com todas as outras informações da API do Interact, consulte o Javadoc instalado no servidor de runtime em `Interact_Home/docs/apiJavaDoc`.
- A API REST retorna SessionIDs e as mensagens no formato HTML de escape e não no formato Unicode.

Além das informações mencionadas aqui, a API REST suporta todos os métodos que são suportados pelos outros protocolos para usar a API do Interact.

JavaDoc da API

Além do Guia do Administrador do Interact, o Javadoc da API do Interact é instalado com o servidor de runtime. O Javadoc é instalado para referência no diretório `Interact_Home/docs/apiJavaDoc`.

Exemplos de API

Todos os exemplos no guia foram criados com o adaptador de serialização do Java sobre HTTP. As classes geradas a partir do WSDL podem variar com base no kit de ferramentas do SOAP e nas opções selecionadas. Se você estiver usando SOAP, esses exemplos poderão não funcionar da mesma forma no seu ambiente.

Trabalhando com dados da sessão

Quando você inicia uma sessão com o método `startSession`, os dados da sessão são carregados na memória. Durante a sessão, é possível ler e gravar os dados da sessão (que é um superconjunto dos dados do perfil estático).

A sessão contém os seguintes dados:

- Dados do perfil estático
- Designações de segmento
- Dados em tempo real
- Recomendações de oferta

Todos os dados da sessão ficam disponíveis até que você chame o método `endSession` ou que o tempo `sessionTimeout` seja decorrido. Quando a sessão termina, todos os dados não salvos explicitamente no histórico de contatos ou respostas ou alguma outra tabela de banco de dados são perdidos.

Os dados são armazenados como um conjunto de pares nome-valor. Se os dados forem lidos a partir de uma tabela de banco de dados, o nome será a coluna da tabela.

É possível criar esses pares nome-valor ao longo do trabalho com a API do Interact. Não é necessário declarar todos os pares nome-valor em uma área global. Se você configurar novos parâmetros de evento como pares nome-valor, o ambiente de tempo de execução incluirá os pares nome-valor nos dados da sessão. Por exemplo, se você usar os parâmetros de evento com o método `postEvent`, o

ambiente de tempo de execução incluirá os parâmetros de evento nos dados da sessão, mesmo que os parâmetros de evento não estejam disponíveis nos dados do perfil. Esses dados somente existem nos dados da sessão.

É possível sobrescrever os dados da sessão a qualquer momento. Por exemplo, se parte do perfil do cliente incluir `creditScore`, será possível transmitir um parâmetro de evento usando o tipo customizado `NameValuePair`. Na classe `NameValuePair`, é possível usar os métodos `setName` e `setValueAsNumeric` para alterar o valor. O nome precisa corresponder. Nos dados da sessão, o nome não faz distinção entre maiúsculas e minúsculas. Portanto, o nome `creditscore` ou `CrEdItScOrE` sobrescreveria `creditScore`.

Somente os últimos dados gravados nos dados da sessão são mantidos. Por exemplo, `startSession` carrega os dados do perfil para o valor de `lastOffer`. Um método `postEvent` substitui o `lastOffer`. Em seguida, um segundo método `postEvent` substitui `lastOffer`. O ambiente de tempo de execução somente mantém os dados gravados pelo segundo método `postEvent` nos dados da sessão.

Quando a sessão terminar, os dados serão perdidos, a menos que tenham sido feitas considerações especiais, como usar um processo de Captura instantânea no fluxograma interativo para gravar os dados em uma tabela de banco de dados. Se você estiver planejando usar processos de Captura instantânea, lembre-se de que os nomes precisam corresponder às limitações do banco de dados. Por exemplo, se somente forem permitidos 256 caracteres para o nome de uma coluna, o nome do par nome-valor não deverá exceder 256 caracteres.

Sobre a classe InteractAPI

A classe InteractAPI contém os métodos usados para integrar o ponto de contato ao servidor de runtime. Todos os outros métodos e classes e na API do Interact suportam os métodos nessa classe.

Você deve compilar a implementação com relação ao `interact_client.jar` localizado no diretório `lib` da instalação do ambiente de tempo de execução do Interact.

endSession

O método `endSession` marca o fim da sessão de tempo de execução. Quando o servidor de runtime receber este método, o servidor de runtime registrará no histórico, limpará a memória e assim por diante.

`endSession(String sessionID)`

- **sessionID** - Sequência exclusiva que identifica a sessão.

Se o método endSession não for chamado, as sessões de tempo de execução atingirão o tempo limite. O período de tempo limite é configurável com a propriedade `sessionTimeout`.

Valor de retorno

O servidor de runtime responde ao método `endSession` com o objeto `Response` com os atributos a seguir preenchidos:

- SessionID
- ApiVersion
- StatusCode

- AdvisoryMessages

Exemplo

O exemplo a seguir mostra o método endSession e como é possível analisar a resposta. sessionId é a mesma sequência para identificar a sessão usada pela chamada startSession que iniciou esta sessão.

```
response = api.endSession(sessionId);
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("endSession call processed with no warnings or errors");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("endSession call processed with a warning");
    }
    else
    {
        System.out.println("endSession call processed with an error");
    }
    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("endSession",
   response.getAdvisoryMessages());
```

executeBatch

O método executeBatch permite executar vários métodos com uma única solicitação para o servidor de runtime.

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

- **sessionID**- uma sequência que identifica o ID de sessão. Esse ID de sessão é usado para todos os comandos executados por esta chamada de método.
- **commandImpl[]**- uma matriz de objetos CommandImpl, uma para cada comando que desejar executar.

O resultado de chamar esse método é equivalente a chamar explicitamente cada método na matriz Comando. Esse método minimiza o número de solicitações reais ao servidor de runtime. O servidor de runtime executa cada método em série; para cada chamada, qualquer erro ou aviso é capturado no objeto Resposta que corresponde a essa chamada de método. Se um erro for encontrado, executeBatch continua com o resto das chamadas no lote. Se a execução de qualquer método resultar em um erro, o status de nível superior para o objeto BatchResponse refletirá esse erro. Se nenhum erro tiver ocorrido, o status de nível superior refletirá quaisquer avisos que possam ter ocorrido. Se nenhum aviso tiver ocorrido, o status de nível superior refletirá uma execução bem-sucedida do lote.

Valor de retorno

O servidor de runtime responde a executeBatch com um objeto BatchResponse.

Exemplo

O exemplo a seguir mostra como chamar todos os métodos getOffer e postEvent com uma única chamada executeBatch e uma sugestão de como lidar com a resposta.

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

```
int numberRequested=1;
String eventName = "logOffer";

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

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

/** Build command array */
Command[] commands =
{
  getOffersCommand,
  postEventCommand,
};

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

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

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

Compondo solicitações XML executeBatch() para a API SOAP do Interact

Use estas etapas para compor solicitações XML `executeBatch()` para a API SOAP de Interact.

Sobre Esta Tarefa

A XML de solicitação para uma chamada API SOAP de operação única (`startSession`, `getOffers`, `setAudience`, `endSession`, e assim por diante) não deve ser copiada ou colada diretamente em uma chamada `executeBatch()` de operação múltipla. Os subcomandos nas chamadas `executeBatch()` possuem estruturas de solicitação WSDL e XML levemente diferentes do que aquelas das chamadas de API de operação única. As diferenças estruturais causam respostas de

falha do servidor se os elementos XML são copiados e colados a partir de solicitação de API de operação única em solicitações `executeBatch` de operação múltipla.

Respostas de falha de amostra:

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

Use estas etapas para compor uma solicitação XML `executeBatch()`. É possível fazer referência a solicitações de chamada de API de operação única para valores de parâmetro durante estas etapas, mas não copiar e colar elementos XML.

Procedimento

1. Use uma ferramenta de processamento WSDL (por exemplo, SoapUI) para criar uma solicitação XML `executeBatch()` bem formada a partir do arquivo WSDL de Interact.
2. Inclua subcomandos na solicitação após a definição WSDL para elementos-filhos `executeBatch()`.
3. Conclua os argumentos do subcomando após a definição WSDL para os elementos filhos `executeBatch()`.

getInstance

O método `getInstance` cria uma instância da API do Interact que comunica-se com o servidor de runtime especificado.

```
getInstance(String URL)
```

Importante: Cada aplicativo que você gravar usando a API do Interact deve chamar `getInstance` para instanciar um objeto InteractAPI que é mapeado para um servidor de runtime especificado pelo parâmetro URL.

Para grupos de servidores, se você estiver usando um balanceador de carga, use o nome do host e porta que configurar com o balanceador de carga. Se você não tiver um balanceador de carga, você terá que incluir a lógica para girar entre os servidores de runtime disponíveis.

Este método é aplicável para a serialização Java sobre o adaptador HTTP apenas. Não há método correspondente definido no WSDL SOAP. Cada implementação do cliente SOAP possui sua própria forma de estabelecer a URL de terminal.

- **URL** - Uma sequência que identifica a URL para a instância do tempo de execução. Por exemplo, `http://localhost:7001/Interact/servlet/InteractJSService`.

Valor de retorno

O servidor de runtime retorna a InteractAPI.

Exemplo

O exemplo a seguir mostra como instanciar um objeto InteractAPI que aponta para uma instância do servidor de runtime em execução na mesma máquina que seu ponto de contato.

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

getOffers

O método `getOffers` permite solicitar ofertas do servidor de runtime.

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

- **sessionID** - uma sequência que identifica a sessão atual.
- **interactionPoint**- uma sequência que identifica o nome do ponto de interação que este método referencia.

  Nota: Esse nome deve corresponder exatamente ao nome do ponto de interação definido no canal interativo.
- **numberOfOffers**- um número inteiro que identifica o número de ofertas solicitadas.

O método `getOffers` aguarda o número de milissegundos definidos na propriedade `segmentationMaxWaitTimeInMS` para que toda a ressegmentação seja concluída antes da execução. Portanto, se você chamar um método `postEvent` que aciona uma ressegmentação ou um método `setAudience` imediatamente antes de uma chamada `getOffers`, pode haver um atraso.

Valor de retorno

O servidor de runtime responde a `getOffers` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

Exemplo

Este exemplo mostra a solicitação de uma única oferta para o ponto de interação do Banner 1 da Página de Visão Geral e uma forma de manipular a resposta.

`sessionId` é a mesma sequência para identificar a sessão de tempo de execução usada pela chamada `startSession` que começou esta sessão.

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

/** Make the call */
response = api.getOffers(sessionId, interactionPoint, numberRequested);

/** Process the response appropriately */
 // check if response is successful or not
 if(response.getStatusCode() == Response.STATUS_SUCCESS)
 {
  System.out.println("getOffers call processed with no warnings or errors");

        /** Check to see if there are any offers */
        OfferList offerList=response.getOfferList();

        if(offerList.getRecommendedOffers() != null)
        {
            for(Offer offer : offerList.getRecommendedOffers())
            {
```

```
            // print offer
            System.out.println("Offer Name:"+offer.getOfferName());
        }
    }
    else // count on the default Offer String
    System.out.println("Default offer:"+offerList.getDefaultString());
}
else if(response.getStatusCode() == Response.STATUS_WARNING)
{
    System.out.println("getOffers call processed with a warning");
}
else
{
    System.out.println("getOffers call processed with an error");
}
// For any non-successes, there should be advisory messages explaining why
if(response.getStatusCode() != Response.STATUS_SUCCESS)
    printDetailMessageOfWarningOrError("getOffers",
response.getAdvisoryMessages());
```

getOffersForMultipleInteractionPoints

O método `getOffersForMultipleInteractionPoints` permite solicitar ofertas do servidor de runtime para diversos IPs com deduplicação.

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

- **sessionID** - uma sequência que identifica a sessão atual.
- **requestStr** - uma sequência que fornece uma matriz de objetos `GetOfferRequest`.

  Cada objeto `GetOfferRequest` especifica:

  – **ipName** - O nome do ponto de interação (IP) para o qual o objeto está solicitando ofertas

  – **numberRequested** - O número de ofertas exclusivas que ele precisa para o IP especificado

  – **offerAttributes** - Requisitos nos atributos das ofertas entregues que usam uma instância de `OfferAttributeRequirements`

  – **duplicationPolicy** - ID de política de duplicação para as ofertas que serão entregues

    As políticas de duplicação determinam se as ofertas duplicadas serão retornadas em diferentes pontos de interação em uma única chamada de método. (*Em* um ponto de interação individual, as ofertas duplicadas nuunca são retornadas.) Atualmente, duas políticas de duplicação são suportadas.

    - NO_DUPLICATION (valor de ID = 1). Nenhuma das ofertas que foram incluídas nas instância `GetOfferRequest` anteriores será incluída nesta instância `GetOfferRequest` (isto é, o Interact aplicará a deduplicação).
    - ALLOW_DUPLICATION (valor de ID = 2). Qualquer uma das ofertas que atendem aos requisitos especificados nesta instância `GetOfferRequest` será incluída. As ofertas que foram incluídas nas instâncias `GetOfferRequest` anteriores não serão reconciliadas.

  A ordem das solicitações no parâmetro de matriz também é a ordem de prioridade em que as ofertas estão sendo entregues.

  Por exemplo, suponha que os IPs na solicitação sejam IP1, depois IP2, que nenhuma oferta duplicada seja permitida (um ID de política de duplicação = 1) e cada um está representando duas ofertas. Se o Interact localizar ofertas A, B e C para IP1 e ofertas A e D para IP2, a resposta conterá as ofertas A e B para IP1 e apenas a oferta D para IP2.

Observe também que quando o ID de política de duplicação for 1, as ofertas que foram entregues por um IP com prioridade mais alta não serão entregues por esse IP.

O método `getOffersForMultipleInteractionPoints` aguarda o número de milissegundos definidos na propriedade `segmentationMaxWaitTimeInMS` até que toda a ressegmentação seja concluída antes da execução. Portanto, se você chamar um método `postEvent` que aciona uma ressegmentação ou um método `setAudience` imediatamente antes de uma chamada `getOffers`, pode haver um atraso.

Valor de retorno

O servidor de runtime responde a `getOffersForMultipleInteractionPoints` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- matriz de OfferList
- SessionID
- StatusCode

Exemplo

```
InteractAPI api = InteractAPI.getInstance("url");
 String sessionId = "123";
 String requestForIP1 = "{IP1,5,1,(5,attr1=1|numeric;attr2=value2|string,
     (3,attr3=value3|string)(3,attr4=4|numeric))}";
 String requestForIP2 = "{IP2,3,2,(3,attr5=value5|string)}";
 String requestForIP3 = "{IP3,2,1}";
 String requestStr = requestForIP1 + requestForIP2 + requestForIP3;
 Response response = api.getOffersForMultipleInteractionPoints(sessionId,
          requestStr);

 if (response.getStatusCode() == Response.STATUS_SUCCESS) {
  // Check to see if there are any offers
  OfferList[] allOfferLists = response.getAllOfferLists();
  if (allOfferLists != null) {
   for (OfferList ol : allOfferLists) {
    System.out.println
("The following offers are delivered for interaction
               point " + ol.getInteractionPointName() + ":");
    for (Offer o : ol.getRecommendedOffers()) {
     System.out.println(o.getOfferName());
    }
   }
  }
 }
 else {
  System.out.println("getOffersForMultipleInteractionPoints() method calls
          returns an error with code: " + response.getStatusCode());
 }
```

Observe que a sintaxe de `requestStr` é a seguinte:

```
requests_for_IP[<requests_for_IP]
```

em que

```
<requests_for_IP> = {ip_name,number_requested_for_this_ip,
     dupe_policy[,child_requirements]]}
attribute_requirements = (number_requested_for_these_attribute_requirements
```

```
     [,attribute_requirement[;individual_attribute_requirement])
     [,(attribute_requirements))
individual_attribute_requirement = attribute_name=attribute_value | attribute_type
```

No exemplo mostrado acima, `requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))})` significa, para o ponto de interação chamado IP1, entrega no máximo 5 ofertas distintas que não podem ser retornadas para nenhum outro ponto de interação durante esta mesma chamada de método. Todas as 5 ofertas devem ter um atributo numérico chamado `attr1` que deve ter o valor 1 e deve ter um atributo de sequência chamado `attr2` que deve ter o valor *value2*. Dessas 5 ofertas, no máximo 3 devem ter um atributo de sequência chamado `attr3` que deve ter o valor *value3* e no máximo 3 devem ter um atributo numérico chamado `attr4` que deve ter o valor *4*.

Os tipos de atributos permitidos são numéricos, sequência e data/hora e o formato de um valor de atributo de data/hora deve ser `MM/dd/aaaa HH:mm:ss`. Para recuperar as ofertas retornadas, use o método `Response.getAllOfferLists()`. Para ajudar a entender a sintaxe, o exemplo em `setGetOfferRequests` constrói a mesma instância de `GetOfferRequests`, ao usar objetos Java, o que é preferencial.

getProfile

O método `getProfile` permite recuperar o perfil e informações temporais sobre o visitante que está visitando o ponto de contato.

`getProfile(String sessionID)`

- **sessionID** - uma sequência que identifica o ID de sessão.

Valor de retorno

O servidor de runtime responde a `getProfile` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

Exemplo

A seguir, está um exemplo de uso de `getProfile` e uma forma de manipular a resposta.

`sessionId` é a mesma sequência para identificar a sessão usada pela chamada `startSession` que iniciou esta sessão.

```
response = api.getProfile(sessionId);
/** Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("getProfile call processed with no warnings or errors");
        // Print the profile - it's just an array of NameValuePair objects
        for(NameValuePair nvp : response.getProfileRecord())
        {
            System.out.println("Name:"+nvp.getName());
            if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
            {
                System.out.println("Value:"+nvp.getValueAsDate());
```

```
            }
            else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
            {
                System.out.println("Value:"+nvp.getValueAsNumeric());
            }
            else
            {
                System.out.println("Value:"+nvp.getValueAsString());
            }
        }
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getProfile call processed with a warning");
    }
    else
    {
        System.out.println("getProfile call processed with an error");
    }
    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getProfile",
   response.getAdvisoryMessages());
```

getVersion

O método `getVersion` retorna a versão da implementação atual do servidor de runtime do Interact.

```
getVersion()
```

A melhor prática é usar este método ao inicializar o ponto de contato com a API do Interact.

Valor de retorno

O servidor de runtime responde ao `getVersion` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- StatusCode

Exemplo

Este exemplo mostra uma forma simples de chamar `getVersion` e processar os resultados.

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

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

postEvent

O método postEvent permite executar qualquer evento definido no canal interativo.

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

- **sessionID**: uma sequência que identifica o ID de sessão.
- **eventName**: uma sequência que identifica o nome do evento.

  Nota: O nome do evento deve corresponder ao nome do evento conforme definido no canal interativo. Este nome não faz distinção entre maiúsculas e minúsculas.
- **eventParameters**. Objetos NameValuePairImpl que identificam quaisquer parâmetros que precisam ser passados com o evento. Esses valores serão armazenados nos dados da sessão.

  Se este evento acionar a ressegmentação, você deve assegurar que todos os dados necessários aos fluxogramas interativos estejam disponíveis nos dados da sessão. Se quaisquer um desses valores não tiver sido preenchido por ações anteriores (por exemplo, de startSession ou setAudience ou carregando a tabela de perfis), você deve incluir um eventParameter para cada valor ausente. Por exemplo, se você tiver configurado todas as tabelas de perfis para carregar na memória, você deve incluir um NameValuePair para dados temporais necessários para os fluxogramas interativos.

  Se você estiver usando mais de um nível de público, provavelmente terá diferentes conjuntos de eventParameters para cada nível de público. Você deve incluir alguma lógica para assegurar que está selecionando o conjunto correto de parâmetros para o nível de público.

  Importante: Se este evento for registrado no histórico de respostas, você deverá passar o código de tratamento para a oferta. Você deve definir o nome para NameValuePair como "UACIOfferTrackingCode".

  É possível apenas passar um código de tratamento por evento. Se você não passar o código de tratamento para um contato de oferta, o Interact registrará um contato de oferta para cada oferta na última lista de ofertas recomendadas. Se você não passar o código de tratamento para uma resposta, o Interact retornará um erro.
- Há vários outros parâmetros reservados usados com postEvent e outros métodos e são discutidos posteriormente nesta seção.

Qualquer solicitação de ressegmentação ou gravação no histórico de respostas ou contato não aguardará uma resposta.

A ressegmentação não limpa resultados da segmentação anterior para o nível de audiência atual. É possível usar o parâmetro UACIExecuteFlowchartByName para definir fluxogramas específicos para execução. O método getOffers aguarda a ressegmentação ser concluída antes de executar. Portanto, se você chamar um método postEvent, que aciona a ressegmentação imediatamente antes de uma chamada getOffers, pode haver um atraso.

Valor de retorno

O servidor de runtime responde a `postEvent` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Exemplo

O exemplo a seguir `postEvent` mostra o envio de novos parâmetros para um evento que aciona a ressegmentação e uma forma de manipular a resposta.

`sessionId` é a mesma sequência para identificar a sessão usada pela chamada `startSession` que iniciou esta sessão.

```
String eventName = "SearchExecution";

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

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

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

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

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

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

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

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

/** Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
```

```
{
    System.out.println("postEvent call processed with no warnings or errors");
}
else if(response.getStatusCode() == Response.STATUS_WARNING)
{
    System.out.println("postEvent call processed with a warning");
}
else
{
    System.out.println("postEvent call processed with an error");
}

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

setAudience

O método `setAudience` permite configurar o ID de público e o nível para um visitante.

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

- **sessionID** - uma sequência que identifica o ID de sessão.
- **audienceID** - uma matriz de objetos `NameValuePairImpl` que define o ID de público.
- **audienceLevel** - uma sequência que define o nível de público.
- **parameters** - objetos `NameValuePairImpl` que identificam quaisquer parâmetros que precisam ser transmitidos com `setAudience`. Estes valores são armazenados nos dados da sessão e podem ser usados para segmentação.

  Você deve ter um valor para cada coluna em seu perfil. Este é um superconjunto de todas as colunas em todas as tabelas definidas para o canal interativo e quaisquer dados em tempo real. Se você já tiver preenchido todos os dados da sessão com `startSession` ou `postEvent`, não será necessário enviar novos parâmetros.

O método `setAudience` aciona uma ressegmentação. O método `getOffers` aguarda a ressegmentação ser concluída antes de executar. Portanto, se você chamar um método `setAudience` imediatamente antes de uma chamada `getOffers`, pode haver um atraso.

O método `setAudience` também carrega os dados do perfil para o ID de público. É possível usar o método `setAudience` para forçar um recarregamento dos mesmos dados do perfil carregados pelo método `startSession`.

Valor de retorno

O servidor de runtime responde a `setAudience` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Exemplo

Para este exemplo, o nível de público permanece o mesmo, mas o ID é alterado, como se um usuário anônimo efetuasse login e se tornasse conhecido.

`sessionId` e `audienceLevel` são as mesmas sequências para identificar a sessão e o nível de público usado pela chamada `startSession` que iniciou esta sessão.

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

NameValuePair[] newAudienceId = { custId2 };

/** Os parâmetros podem ser transmitidos também.  Para este exemplo,
não há parâmetros,
* portanto, transmita null */
NameValuePair[] noParameters=null;

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

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

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

setDebug

O método `setDebug` permite configurar o nível de detalhamento da criação de log para todos os caminhos de código da sessão.

`setDebug(String sessionID, boolean debug)`

- **sessionID**- uma sequência que identifica o ID de sessão.
- **debug** - um booleano que permite ou não informações sobre depuração. Os valores válidos são `true` ou `false`. Se true, o Interact registrará informações sobre depuração no log do servidor de runtimeg.

Valor de retorno

O servidor de runtime responde ao `setDebug` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Exemplo

O exemplo a seguir mostra a alteração do nível de depuração da sessão.

`sessionId` é a mesma sequência para identificar a sessão usada pela chamada `startSession` que iniciou esta sessão.

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

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

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

startSession

O método `startSession` cria e define uma sessão de tempo de execução.

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

`startSession` pode acionar até cinco ações:

- crie uma sessão de tempo de execução.
- carregue dados do perfil do visitante para o nível de público atual na sessão de tempo de execução, incluindo quaisquer tabelas de dimensões marcadas para carregamento no mapeamento de tabela definido para o canal interativo.
- segmentação de acionador, executando todos os fluxogramas interativos para o nível de público atual.
- carregue os dados de supressão de oferta na sessão, se a propriedade `enableOfferSuppressionLookup` estiver configurada como true.
- carregue os dados de substituição de pontuação na sessão, se a propriedade `enableScoreOverrideLookup` estiver configurada como true.

O método `startSession` requer os parâmetros a seguir:

- **sessionID**- uma sequência que identifica o ID de sessão. Você deve definir o ID de sessão. Por exemplo, você poderia usar uma combinação do ID do cliente e registro de data e hora.

  Para definir o que faz uma sessão de tempo de execução, um ID de sessão precisa ser especificado. Este valor é gerenciado pelo cliente. Todas as chamadas

de método para o mesmo ID de sessão têm de ser sincronizadas pelo cliente. O comportamento para chamadas API simultâneas com o mesmo ID de sessão é indefinido.

- **relyOnExistingSession** - um booleano que define se esta sessão usa uma sessão nova ou existente. Os valores válidos são `true` ou `false`. Se `true`, você deve fornecer um ID de sessão existente com o método `startSession`. Se `false`, você deve fornecer um novo ID de sessão.

  Se você configurar `relyOnExistingSession` para `true` e existir uma sessão, o ambiente de tempo de execução usará os dados da sessão existentes e não recarregará nenhum dado ou segmentação de acionador. Se a sessão não existir, o ambiente de tempo de execução criará uma nova sessão, incluindo o carregamento de dados e o acionamento da segmentação. Configurar `relyOnExistingSession` para true e usá-lo com todas as chamadas `startSession` é útil se seu ponto de contato tiver uma sessão de comprimento mais longo que a sessão de tempo de execução. Por exemplo, uma sessão de website está ativa há 2 horas, mas a sessão de tempo de execução só está ativa há 20 minutos.

  Se você chamar `startSession` duas vezes com o mesmo ID de sessão, todos os dados da sessão da primeira chamada `startSession` serão perdidos se `relyOnExistingSession` for false.
- **debug** - um booleano que permite ou não informações sobre depuração. Os valores válidos são `true` ou `false`. Se true, Interact registrará informações sobre depuração nos logs do servidor de runtime. O sinalizador de depuração é configurado para cada sessão individualmente. Portanto, é possível rastrear dados de depuração para uma sessão individual.
- **interactiveChannel**- uma sequência que define o nome do canal interativo a que esta sessão se refere. Este nome deve corresponder ao nome do canal interativo definido em Campaign exatamente.
- **audienceID** - uma matriz de objetos NameValuePairImpl em que os nomes devem corresponder aos nomes de coluna física de qualquer tabela que contenha o ID de público.
- **audienceLevel** - uma sequência que define o nível de público.
- **parameters** - objetos NameValuePairImpl que identificam quaisquer parâmetros que precisam ser passados com `startSession`. Estes valores são armazenados nos dados da sessão e podem ser usados para segmentação.

  Se você tiver vários fluxogramas interativos para o mesmo nível de público, deverá incluir um superconjunto de todas as colunas em todas as tabelas. Se você configurar o tempo de execução para carregar a tabela de perfis e a tabela de perfis contiver todas as colunas que você precisa, não será necessário passar parâmetros, a menos que queira sobrescrever os dados na tabela de perfis. Se sua tabela de perfis contiver um subconjunto das colunas necessárias, você deverá incluir as colunas ausentes como parâmetros.

Se `audienceID` ou `audienceLevel` forem inválidos e `relyOnExistingSession` for false, a chamada `startSession` falhará. Se `interactiveChannel` for inválido, `startSession` falhará, independentemente de `relyOnExistingSession` ser true ou false.

Se `relyOnExistingSession` for true e você fizer uma segunda chamada `startSession` usando o mesmo `sessionID`, mas a primeira sessão tiver expirado, Interact criará uma nova sessão.

Se `relyOnExistingSession` for true e você fizer uma segunda chamada `startSession` usando o mesmo `sessionID`, mas um `audienceID` ou `audienceLevel` diferente, o servidor de runtime alterará o público para a sessão existente.

Se relyOnExistingSession for true e você fizer uma segunda chamada startSession usando o mesmo sessionID, mas um interactiveChannel diferente, o servidor de runtime criará uma nova sessão.

Valor de retorno

O servidor de runtime responderá a startSession com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages (se StatusCode não for igual a 0)
- ApiVersion
- SessionID
- StatusCode

Exemplo

O exemplo a seguir mostra uma forma de chamar startSession.

```
String sessionId="MySessionID-123";
String audienceLevel="Customer";
NameValuePair custId = new NameValuePairImpl();
custId.setName("CustomerId");
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
boolean relyOnExistingSession=false;
boolean initialDebugFlag=true;
String interactiveChannel="Accounts Website";
NameValuePair parm1 = new NameValuePairImpl();
parm1.setName("SearchString");
parm1.setValueAsString("");
parm1.setValueDataType(NameValuePair.DATA_TYPE_STRING);

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

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

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

NameValuePair parm5 = new NameValuePairImpl();
parm5.setName("TxAcctValueChange");
parm5.setValueAsNumeric(0.0);
parm5.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

NameValuePair parm6 = new NameValuePairImpl();
parm6.setName("PageTopic");
parm6.setValueAsString("");
parm6.setValueDataType(NameValuePair.DATA_TYPE_STRING);

/** Specifying the parameters (optional) */
NameValuePair[] initialParameters = { parm1,
 parm2,
 parm3,
 parm4,
 parm5,
```

```
 parm6
 };

/** Make the call */
response = api.startSession(sessionId, relyOnExistingSession, initialDebugFlag,
 interactiveChannel, initialAudienceId, audienceLevel, initialParameters);

/** Process the response appropriately */
processStartSessionResponse(response);
```

processStartSessionResponse é um método que manipula o objeto de resposta retornado por startSession.

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

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

Deduplicação de oferta entre atributos de oferta

Usando a interface de programação de aplicativos (API) do Interact, duas chamadas de API entregam ofertas: getOffers e getOffersForMultipleInteractionPoints. getOffersForMultipleInteractionPoints pode impedir o retorno de ofertas duplicadas no nível *OfferID*, mas não pode deduplicar ofertas entre categoria de oferta. Portanto, por exemplo, para que o Interact retorne apenas uma oferta de cada categoria de oferta, uma solução alternativa era necessária anteriormente. Com a introdução de dois parâmetros para a chamada API startSession, a deduplicação de oferta entre atributos de oferta, como categoria, agora é possível.

Esta lista resume os parâmetros que foram incluídos na chamada API startSession. Para obter mais informações sobre esses parâmetros ou qualquer aspecto da API do Interact, consulte o *IBM Interact Administrator's Guide* ou os arquivos Javadoc incluídos com sua instalação do Interact no <Interact_Home>/docs/apiJavaDoc.

- UACIOfferDedupeAttribute. Para criar uma chamada API startSession com deduplicação de oferta, para que as chamadas subsequentes getOffer retornem apenas uma oferta de cada categoria, você deve incluir o parâmetro UACIOfferDedupeAttribute como parte da chamada API. É possível especificar um parâmetro no formato name,value,type, conforme mostrado aqui:

  UACIOfferDedupeAttribute,<attributeName>,string

  Neste exemplo, você substituiria *<attributeName>* pelo nome do atributo de oferta que deseja usar como critério para a deduplicação, como Categoria.

Nota: O Interact examina as ofertas que têm o mesmo valor de atributo que especificar (como Categoria) e deduplica para remover todos menos a oferta que possui a pontuação mais alta. Se as ofertas que possuem o atributo duplicado também possuírem pontuações idênticas, o Interact retornará uma seleção aleatória entre as ofertas correspondentes.

- `UACINoAttributeDedupeIfFewerOf`. Ao incluir `UACIOfferDedupeAttribute` na chamada `startSession`, também é possível configurar este parâmetro `UACINoAttributeDedupeIfFewerOf` para especificar o comportamento em casos em que a lista de ofertas após a deduplicação não contém mais ofertas suficientes para atender à solicitação original.

  Por exemplo, se você configurar `UACIOfferDedupeAttribute` para usar a categoria de oferta para deduplicar ofertas e sua chamada getOffers subsequente solicitar que oito ofertas sejam retornadas, a deduplicação poderá resultar em menos que oito ofertas elegíveis. Nesse caso, configurar o parâmetro `UACINoAttributeDedupeIfFewerOf` para true resultaria na inclusão de alguns dos duplicados na lista de elegíveis parar atender ao número solicitado de ofertas. Neste exemplo, se você configurar o parâmetro como false, o número de ofertas retornadas seria menor que o número solicitado.

  `UACINoAttributeDedupeIfFewerOf` está configurado como true por padrão.

Por exemplo, suponha que você especificou um parâmetro `startSession` que o critério de deduplicação é a oferta Categoria, conforme mostrado aqui:

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

Esses parâmetros juntos fazem com que o Interact deduplique ofertas com base no atributo de oferta "Categoria" e retorne apenas as ofertas deduplicadas mesmo que o número resultante de ofertas seja menor que o solicitado (`UACINoAttributeDedupeIfFewerOffer` é false).

Ao emitir uma chamada API `getOffers`, o conjunto original de ofertas elegíveis pode incluir essas ofertas:

- Category=Electronics: Oferta A1 com uma pontuação de 100 e Oferta A2 com uma pontuação de 50.
- Category=Smartphones: Oferta B1 com uma pontuação de 100, Oferta B2 com uma pontuação de 80 e Oferta B3 com uma pontuação de 50.
- Category=MP3Players: Oferta C1 com uma pontuação de 100, Oferta C2 com uma pontuação de 50.

Neste caso, havia duas ofertas duplicadas que correspondem à primeira categoria, três ofertas duplicadas que correspondem à segunda categoria e duas ofertas duplicadas que correspondem à terceira categoria. As ofertas que são retornadas seriam as ofertas de maior pontuação de cada categoria, que são Oferta A1, Oferta B1 e Oferta C1.

Se a chamada API `getOffers` solicitou seis ofertas, este exemplo configurou `UACINoAttributeDedupeIfFewerOffer` como false, portanto, apenas três ofertas seriam retornadas.

Se a chamada API `getOffers` solicitou seis ofertas e este exemplo omitiu o parâmetro `UACINoAttributeDedupeIfFewerOffer` ou especificamente o configurou como true, algumas das ofertas duplicadas seriam incluídas no resultado para atender ao número solicitado.

Parâmetros reservados

Há vários parâmetros reservados usados com a API do Interact. Alguns são necessários para o servidor de runtime e outros podem ser usados para recursos adicionais.

Recursos postEvent

Recurso	Parâmetro	Descrição
Registrar a tabela customizada	`UACICustomLoggerTableName`	O nome de uma tabela na origem de dados de tabelas de tempo de execução. Se você fornecer este a parâmetro um nome de tabela válido, o ambiente de tempo de execução gravará todos os dados da sessão na tabela selecionada. Todos os nomes de coluna na tabela que correspondem aos dados da sessão NameValuePair são preenchidos. O ambiente de tempo de execução preenche qualquer coluna que não corresponda a um par de nome-valor da sessão com um nulo. É possível gerenciar o processo que grava no banco de dados com as propriedades de configuração `customLogger`.
Diversos tipos de resposta	`UACILogToLearning`	Um número inteiro com o valor 1 ou 0. 1 indica que o ambiente de tempo de execução deve criar o log de evento como uma aceitação para o sistema de aprendizado ou ativar a supressão de oferta em uma sessão. 0 indica que o ambiente de tempo de execução não deve criar log do evento para o sistema de aprendizado ou ativar a supressão de oferta em uma sessão. Este parâmetro permite criar vários métodos `postEvent` criando logs de diferentes tipos de resposta sem influenciar o aprendizado. Não é necessário definir esse parâmetro para eventos configurados para registrar um contato, aceitação ou rejeição. Você deve usar esse parâmetro em conjunto com `UACIResponseTypeCode`. Se você não definir `UACILOGTOLEARNING`, o ambiente de tempo de execução assumirá o valor padrão de 0 (a menos que o evento acione um contato, aceitação ou rejeição de log).
	`UACIResponseTypeCode`	Um valor que representa um código do tipo de resposta. O valor deve ser uma entrada válida na tabela `UA_UsrResponseType`
Rastreamento de resposta	`UACIOfferTrackingCode`	O código de tratamento para a oferta. Você deve definir este parâmetro se o evento registrar o histórico de respostas ou contatos. É possível apenas passar um código de tratamento por evento. Se você não passar o código de tratamento para um contato de oferta, o ambiente de tempo de execução registrará um contato de oferta para cada oferta na última lista de ofertas recomendadas. Se você não passar o código de tratamento para uma resposta, o ambiente de tempo de execução retornará um erro. Se você configurar o rastreamento de resposta de sessão cruzada, será possível usar o parâmetro `UACIOfferTrackingcodeType` para definir que tipo de código de restreamento você usa que não o código de tratamento.
Rastreamento de resposta de sessão cruzada	`UACIOfferTrackingCodeType`	Um número que define o tipo de código de rastreamento. 1 é o código de tratamento padrão e 2 é o código de oferta. Todos os códigos devem ser entradas válidas na tabela `UACI_TrackingType`. É possível incluir outros códigos customizados nesta tabela.
Execução de fluxograma específico	`UACIExecuteFlowchartByName`	Se você definir este parâmetro para qualquer método que acionar a segmentação (`startSession`, `setAudience` ou um `postEvent` que aciona a ressegmentação), em vez de executar todos os fluxogramas para o nível de público atual, o Interact executa somente os fluxogramas denominados. É possível fornecer uma lista de fluxogramas separados por um caractere de barra vertical (|).

Parâmetros reservados do ambiente de tempo de execução

Os parâmetros reservados a seguir são usados pelo ambiente de tempo de execução. Não use esses nomes para seus parâmetros de evento.

- `UACIEventID`
- `UACIEventName`
- `UACIInteractiveChannelID`
- `UACIInteractiveChannelName`

- UACIInteractionPointID
- UACIInteractionPointName
- UACISessionID

Sobre a classe AdvisoryMessage

A classe advisoryMessage contém métodos que definem o código de mensagem de recomendação. O objeto de mensagem de recomendação está contido no objeto Resposta. Cada método no InteractAPI retorna um objeto Resposta. (Exceto o método executeBatch, que retorna um objeto batchResponse.)

Se houver um erro ou um aviso, o servidor Interact preencherá o objeto de mensagem de recomendação. O objeto de mensagem de recomendação contém os seguintes atributos:

- **DetailMessage** - uma descrição detalhada da mensagem de recomendação. Esse atributo pode não estar disponível para todas as mensagens de recomendação. Se estiver disponível, o DetailMessage poderá não ser localizado.
- **Mensagem** - uma descrição simples da mensagem de recomendação.
- **MessageCode** - um número de código para a mensagem de recomendação.
- **StatusLevel** - um número de código para a gravidade da mensagem de recomendação.

É possível recuperar os objetos advisoryMessage usando o método getAdvisoryMessages.

getDetailMessage

O método getDetailMessage retorna a descrição detalhada de um objeto Mensagem de Recomendação. Nem todas as mensagens têm uma mensagem detalhada.

```
getDetailMessage()
```

Valor de retorno

O objeto Mensagem de Recomendação retorna uma sequência.

Exemplo

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

getMessage

O método getMessage retorna a breve descrição de um objeto Mensagem de Recomendação.

```
getMessage()
```

Valor de retorno

O objeto Mensagem de Recomendação retorna uma sequência.

Exemplo

O método a seguir imprime a mensagem e a mensagem detalhada de um objeto AdvisoryMessage.

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

getMessageCode

O método `getMessageCode` retorna o código de erro interno associado a um objeto de Mensagem de recomendação se o nível de status for 2 (STATUS_LEVEL_ERROR).

```
getMessageCode()
```

Valor de retorno

O objeto AdvisoryMessage retorna um número inteiro.

Exemplo

O método a seguir imprime o código de mensagem de um objeto AdvisoryMessage.

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

getStatusLevel

O método `getStatusLevel` retorna o nível de status de um objeto do Advisory Message.

```
getStatusLevel()
```

Valor de retorno

O objeto do Advisory Message retorna um número inteiro.

- 0 - STATUS_LEVEL_SUCCESS- O método chamado concluído sem erros.
- 1 - STATUS_LEVEL_WARNING- O método chamado concluído com pelo menos um aviso (mas sem erros).
- 2 - STATUS_LEVEL_ERROR- O método chamado concluído não foi concluído com sucesso e possui pelo menos um erro.

Exemplo

O método a seguir imprime o nível de status de um objeto AdvisoryMessage.

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

Sobre a classe AdvisoryMessageCode

A classe `advisoryMessageCode` contém métodos que definem o código de mensagem de recomendação. É possível recuperar os códigos de mensagem de recomendação com o método `getMessageCode`.

Códigos de mensagem de recomendação

Você recupera os códigos de mensagem de recomendação com o método `getMessageCode`.

Esta tabela lista e descreve os códigos de mensagem de recomendação.

Código	Texto da mensagem	Descrição
1	INVALID_SESSION_ID	O ID de sessão não faz referência a uma sessão válida.
2	ERROR_TRYING_TO_ABORT_ SEGMENTATION	Ocorreu um erro ao tentar abortar a segmentação durante uma chamada `endSession`.
3	INVALID_INTERACTIVE_CHANNEL	O argumento que foi transmitido para o canal interativo não faz referência a um canal interativo válido.
4	INVALID_EVENT_NAME	O argumento que foi transmitido para o evento não faz referência a um evento válido para o canal interativo atual.
5	INVALID_INTERACTION_POINT	O argumento que foi transmitido para o ponto de interação não faz referência a um ponto de interação válido para o canal interativo atual.
6	ERROR_WHILE_MAKING_INITIAL_ SEGMENTATION_REQUEST	Foi encontrado um erro ao enviar uma solicitação de segmentação.
7	SEGMENTATION_RUN_FAILED	A segmentação foi executada parcialmente, mas resultou em um erro.
8	PROFILE_LOAD_FAILED	A tentativa de carregar as tabelas de perfil ou dimensões falhou.
9	OFFER_SUPPRESSION_LOAD_FAILED	A tentativa de carregar a tabela de supressão de ofertas falhou.
10	COMMAND_METHOD_ UNRECOGNIZED	Um método de comando que foi especificado para um comando em uma chamada `executeBatch` não é válido.
11	ERROR_TRYING_TO_POST_EVENT _PARAMETERS	Ocorreu um erro ao postar os parâmetros de evento.
12	LOG_SYSTEM_EVENT_EXCEPTION	Ocorreu uma exceção ao tentar enviar um evento do sistema (Sessão de encerrameto, Obter oferta, Obter perfil, Configurar público, Configurar depuração ou Iniciar sessão) para criação de log.
13	LOG_USER_EVENT_EXCEPTION	Ocorreu uma exceção ao tentar enviar o evento do usuário para criação de log.
14	ERROR_TRYING_TO_LOOK_UP_EVENT	Ocorreu um erro ao tentar consultar o nome do evento.
15	ERROR_TRYING_TO_LOOK_UP _INTERACTIVE_CHANNEL	Ocorreu um erro ao tentar consultar o nome do canal interativo.
16	ERROR_TRYING_TO_LOOK_UP _INTERACTION_POINT	Ocorreu um erro ao tentar consultar o nome do ponto de interação.
17	RUNTIME_EXCEPTION_ ENCOUNTERED	Foi encontrada uma exceção de tempo de execução inesperada.
18	ERROR_TRYING_TO_EXECUTE_ ASSOCIATED_ACTION	Ocorreu um erro ao tentar executar a ação associada (Ressegmentação do acionador, Contato de oferta de log, Aceitação de oferta de log ou Rejeição de oferta de log).
19	ERROR_TRYING_RUN_FLOWCHART	Ocorreu um erro ao tentar executar o fluxograma.
20	FLOWCHART_FAILED	Uma execução do fluxograma falhou.
21	FLOWCHART_ABORTED	Uma execução do fluxograma foi interrompida.
22	FLOWCHART_NEVER_RUN	Um fluxograma especificado nunca foi executado.
23	FLOWCHART_STILL_RUNNING	Um fluxograma ainda está em execução.

Código	Texto da mensagem	Descrição
24	ERROR_WHILE_READING_ PARAMETERS	Ocorreu um erro ao ler parâmetros.
25	ERROR_WHILE_LOADING_ RECOMMENDED_OFFERS	Erro ao carregar ofertas recomendadas
26	ERROR_WHILE_LOGGING_DEFAULT _TEXT_STATISTICS	Ocorreu um erro ao registrar estatísticas de texto padrão (o número de vezes que a sequência padrão para o ponto de interação foi exibida).
27	SCORE_OVERRIDE_LOAD_FAILED	A tabela de substituição de pontuação falhou ao carregar.
28	NULL_AUDIENCE_ID	O identificador de público está vazio.
29	UNRECOGNIZED_AUDIENCE_LEVEL	Um nível de público não reconhecido foi especificado.
30	MISSING_AUDIENCE_FIELD	Um campo de público está ausente.
31	INVALID_AUDIENCE_FIELD_TYPE	Um tipo de campo de público inválido foi especificado.
32	UNSUPPORTED_AUDIENCE_FIELD _TYPE	Tipo de campo de público não suportado
33	TIMEOUT_REACHED_ON_GET_ OFFERS_CALL	A chamada `getOffers` atingiu o tempo limite sem retornar ofertas.
34	INTERACT_INITIALIZATION_NOT_ COMPLETED_SUCCESSFULLY	A inicialização do servidor de runtime não foi concluída com sucesso.
35	SESSION_ID_UNDEFINED	O identificador de sessão é indefinido.
36	INVALID_NUMBER_OF_OFFERS_ REQUESTED	Um número inválido de ofertas foi solicitado.
37	NO_SESSION_EXIST_BUT_WILL_ CREATE_NEW_ONE	Nenhuma sessão existia, mas uma foi criada.
38	AUDIENCE_ID_NOT_FOUND_IN_ PROFILE_TABLE	O identificador do público especificado não está na tabela de perfis.
39	LOG_CUSTOM_LOGGER_EVENT_ EXCEPTION	Ocorreu uma exceção ao tentar enviar o evento de dados de criação de log customizado.
40	SPECIFIED_FLOWCHART_FOR_ EXECUTION_DOES_NOT_EXIST	O fluxograma especificado não pode ser executado, porque não existe.
41	AUDIENCE_NOT_DEFINED_IN_ CONFIGURATION	O público especificado não está definido na configuração.

Sobre a classe BatchResponse

A classe `BatchResponse` contém métodos que definem os resultados do método `executeBatch`.

O objeto Resposta em lote contém os seguintes atributos:

- **BatchStatusCode** - O valor de Código de status mais alto de todas as respostas solicitadas pelo método `executeBatch`.
- **Respostas** - Uma matriz dos objetos Resposta solicitados pelo método `executeBatch`.

getBatchStatusCode

O método `getBatchStatusCode` retorna o código de status mais alto da matriz de comandos executados pelo método `executeBatch`.

```
getBatchStatusCode()
```

Valor de retorno

O método `getBatchStatusCode` retorna um número inteiro.

- 0 - STATUS_SUCCESS- O método chamado concluído sem erros.
- 1 - STATUS_WARNING- O método chamado concluído com pelo menos um aviso (mas sem erros).
- 2 - STATUS_ERROR- O método chamado não foi concluído com sucesso e tem pelo menos um erro.

Exemplo

A amostra de código a seguir dá um exemplo de como recuperar `BatchStatusCode`.

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

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

getResponses

O método `getResponses` retorna a matriz de objetos de resposta que corresponde à matriz de comandos executados pelo método `executeBatch`.

```
getResponses()
```

Valor de retorno

O método `getResponses` retorna uma matriz de objetos `Response`.

Exemplo

O exemplo a seguir seleciona todas as respostas e imprime quaisquer mensagens de recomendação se o comando não foi bem-sucedido.

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

Sobre a interface de comando

O método `executeBatch` requer que seja transmitida uma matriz de objetos que implemente a interface de comando. Você deve usar a implementação padrão, `CommandImpl` para transmitir os objetos de Comando.

A tabela a seguir lista o comando, o método da classe InteractAPI que o comando representa e os métodos da interface de comando que devem ser usados para cada comando. Não é necessário incluir um ID de sessão porque o método `executeBatch` já inclui o ID da sessão.

Comando	Método da API do Interact	Métodos da interface de comando
COMMAND_ENDSESSION	endSession	None.

Comando	Método da API do Interact	Métodos da interface de comando
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	None.
COMMAND_GETVERSION	getVersion	None.
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

O método `setAudienceID` define o AudienceID para os comandos `setAudience` e `startSession`.

`setAudienceID(`*`audienceID`*`)`

- **audienceID**- uma matriz de objetos NameValuePair que definem o AudienceID.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método `executeBatch` que chama `startSession` e `setAudience`.

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

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

setAudienceLevel

O método `setAudienceLevel` define o Nível de público para os comandos `setAudience` e `startSession`.

`setAudienceLevel(`*`audienceLevel`*`)`

- *audienceLevel*- uma sequência que contém o Nível de público.

  Importante: O nome do *audienceLevel* deve corresponder ao nome do nível de público conforme definido no Campaign exatamente. Faz distinção entre maiúsculas e minúsculas.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método `executeBatch` que chama `startSession` e `setAudience`.

```
String audienceLevel="Customer";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceLevel(audienceLevel);
. . .
/** Build command array */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Make the call */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

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

setDebug

O método `setDebug` define o nível de depuração para o comando `startSession`.

`setDebug(`*`debug`*`)`

Se `true`, o servidor de runtime registrará informações sobre depuração no log do servidor de runtime. Se `false`, o servidor de runtime não registrará nenhuma informação sobre depuração. O sinalizador de depuração é configurado para cada sessão individualmente. Portanto, é possível rastrear os dados de depuração para uma sessão de tempo de execução individual.

- **debug**- um booleano (`true` ou `false`).

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método executeBatch que chama startSession e setDebug.

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

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

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

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

setEvent

O método setEvent define o nome do evento usado pelo comando postEvent.

setEvent(*event*)

- **event**- Uma sequência que contém o nome do evento.

Importante: O nome do *event* deve corresponder ao nome do evento conforme definido no canal interativo exatamente. Faz distinção entre maiúsculas e minúsculas.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método executeBatch que chama postEvent.

```
String eventName = "SearchExecution";

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

setEventParameters

O método setEventParameters define os parâmetros de evento usados pelo comando postEvent. Esses valores serão armazenados nos dados da sessão.

setEventParameters(*eventParameters*)

- **eventParameters**- uma matriz de objetos NameValuePair que define parâmetros de evento.

Por exemplo, se o evento estiver registrando uma oferta no histórico de contato, você deve incluir o código de tratamento da oferta.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método `executeBatch` que chama `postEvent`.

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

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

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

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

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

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

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

setGetOfferRequests

O método **setGetOfferRequests** configura o parâmetro para recuperar ofertas usadas pelo comando `getOffersForMultipleInteractionPoints`.

`setGetOfferRequests(`*numberRequested*`)`

- **numberRequested** - uma matriz de objetos `GetOfferRequest` que define o parâmetro para recuperar ofertas.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de uma chamada de método `GetOfferRequest 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

O método `setInteractiveChannel` define o nome do canal interativo usado pelo comando `startSession`.

`setInteractiveChannel(`*`interactiveChannel`*`)`

- **interactiveChannel**- uma sequência que contém o nome do canal interativo.

Importante: O *interactiveChannel* deve corresponder exatamente ao nome do canal interativo conforme definido no Campaign. Faz distinção entre maiúsculas e minúsculas.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método `executeBatch` que chama `startSession`.

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

setInteractionPoint

O método `setInteractionPoint` define o nome do ponto de interação usado pelos comandos `getOffers` e `postEvent`.

`setInteractionPoint(`*`interactionPoint`*`)`

- **interactionPoint**- uma sequência que contém o nome do ponto de interação.

Importante: O *interactionPoint* deve corresponder ao nome do ponto de interação conforme definido no canal interativo exatamente. Faz distinção entre maiúsculas e minúsculas.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método `executeBatch` que chama `getOffers`.

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

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

setMethodIdentifier

O método `setMethodIdentifier` define o tipo de comando contido no objeto de comando.

`setMethodIdentifier(`*`methodIdentifier`*`)`

- **methodIdentifier**- uma sequência que contém o tipo de comando.

  Os valores válidos são:

  - **COMMAND_ENDSESSION**- representa o método `endSession`.
  - **COMMAND_GETOFFERS**- representa o método `getOffers`.
  - **COMMAND_GETPROFILE**- representa o método `getProfile`.
  - **COMMAND_GETVERSION**- representa o método `getVersion`.
  - **COMMAND_POSTEVENT**- representa o método `postEvent`.
  - **COMMAND_SETAUDIENCE**- representa o método `setAudience`.
  - **COMMAND_SETDEBUG**- representa o método `setDebug`.
  - **COMMAND_STARTSESSION**- representa o método `startSession`.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método executeBatch que chama getVersion e endSession.

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

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

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

setNumberRequested

O método setNumberRequested define o número de ofertas solicitadas pelo comando getOffers.

setNumberRequested(*numberRequested*)

- **numberRequested**- um número inteiro que define o número de ofertas solicitadas pelo comando getOffers.

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método executeBatch que chama getOffers.

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

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

setRelyOnExistingSession

O método setRelyOnExistingSession define um booleano que define se o comando startSession usa uma sessão existente ou não.

setRelyOnExistingSession(*relyOnExistingSession*)

Se true, o ID de sessão para executeBatch deverá corresponder a um ID de sessão existente. Se false, você deverá fornecer um novo ID de sessão com o método executeBatch.

- **relyOnExistingSession**- um booleano true ou false).

Valor de retorno

None.

Exemplo

O exemplo a seguir é um extrato de um método executeBatch que chama startSession.

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

Sobre a interface NameValuePair

Muitos métodos na API do Interact retornam objetos NameValuePair ou requerem que objetos NameValuePair sejam transmitidos como argumentos. Ao transmitir como argumentos em um método, você deverá usar a implementação padrão NameValuePairImpl.

getName

O método getName retorna o componente do nome de um objeto NameValuePair.

```
getName()
```

Valor de retorno

O método getName retorna uma sequência.

Exemplo

O exemplo a seguir é um extrato de um método que processa o objeto de resposta para getProfile.

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

getValueAsDate

O método getValueAsDate retorna o valor de um objeto NameValuePair.

```
getValueAsDate()
```

Você deve usar getValueDataType antes de usar getValueAsDate para confirmar que está referenciando o tipo de dados correto.

Valor de retorno

O método getValueAsDate retorna uma data.

Exemplo

O exemplo a seguir é um extrato de um método que processa um NameValuePair e imprime o valor se ele for uma data.

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

getValueAsNumeric

O método getValueAsNumeric retorna o valor de um objeto NameValuePair.

```
getValueAsNumeric()
```

Você deve usar getValueDataType antes de usar getValueAsNumeric para confirmar que está referenciando o tipo de dados correto.

Valor de retorno

O método getValueAsNumeric retorna um duplo. Se, por exemplo, você estiver recuperando um valor originalmente armazenado em sua tabela de perfis como um Número inteiro, getValueAsNumeric retornará um duplo.

Exemplo

O exemplo a seguir é um extrato de um método que processa um NameValuePair e imprime o valor se ele for numérico.

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

getValueAsString

O método getValueAsString retorna o valor de um objeto NameValuePair.

```
getValueAsString()
```

Você deve usar getValueDataType antes de usar getValueAsString para confirmar que está referenciando o tipo de dados correto.

Valor de retorno

O método getValueAsString retorna uma sequência. Se, por exemplo, você estiver recuperando um valor originalmente armazenado em sua tabela de perfis como um caractere, varchar ou char[10], getValueAsString retornará uma sequência.

Exemplo

O exemplo a seguir é um extrato de um método que processa um NameValuePair e imprime o valor se ele for uma sequência.

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

getValueDataType

O método getValueDataType retorna m tipo de dados de um objeto NameValuePair.

```
getValueDataType()
```

Você deve usar getValueDataType antes de usar getValueAsDate, getValueAsNumeric ou getValueAsString para confirmar que está referenciando o tipo de dados correto.

Valor de retorno

O método `getValueDataType` retorna uma sequência que indica se o NameValuePair contém dados, número ou sequência.

Os valores válidos são:

- **DATA_TYPE_DATETIME**- uma data que contém um valor de data e hora.
- **DATA_TYPE_NUMERIC**- um duplo que contém um valor de número.
- **DATA_TYPE_STRING**- uma sequência que contém um valor de texto.

Exemplo

O exemplo a seguir é um extrato de um método que processa o objeto de resposta de um método `getProfile`.

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

setName

O método `setName` define o componente do nome de um objeto NameValuePair.

`setName(`*`name`*`)`

- **name**- uma sequência que contém o componente do nome de um objeto NameValuePair.

Valor de retorno

None.

Exemplo

O exemplo a seguir mostra como definir o componente do nome de um NameValuePair.

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

setValueAsDate

O método `setValueAsDate` define o valor de um objeto NameValuePair.

`setValueAsDate(`*`valueAsDate`*`)`

- **valueAsDate**- uma data que contém o valor de data e hora do objeto NameValuePair.

Valor de retorno

None.

Exemplo

O exemplo a seguir mostra como definir o componente de valor de um NameValuePair se o valor for uma data.

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

setValueAsNumeric

O método `setValueAsNumeric` define o valor de um objeto NameValuePair.

`setValueAsNumeric(`*`valueAsNumeric`*`)`

- **valueAsNumeric**- um duplo que contém o valor numérico do objeto NameValuePair.

Valor de retorno

None.

Exemplo

O exemplo a seguir mostra como definir o componente de valor de um NameValuePair se o valor for um numérico.

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

setValueAsString

O método `setValueAsString` define o valor de um objeto NameValuePair.

`setValueAsString(`*`valueAsString`*`)`

- **valueAsString**- uma sequência que contém o valor de um objeto NameValuePair

Valor de retorno

None.

Exemplo

O exemplo a seguir mostra como definir o componente de valor de um NameValuePair se o valor for um numérico.

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

setValueDataType

O método `setValueDataType` define o tipo de dados de um objeto NameValuePair.

`getValueDataType(`*`valueDataType`*`)`

Os valores válidos são:

- **DATA_TYPE_DATETIME**- uma data que contém um valor de data e hora.
- **DATA_TYPE_NUMERIC**- um duplo que contém um valor de número.
- **DATA_TYPE_STRING**- uma sequência que contém um valor de texto.

Valor de retorno

None.

Exemplo

Os exemplos a seguir mostram como configurar o tipo de dados do valor de um NameValuePair.

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

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

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

Sobre a classe Oferta

A classe `Oferta` contém métodos que definem um objeto Oferta. Esse objeto de oferta contém muitas das mesmas propriedades de uma oferta no Campaign.

O objeto de oferta contém os seguintes atributos:

- **AdditionalAttributes** - NameValuePairs contendo todos os atributos de oferta customizada que foram definidos no Campaign.
- **Descrição** - A descrição da oferta.
- **EffectiveDate** - A data efetiva da oferta.
- **ExpirationDate** - A data de expiração da oferta.
- **OfferCode** - O código de oferta da oferta.
- **OfferName** - O nome da oferta.
- **TreatmentCode** - O código de tratamento da oferta.
- **Pontuação** - A pontuação de marketing da oferta ou a pontuação definida pela `ScoreOverrideTable` se a propriedade `enableScoreOverrideLookup` for true.

getAdditionalAttributes

O método `getAdditionalAttributes` retorna os atributos de oferta customizada definidos no Campaign.

```
getAdditionalAttributes()
```

Valor de retorno

O método `getAdditionalAttributes` retorna uma matriz de objetos NameValuePair.

Exemplo

O exemplo a seguir classifica através de todos os atributos adicionais, verificando a data efetiva e a data de expiração e imprimindo os outros atributos.

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

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

getDescription

O método getDescription retorna a descrição da oferta definida em Campaign.

```
getDescription()
```

Valor de retorno

O método getDescription retorna uma sequência.

Exemplo

O exemplo a seguir imprime a descrição de uma oferta.

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

getOfferCode

O método getOfferCode retorna o código de oferta da oferta conforme definido no Campaign.

```
getOfferCode()
```

Valor de retorno

O método getOfferCode retorna uma matriz de sequências que contém o código de oferta da oferta.

Exemplo

O exemplo a seguir imprime o código de oferta de uma oferta.

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

getOfferName

O método getOfferName retorna o nome da oferta conforme definido no Campaign.

```
getOfferName()
```

Valor de retorno

O método getOfferName retorna a sequência.

Exemplo

O exemplo a seguir imprime o nome de uma oferta.

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

getScore

O método getScore retorna uma pontuação, que é baseada nas ofertas configuradas.

```
getScore()
```

O método getScore retorna um dos seguintes:

- Se você não ativou a tabela de ofertas padrão, a tabela de substituição de pontuação ou o aprendizado integrado, este método retorna a pontuação de marketing da oferta conforme definido na guia Estratégia de interação.
- Se você ativou as ofertas padrão ou tabela de substituição de pontuação e não ativou o aprendizado integrado, este método retornará a pontuação da oferta conforme definido pela ordem de precedência entre a tabela de ofertas padrão, a pontuação do comerciante e a tabela de substituição de pontuação.
- Se você ativou o aprendizado integrado, este método retornará a pontuação final que o aprendizado integrado usou para solicitar ofertas.

Valor de retorno

O método getScore retorna um número inteiro que representa a pontuação da oferta.

Exemplo

O exemplo a seguir imprime a pontuação de uma oferta.

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

getTreatmentCode

O método getTreatmentCode retorna o código de tratamento da oferta conforme definido em Campaign.

```
getTreatmentCode()
```

Como o Campaign usa o código de tratamento para identificar a instância da oferta entregue, esse código deve ser retornado como um parâmetro de evento ao usar o método postEvent para registrar um evento de contato, aceitação ou rejeição da oferta. Se você estiver registrando uma aceitação ou rejeição da oferta, você deve configurar o valor do nome do NameValuePair que representa o código de tratamento para UACIOfferTrackingCode.

Valor de retorno

O método getTreatmentCode retorna uma sequência.

Exemplo

O exemplo a seguir imprime o código de tratamento de uma oferta.

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

Sobre a classe OfferList

A classe OfferList contém métodos que definem os resultados do método getOffers.

O objeto OfferList contém os seguintes atributos:

- **DefaultString** - A sequência padrão definida para o ponto de interação no canal interativo.
- **RecommendedOffers** - Uma matriz dos objetos Oferta solicitados pelo método getOffers.

A classe OfferList funciona com listas de ofertas. Essa classe não está relacionada às listas de ofertas do Campaign.

getDefaultString

O método getDefaultString retorna a sequência padrão para o ponto de interação conforme definido no Campaign.

```
getDefaultString()
```

Se o objeto RecommendedOffers estiver vazio, você deve configurar seu ponto de contato para apresentar esta sequência para assegurar que algum conteúdo seja apresentado. O Interact preenche o objeto DefaultString apenas se o objeto RecommendedOffers estiver vazio.

Valor de retorno

O método getDefaultString retorna uma sequência.

Exemplo

O exemplo a seguir obtém a sequência padrão se o objeto offerList não contiver ofertas.

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

getRecommendedOffers

O método `getRecommendedOffers` retorna uma matriz de objetos de Oferta solicitada pelo método `getOffers`.

```
getRecommendedOffers()
```

Se a resposta para `getRecommendedOffer` for vazia, o ponto de contato deverá apresentar o resultado de `getDefaultString`.

Valor de retorno

O método `getRecommendedOffers` retorna um objeto Oferta.

Exemplo

O exemplo a seguir processa o objeto OfferList e imprime o nome da oferta para todas as ofertas recomendadas.

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

Sobre a classe Resposta

A classe `Resposta` contém métodos que definem os resultados de quaisquer métodos de classe InteractAPI.

O objeto Resposta contém os seguintes atributos:

- **AdvisoryMessages** - uma matriz de mensagens de recomendação. Este atributo somente será preenchido se houver avisos ou erros quando o método for executado.
- **ApiVersion** - uma sequência contendo a versão da API. Este atributo é preenchido pelo método `getVersion`.
- **OfferList** - o objeto OfferList que contém as ofertas solicitadas pelo método `getOffers`.

- **ProfileRecord** - uma matriz de NameValuePairs que contém dados do perfil. Esse atributo é preenchido pelo método `getProfile`.
- **SessionID** - uma sequência que define o ID de sessão. É retornado por todos os métodos de classe `InteractAPI`.
- **StatusCode** - um número indicando se o método foi executado sem erro, com um aviso ou com erros. É retornado por todos os métodos de classe `InteractAPI`.

getAdvisoryMessages

O método `getAdvisoryMessages` retorna uma matriz de Mensagens de Recomendação do objeto Resposta.

```
getAdvisoryMessages()
```

Valor de retorno

O método `getAdvisoryMessages` retorna uma matriz de objetos de Mensagem de Recomendação.

Exemplo

O exemplo a seguir obtém os objetos do AdvisoryMessage de um objeto Resposta e itera através deles, imprimindo as mensagens.

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

getApiVersion

O método `getApiVersion` retorna a versão de API de um objeto Resposta.

```
getApiVersion()
```

O método `getVersion` preenche o atributo ApiVersion de um objeto Resposta.

Valor de retorno

O objeto Resposta retorna uma sequência.

Exemplo

O exemplo a seguir é um extrato de um método que processa o objeto de resposta para `getVersion`.

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

getOfferList

O método `getOfferList` retorna o objeto OfferList de um objeto Resposta.

```
getOfferList()
```

O método `getOffers` preenche o objeto OfferList de um objeto Resposta.

Valor de retorno

O objeto Resposta retorna um objeto OfferList.

Exemplo

O exemplo a seguir é um extrato de um método que processa o objeto de resposta para `getOffers`.

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

getAllOfferLists

O método `getAllOfferLists` retorna uma matriz de todas as OfferLists de um objeto Resposta.

```
getAllOfferLists()
```

Isto é usado pelo método `getOffersForMultipleInteractionPoints` que preenche o objeto da matriz OfferList de um objeto Resposta.

Valor de retorno

O objeto Resposta retorna uma matriz OfferList.

Exemplo

O exemplo a seguir é um extrato de um método que processa o objeto de resposta para `getOffers`.

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

getProfileRecord

O método `getProfileRecord` retorna os registros de perfil para a sessão atual como uma matriz de objetos NameValuePair. Esses registros de perfil também incluem quaisquer `eventParameters` incluídos anteriormente na sessão de tempo de execução.

```
getProfileRecord()
```

O método `getProfile` preenche o registro de perfil NameValuePair de um objeto Resposta.

Valor de retorno

O objeto Resposta retorna uma matriz de objetos NameValuePair.

Exemplo

O exemplo a seguir é um extrato de um método que processa o objeto de resposta para `getOffers`.

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

getSessionID

O método `getSessionID` retorna o ID de sessão.

```
getSessionID()
```

Valor de retorno

O método `getSessionID` retorna uma sequência.

Exemplo

O exemplo a seguir mostra uma mensagem que pode ser exibida no fim ou começo de sua manipulação de erros para indicar a qual sessão pertencem os erros.

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

getStatusCode

O método `getStatusCode` retorna o código de status de um objeto Resposta.

```
getStatusCode()
```

Valor de retorno

O objeto Resposta retorna um número inteiro.

- 0 - STATUS_SUCCESS - O método chamado concluído sem erros. Pode ou não haver Mensagens de Recomendação.
- 1 - STATUS_WARNING - O método chamado concluído com pelo menos uma mensagem de aviso (mas sem erros). Consulte Mensagens de Recomendação para obter mais detalhes.
- 2 - STATUS_ERROR - O método chamado não foi concluído com sucesso e tem pelo menos uma mensagem de erro. Consulte Mensagens de Recomendação para obter mais detalhes.

Exemplo

A seguir, está um exemplo de como é possível usar `getStatusCode` na manipulação de erros.

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

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

Capítulo 8. Classes e métodos para a API JavaScript do IBM Interact

As seções a seguir listam requisitos e outros detalhes que você deve conhecer antes de trabalhar com a API JavaScript do Interact.

A API do Interact suporta um tipo de javascript para permitir comunicação do cliente do usuário final (navegador) com o servidor.

Nota: Esta seção assume que você está familiarizado com uma API baseada em JavaScript.

Nota: Várias ocorrências de qualquer parâmetro em uma única chamada de API não são suportadas.

Pré-requisitos JavaScript

Antes de usar a API JavaScript do Interact em um website, você deve incluir o arquivo `interactapi.js` em suas páginas da web.

Trabalhando com dados da sessão

Quando você inicia uma sessão com o método `startSession`, os dados da sessão são carregados na memória. Durante a sessão, é possível ler e gravar os dados da sessão (que é um superconjunto dos dados do perfil estático).

A sessão contém os seguintes dados:

- Dados do perfil estático
- Designações de segmento
- Dados em tempo real
- Recomendações de oferta

Todos os dados da sessão ficam disponíveis até que você chame o método `endSession` ou que o tempo `sessionTimeout` seja decorrido. Quando a sessão termina, todos os dados não salvos explicitamente no histórico de contatos ou respostas ou alguma outra tabela de banco de dados são perdidos.

Os dados são armazenados como um conjunto de pares nome-valor. Se os dados forem lidos a partir de uma tabela de banco de dados, o nome será a coluna da tabela.

É possível criar esses pares nome-valor ao longo do trabalho com a API do Interact. Não é necessário declarar todos os pares nome-valor em uma área global. Se você configurar novos parâmetros de evento como pares nome-valor, o ambiente de tempo de execução incluirá os pares nome-valor nos dados da sessão. Por exemplo, se você usar os parâmetros de evento com o método `postEvent`, o ambiente de tempo de execução incluirá os parâmetros de evento nos dados da sessão, mesmo que os parâmetros de evento não estejam disponíveis nos dados do perfil. Esses dados somente existem nos dados da sessão.

É possível sobrescrever os dados da sessão a qualquer momento. Por exemplo, se parte do perfil do cliente incluir `creditScore`, será possível transmitir um parâmetro de evento usando o tipo customizado `NameValuePair`. Na classe `NameValuePair`, é possível usar os métodos `setName` e `setValueAsNumeric` para alterar o valor. O nome precisa corresponder. Nos dados da sessão, o nome não faz distinção entre maiúsculas e minúsculas. Portanto, o nome `creditscore` ou `CrEdItScOrE` sobrescreveria `creditScore`.

Somente os últimos dados gravados nos dados da sessão são mantidos. Por exemplo, `startSession` carrega os dados do perfil para o valor de `lastOffer`. Um método `postEvent` substitui o `lastOffer`. Em seguida, um segundo método `postEvent` substitui `lastOffer`. O ambiente de tempo de execução somente mantém os dados gravados pelo segundo método `postEvent` nos dados da sessão.

Quando a sessão terminar, os dados serão perdidos, a menos que tenham sido feitas considerações especiais, como usar um processo de Captura instantânea no fluxograma interativo para gravar os dados em uma tabela de banco de dados. Se você estiver planejando usar processos de Captura instantânea, lembre-se de que os nomes precisam corresponder às limitações do banco de dados. Por exemplo, se somente forem permitidos 256 caracteres para o nome de uma coluna, o nome do par nome-valor não deverá exceder 256 caracteres.

Trabalhando com o parâmetro de retorno de chamada

A função de retorno de chamada é um parâmetro adicional de cada método da API JavaScript do Interact.

O processo do navegador principal é um loop de evento encadeado único. Executar uma operação de longa execução com um loop de evento encadeado único, bloqueia o processo. Isto interrompe o processamento de outros eventos enquanto o processo aguarda que sua operação seja concluída. Para evitar bloqueios em operações de longa execução, o XMLHttpRequest fornece uma interface assíncrona. Você passa um retorno de chamada para ser executado após a operação ser concluída, e enquanto ele processa, ele dá o controle de volta ao loop do evento principal em vez de bloquear o processo.

Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

Por exemplo, se você desejava exibir ofertas em sua página da web, poderia usar o método `getOffers` e usar o retorno de chamada para exibição na página. A página da web se comporta normalmente e não aguarda que o Interact retorne as ofertas. Em vez disso, quando o Interact retorna as ofertas, a resposta é enviada de volta no parâmetro de retorno de chamada. É possível analisar os dados de retorno de chamada e mostrar ofertas na página.

É possível usar um retorno de chamada genérico para todas as funções ou também é possível usar retornos de chamada específicos para funções específicas.

É possível usar `var callback = InteractAPI.Callback.create(onSuccess, onError);` para criar uma função de retorno de chamada genérica.

É possível usar a função a seguir para criar uma função de retorno de chamada específica para o método getOffers.

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

Sobre a classe InteractAPI

A classe InteractAPI contém os métodos usados para integrar o ponto de contato ao servidor de runtime. Todos os outros métodos e classes e na API do Interact suportam os métodos nessa classe.

Você deve compilar a implementação com relação ao `interact_client.jar` localizado no diretório `lib` da instalação do ambiente de tempo de execução do Interact.

startSession

O método `startSession` cria e define uma sessão de tempo de execução.

```
function callStartSession(commandsToExecute, callback) {

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

        InteractAPI.startSession(ssId, icName,
                                    getNameValuePairs(audId), audLevel,
                                    getNameValuePairs(params), relyOldSs,
                                    debug, callback) ;

}
```

`startSession` pode acionar até cinco ações:

- crie uma sessão de tempo de execução.
- carregue dados do perfil do visitante para o nível de público atual na sessão de tempo de execução, incluindo quaisquer tabelas de dimensões marcadas para carregamento no mapeamento de tabela definido para o canal interativo.
- segmentação de acionador, executando todos os fluxogramas interativos para o nível de público atual.
- carregue os dados de supressão de oferta na sessão, se a propriedade `enableOfferSuppressionLookup` estiver configurada como true.
- carregue os dados de substituição de pontuação na sessão, se a propriedade `enableScoreOverrideLookup` estiver configurada como true.

O método `startSession` requer os parâmetros a seguir:

- **sessionID**- uma sequência que identifica o ID de sessão. Você deve definir o ID de sessão. Por exemplo, você poderia usar uma combinação do ID do cliente e registro de data e hora.

  Para definir o que faz uma sessão de tempo de execução, um ID de sessão precisa ser especificado. Este valor é gerenciado pelo cliente. Todas as chamadas de método para o mesmo ID de sessão têm de ser sincronizadas pelo cliente. O comportamento para chamadas API simultâneas com o mesmo ID de sessão é indefinido.
- **relyOnExistingSession** - um booleano que define se esta sessão usa uma sessão nova ou existente. Os valores válidos são `true` ou `false`. Se `true`, você deve fornecer um ID de sessão existente com o método `startSession`. Se `false`, você deve fornecer um novo ID de sessão.

Se você configurar relyOnExistingSession para true e existir uma sessão, o ambiente de tempo de execução usará os dados da sessão existentes e não recarregará nenhum dado ou segmentação de acionador. Se a sessão não existir, o ambiente de tempo de execução criará uma nova sessão, incluindo o carregamento de dados e o acionamento da segmentação. Configurar relyOnExistingSession para true e usá-lo com todas as chamadas startSession é útil se seu ponto de contato tiver uma sessão de comprimento mais longo que a sessão de tempo de execução. Por exemplo, uma sessão de website está ativa há 2 horas, mas a sessão de tempo de execução só está ativa há 20 minutos.

Se você chamar startSession duas vezes com o mesmo ID de sessão, todos os dados da sessão da primeira chamada startSession serão perdidos se relyOnExistingSession for false.

- **debug** - um booleano que permite ou não informações sobre depuração. Os valores válidos são true ou false. Se true, Interact registrará informações sobre depuração nos logs do servidor de runtime. O sinalizador de depuração é configurado para cada sessão individualmente. Portanto, é possível rastrear dados de depuração para uma sessão individual.
- **interactiveChannel**- uma sequência que define o nome do canal interativo a que esta sessão se refere. Este nome deve corresponder ao nome do canal interativo definido em Campaign exatamente.
- **audienceID** - uma matriz de objetos NameValuePairImpl em que os nomes devem corresponder aos nomes de coluna física de qualquer tabela que contenha o ID de público.
- **audienceLevel** - uma sequência que define o nível de público.
- **parameters** - objetos NameValuePairImpl que identificam quaisquer parâmetros que precisam ser passados com startSession. Estes valores são armazenados nos dados da sessão e podem ser usados para segmentação.

  Se você tiver vários fluxogramas interativos para o mesmo nível de público, deverá incluir um superconjunto de todas as colunas em todas as tabelas. Se você configurar o tempo de execução para carregar a tabela de perfis e a tabela de perfis contiver todas as colunas que você precisa, não será necessário passar parâmetros, a menos que queira sobrescrever os dados na tabela de perfis. Se sua tabela de perfis contiver um subconjunto das colunas necessárias, você deverá incluir as colunas ausentes como parâmetros.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama onSuccess. Se o método falhou, a função de retorno de chamada chama onError.

Se audienceID ou audienceLevel forem inválidos e relyOnExistingSession for false, a chamada startSession falhará. Se interactiveChannel for inválido, startSession falhará, independentemente de relyOnExistingSession ser true ou false.

Se relyOnExistingSession for true e você fizer uma segunda chamada startSession usando o mesmo sessionID, mas a primeira sessão tiver expirado, Interact criará uma nova sessão.

Se relyOnExistingSession for true e você fizer uma segunda chamada startSession usando o mesmo sessionID, mas um audienceID ou audienceLevel diferente, o servidor de runtime alterará o público para a sessão existente.

Se relyOnExistingSession for true e você fizer uma segunda chamada startSession usando o mesmo sessionID, mas um interactiveChannel diferente, o servidor de runtime criará uma nova sessão.

Valor de retorno

O servidor de runtime responderá a `startSession` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages (se StatusCode não for igual a 0)
- ApiVersion
- SessionID
- StatusCode

Deduplicação de oferta entre atributos de oferta

Usando a interface de programação de aplicativos (API) do Interact, duas chamadas de API entregam ofertas: `getOffers` e `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` pode impedir o retorno de ofertas duplicadas no nível *OfferID*, mas não pode deduplicar ofertas entre categoria de oferta. Portanto, por exemplo, para que o Interact retorne apenas uma oferta de cada categoria de oferta, uma solução alternativa era necessária anteriormente. Com a introdução de dois parâmetros para a chamada API `startSession`, a deduplicação de oferta entre atributos de oferta, como categoria, agora é possível.

Esta lista resume os parâmetros que foram incluídos na chamada API `startSession`. Para obter mais informações sobre esses parâmetros ou qualquer aspecto da API do Interact, consulte o *IBM Interact Administrator's Guide* ou os arquivos Javadoc incluídos com sua instalação do Interact no `<Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Para criar uma chamada API `startSession` com deduplicação de oferta, para que as chamadas subsequentes `getOffer` retornem apenas uma oferta de cada categoria, você deve incluir o parâmetro `UACIOfferDedupeAttribute` como parte da chamada API. É possível especificar um parâmetro no formato `name,value,type`, conforme mostrado aqui:

  `UACIOfferDedupeAttribute,<attributeName>,string`

  Neste exemplo, você substituiria *<attributeName>* pelo nome do atributo de oferta que deseja usar como critério para a deduplicação, como Categoria.

  Nota: O Interact examina as ofertas que têm o mesmo valor de atributo que especificar (como Categoria) e deduplica para remover todos menos a oferta que possui a pontuação mais alta. Se as ofertas que possuem o atributo duplicado também possuírem pontuações idênticas, o Interact retornará uma seleção aleatória entre as ofertas correspondentes.

- `UACINoAttributeDedupeIfFewerOffer`. Ao incluir o `UACIOfferDedupeAttribute` na chamada `startSession`, também é possível configurar esse parâmetro `UACINoAttributeDedupeIfFewerOffer` para especificar o comportamento em casos em que a lista de ofertas após a deduplicação não contenha mais ofertas suficientes para satisfazer a solicitação original.

  Por exemplo, se você configurar `UACIOfferDedupeAttribute` para usar a categoria de oferta para deduplicar ofertas e sua chamada getOffers subsequente solicitar que oito ofertas sejam retornadas, a deduplicação poderá resultar em menos que oito ofertas elegíveis. Nesse caso, configurar o parâmetro `UACINoAttributeDedupeIfFewerOffer` como true resultaria na inclusão de alguns dos duplicados na lista de elegíveis para satisfazer o número solicitado de

ofertas. Neste exemplo, se você configurar o parâmetro como false, o número de ofertas retornadas seria menor que o número solicitado.

`UACINoAttributeDedupeIfFewerOffer` é configurado como true por padrão.

Por exemplo, suponha que você especificou um parâmetro `startSession` que o critério de deduplicação é a oferta Categoria, conforme mostrado aqui:

```
UACIOfferDedupeAttribute,Category,string;

UACINoAttributeDedupeIfFewerOffer,1,string
```

Por padrão, o `UACIOfferDedupeAttribute` não deduplicará ofertas se menos do que o número solicitado for retornado. No entanto, para assegurar que a deduplicação aconteça quando o número de ofertas retornadas é menor do que o número de ofertas solicitadas, o parâmetro `UACINoAttributeDedupeIfFewerOffer` deve ser fornecido e deve ser configurado como 1.

Esses parâmetros juntos fazem com que o Interact deduplique ofertas com base no atributo de oferta "Categoria" e retorne apenas as ofertas deduplicadas mesmo que o número resultante de ofertas seja menor que o solicitado (`UACINoAttributeDedupeIfFewerOffer` é false).

Ao emitir uma chamada API `getOffers`, o conjunto original de ofertas elegíveis pode incluir essas ofertas:

- Category=Electronics: Oferta A1 com uma pontuação de 100 e Oferta A2 com uma pontuação de 50.
- Category=Smartphones: Oferta B1 com uma pontuação de 100, Oferta B2 com uma pontuação de 80 e Oferta B3 com uma pontuação de 50.
- Category=MP3Players: Oferta C1 com uma pontuação de 100, Oferta C2 com uma pontuação de 50.

Neste caso, havia duas ofertas duplicadas que correspondem à primeira categoria, três ofertas duplicadas que correspondem à segunda categoria e duas ofertas duplicadas que correspondem à terceira categoria. As ofertas que são retornadas seriam as ofertas de maior pontuação de cada categoria, que são Oferta A1, Oferta B1 e Oferta C1.

Se a chamada API `getOffers` solicitou seis ofertas, este exemplo configurou `UACINoAttributeDedupeIfFewerOffer` como false, portanto, apenas três ofertas seriam retornadas.

Se a chamada API `getOffers` solicitou seis ofertas e este exemplo omitiu o parâmetro `UACINoAttributeDedupeIfFewerOffer` ou especificamente o configurou como true, algumas das ofertas duplicadas seriam incluídas no resultado para atender ao número solicitado.

postEvent

O método `postEvent` permite executar qualquer evento definido no canal interativo.

```
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**: uma sequência que identifica o ID de sessão.
- **eventName**: uma sequência que identifica o nome do evento.

  Nota: O nome do evento deve corresponder ao nome do evento conforme definido no canal interativo. Este nome não faz distinção entre maiúsculas e minúsculas.
- **eventParameters**. Objetos `NameValuePairImpl` que identificam quaisquer parâmetros que precisam ser passados com o evento. Esses valores serão armazenados nos dados da sessão.

  Se este evento acionar a ressegmentação, você deve assegurar que todos os dados necessários aos fluxogramas interativos estejam disponíveis nos dados da sessão. Se quaisquer um desses valores não tiver sido preenchido por ações anteriores (por exemplo, de `startSession` ou `setAudience` ou carregando a tabela de perfis), você deve incluir um eventParameter para cada valor ausente. Por exemplo, se você tiver configurado todas as tabelas de perfis para carregar na memória, você deve incluir um NameValuePair para dados temporais necessários para os fluxogramas interativos.

  Se você estiver usando mais de um nível de público, provavelmente terá diferentes conjuntos de eventParameters para cada nível de público. Você deve incluir alguma lógica para assegurar que está selecionando o conjunto correto de parâmetros para o nível de público.

  Importante: Se este evento for registrado no histórico de respostas, você deverá passar o código de tratamento para a oferta. Você deve definir o nome para NameValuePair como "`UACIOfferTrackingCode`".

  É possível apenas passar um código de tratamento por evento. Se você não passar o código de tratamento para um contato de oferta, o Interact registrará um contato de oferta para cada oferta na última lista de ofertas recomendadas. Se você não passar o código de tratamento para uma resposta, o Interact retornará um erro.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.
- Há vários outros parâmetros reservados usados com `postEvent` e outros métodos e são discutidos posteriormente nesta seção.

Qualquer solicitação de ressegmentação ou gravação no histórico de respostas ou contato não aguardará uma resposta.

A ressegmentação não limpa resultados da segmentação anterior para o nível de audiência atual. É possível usar o parâmetro `UACIExecuteFlowchartByName` para definir fluxogramas específicos para execução. O método `getOffers` aguarda a ressegmentação ser concluída antes de executar. Portanto, se você chamar um método `postEvent`, que aciona a ressegmentação imediatamente antes de uma chamada `getOffers`, pode haver um atraso.

Valor de retorno

O servidor de runtime responde a `postEvent` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getOffers

O método `getOffers` permite solicitar ofertas do servidor de runtime.

```
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);

}
```

- **ID de sessão** - uma sequência identificando a sessão atual.
- **Ponto de interação**-uma sequência identificando o nome do ponto de interação ao qual este método se referencia.

  Nota: Esse nome deve corresponder exatamente ao nome do ponto de interação definido no canal interativo.
- **nofRequested**-um número inteiro identificando o número de ofertas solicitadas.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

O método `getOffers` aguarda o número de milissegundos definidos na propriedade `segmentationMaxWaitTimeInMS` para que toda a ressegmentação seja concluída antes da execução. Portanto, se você chamar um método `postEvent` que aciona uma ressegmentação ou um método `setAudience` imediatamente antes de uma chamada `getOffers`, pode haver um atraso.

Valor de retorno

O servidor de runtime responde a `getOffers` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getOffersForMultipleInteractionPoints

O método `getOffersForMultipleInteractionPoints` permite solicitar ofertas do servidor de runtime para diversos IPs com deduplicação.

```
function callGetOffersForMultipleInteractionPoints(commandsToExecute, callback) {

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

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

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

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

        //build get offer requests
        var getOffReqs = [];
        for(var i = 0; i < parts.length; i += 1) {
            var getofReqObj = parseGetOfferReq(parts[i]);
            if (getofReqObj) {
                getOffReqs.push(getofReqObj);

        InteractAPI.getOffersForMultipleInteractionPoints
        (ssId, getOffReqs, callback);

}
```

- **ID de sessão** - uma sequência identificando a sessão atual.
- **requestDetailsStr** - uma sequência fornecendo uma matriz de objetos `GetOfferRequest`.

  Cada objeto `GetOfferRequest` especifica:

  – **ipName** - O nome do ponto de interação (IP) para o qual o objeto está solicitando ofertas

  – **numberRequested** - O número de ofertas exclusivas que ele precisa para o IP especificado

  – **offerAttributes** - Requisitos nos atributos das ofertas entregues que usam uma instância de `OfferAttributeRequirements`

  – **duplicationPolicy** - ID de política de duplicação para as ofertas que serão entregues

    As políticas de duplicação determinam se as ofertas duplicadas serão retornadas em diferentes pontos de interação em uma única chamada de método. (*Em* um ponto de interação individual, as ofertas duplicadas nuunca são retornadas.) Atualmente, duas políticas de duplicação são suportadas.

    - NO_DUPLICATION (valor de ID = 1). Nenhuma das ofertas que foram incluídas nas instância `GetOfferRequest` anteriores será incluída nesta instância `GetOfferRequest` (isto é, o Interact aplicará a deduplicação).
    - ALLOW_DUPLICATION (valor de ID = 2). Qualquer uma das ofertas que atendem aos requisitos especificados nesta instância `GetOfferRequest` será incluída. As ofertas que foram incluídas nas instâncias `GetOfferRequest` anteriores não serão reconciliadas.

  – **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

  A ordem das solicitações no parâmetro de matriz também é a ordem de prioridade em que as ofertas estão sendo entregues.

  Por exemplo, suponha que os IPs na solicitação sejam IP1, depois IP2, que nenhuma oferta duplicada seja permitida (um ID de política de duplicação = 1) e cada um está representando duas ofertas. Se o Interact localizar ofertas A, B e C para IP1 e ofertas A e D para IP2, a resposta conterá as ofertas A e B para IP1 e apenas a oferta D para IP2.

Observe também que quando o ID de política de duplicação for 1, as ofertas que foram entregues por um IP com prioridade mais alta não serão entregues por esse IP.

O método `getOffersForMultipleInteractionPoints` aguarda o número de milissegundos definidos na propriedade `segmentationMaxWaitTimeInMS` até que toda a ressegmentação seja concluída antes da execução. Portanto, se você chamar um método `postEvent` que aciona uma ressegmentação ou um método `setAudience` imediatamente antes de uma chamada `getOffers`, pode haver um atraso.

Valor de retorno

O servidor de runtime responde a `getOffersForMultipleInteractionPoints` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- Matriz de OfferList
- Profile
- SessionID
- StatusCode

setAudience

O método `setAudience` permite configurar o ID de público e o nível para um visitante.

```
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** - uma sequência que identifica o ID de sessão.
- **audienceID** - uma matriz de objetos `NameValuePairImpl` que define o ID de público.
- **audienceLevel** - uma sequência que define o nível de público.
- **parameters** - objetos `NameValuePairImpl` que identificam quaisquer parâmetros que precisam ser transmitidos com `setAudience`. Estes valores são armazenados nos dados da sessão e podem ser usados para segmentação.

  Você deve ter um valor para cada coluna em seu perfil. Este é um superconjunto de todas as colunas em todas as tabelas definidas para o canal interativo e quaisquer dados em tempo real. Se você já tiver preenchido todos os dados da sessão com `startSession` ou `postEvent`, não será necessário enviar novos parâmetros.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

O método setAudience aciona uma ressegmentação. O método getOffers aguarda a ressegmentação ser concluída antes de executar. Portanto, se você chamar um método setAudience imediatamente antes de uma chamada getOffers, pode haver um atraso.

O método setAudience também carrega os dados do perfil para o ID de público. É possível usar o método setAudience para forçar um recarregamento dos mesmos dados do perfil carregados pelo método startSession.

O método setAudience recarrega a lista de desbloqueio e a tabela de lista de bloqueio em uma sessão existente. É possível usar o método setAudience com os parâmetros UACIPurgePriorWhiteListOnLoad e UACIPurgePriorBlackListOnLoad para recarregar a tabela de lista de desbloqueio e a tabela de lista de bloqueio em uma sessão existente.

Por padrão, quando o método setAudience é chamado, todo o conteúdo da lista de bloqueio é removido. É possível configurar os parâmetros UACIPurgePriorWhiteListOnLoad e UACIPurgePriorBlackListOnLoad na chamada setAudience conforme a seguir:

- Se você configurar UACIPurgePriorBlackListOnLoad= 0, todo o conteúdo da tabela de lista de desbloqueio será preservado.
- Se você configurar UACIPurgePriorWhiteListOnLoad= 1 o conteúdo da tabela será removido e o conteúdo da lista de desbloqueio ou da lista de bloqueio para o ID de público será carregado a partir do banco de dados. Depois de concluído, a ressegmentação iniciará.

Valor de retorno

O servidor de runtime responde a setAudience com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getProfile

O método getProfile permite recuperar o perfil e informações temporais sobre o visitante que está visitando o ponto de contato.

```
function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        InteractAPI.getProfile(ssId, callback);

}
```

- **ID de sessão**-uma sequência identificando o ID de sessão.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama onSuccess. Se o método falhou, a função de retorno de chamada chama onError.

Valor de retorno

O servidor de runtime responde a `getProfile` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- OfferList
- ProfileRecord
- SessionID
- StatusCode

endSession

O método `endSession` marca o fim da sessão de tempo de execução. Quando o servidor de runtime receber este método, o servidor de runtime registrará no histórico, limpará a memória e assim por diante.

```
function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        InteractAPI.endSession(ssId, callback);

}
```

- **ID de sessão** - Sequência exclusiva identificando a sessão.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

Se o método endSession não for chamado, as sessões de tempo de execução atingirão o tempo limite. O período de tempo limite é configurável com a propriedade `sessionTimeout`.

Valor de retorno

O servidor de runtime responde ao método `endSession` com o objeto `Response` com os atributos a seguir preenchidos:

- SessionID
- ApiVersion
- OfferList
- Profile
- StatusCode
- AdvisoryMessages

setDebug

O método `setDebug` permite configurar o nível de detalhamento da criação de log para todos os caminhos de código da sessão.

```
function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        InteractAPI.setDebug(ssId, isDebug, callback);

}
```

- **sessionID**- uma sequência que identifica o ID de sessão.
- **debug** - um booleano que permite ou não informações sobre depuração. Os valores válidos são `true` ou `false`. Se true, o Interact registrará informações sobre depuração no log do servidor de runtimeg.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

Valor de retorno

O servidor de runtime responde ao `setDebug` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getVersion

O método `getVersion` retorna a versão da implementação atual do servidor de runtime do Interact.

```
function callGetVersion(commandsToExecute, callback) {

        InteractAPI.getVersion(callback);


}
```

A melhor prática é usar este método ao inicializar o ponto de contato com a API do Interact.

- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

Valor de retorno

O servidor de runtime responde ao `getVersion` com um objeto Resposta com os atributos a seguir preenchidos:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

executeBatch

O método `executeBatch` permite executar vários métodos com uma única solicitação para o servidor de runtime.

```
function callExecuteBatch(commandsToExecute, callback) {

        if (!commandsToExecute)
```

```
            return ;

        InteractAPI.executeBatch(commandsToExecute.ssid,
        commandsToExecute.commands, callback);
}
```

- **ID de sessão**-Uma sequência identificando o ID de sessão. Esse ID de sessão é usado para todos os comandos executados por esta chamada de método.
- **comandos**-Uma matriz de objetos de comando, um para cada comando que você deseja executar.
- **retorno de chamada** - Se o método foi bem-sucedido, a função de retorno de chamada chama `onSuccess`. Se o método falhou, a função de retorno de chamada chama `onError`.

O resultado de chamar esse método é equivalente a chamar explicitamente cada método na matriz Comando. Esse método minimiza o número de solicitações reais ao servidor de runtime. O servidor de runtime executa cada método em série; para cada chamada, qualquer erro ou aviso é capturado no objeto Resposta que corresponde a essa chamada de método. Se um erro for encontrado, executeBatch continua com o resto das chamadas no lote. Se a execução de qualquer método resultar em um erro, o status de nível superior para o objeto BatchResponse refletirá esse erro. Se nenhum erro tiver ocorrido, o status de nível superior refletirá quaisquer avisos que possam ter ocorrido. Se nenhum aviso tiver ocorrido, o status de nível superior refletirá uma execução bem-sucedida do lote.

Valor de retorno

O servidor de runtime responde a `executeBatch` com um objeto BatchResponse.

Exemplo de API JavaScript

```
function isJavaScriptAPISelected() {
    var radios = document.getElementsByName('api');
    for (var i = 0, length = radios.length; i < length; i++) {
        if (radios[i].checked) {
            if (radios[i].value === 'JavaScript')
                return true ;
            else  // only one radio can be logically checked
                break;
        }
    }
    return false;
}

function processFormForJSInvocation(e) {

    if (!isJavaScriptAPISelected())
        return;

    if (e.preventDefault) e.preventDefault();

    var serverurl = document.getElementById('serviceUrl').value ;
    InteractAPI.init( { "url" : serverurl } );

    var commandsToExecute = { "ssid" : null, "commands" : [] };
    var callback = InteractAPI.Callback.create(onSuccess, onError);

    callStartSession(commandsToExecute, callback);
    callGetOffers(commandsToExecute, callback);
    callGetOffersForMultipleInteractionPoints(commandsToExecute, callback);
    callPostEvent(commandsToExecute, callback);
    callSetAudience(commandsToExecute, callback);
```

```
    callGetProfile(commandsToExecute, callback);
    callEndSession(commandsToExecute, callback);
    callSetDebug(commandsToExecute, callback);
    callGetVersion(commandsToExecute, callback);

    callExecuteBatch(commandsToExecute, callback);

    // You must return false to prevent the default form behavior
    return false;
}

function callStartSession(commandsToExecute, callback) {

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

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                createStartSessionCmd(
                                icName, getNameValuePairs(audId),
                                audLevel, getNameValuePairs(params),
                                relyOldSs, debug));
    }
    else {
        InteractAPI.startSession(ssId, icName,
                               getNameValuePairs(audId), audLevel,
                               getNameValuePairs(params), relyOldSs,
                               debug, callback) ;
    }

}

function callGetOffers(commandsToExecute, callback) {

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

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                createGetOffersCmd(ip, nofRequested));
    }
    else {
        InteractAPI.getOffers(ssId, ip, nofRequested, callback);
    }
}

function callPostEvent(commandsToExecute, callback) {

    var ssId = document.getElementById('pe_sessionId').value;
```

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

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.
                                    CommandUtil.createPostEventCmd
                                    (ev, getNameValuePairs(params)));
    }
    else {
        InteractAPI.postEvent(ssId, ev, getNameValuePairs(params), callback);
    }
}

function callGetOffersForMultipleInteractionPoints
(commandsToExecute, callback) {

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

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

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

    //build get offer requests
    var getOffReqs = [];
    for(var i = 0; i < parts.length; i += 1) {
        var getofReqObj = parseGetOfferReq(parts[i]);
        if (getofReqObj) {
            getOffReqs.push(getofReqObj);
        }
    }

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                    createGetOffersForMultiple
                                    InteractionPointsCmd(getOffReqs));
    }
    else {
        InteractAPI.getOffersForMultipleInteractionPoints
                                    (ssId, getOffReqs, callback);
    }
}

function parseGetOfferReq(ofReqStr) {

    if (!ofReqStr || ofReqStr==="")
        return null;

    var posIp = ofReqStr.indexOf(',');
    var ip = ofReqStr.substring(0,posIp);
    var posNmReq = ofReqStr.indexOf(',', posIp+1);
    var numReq = ofReqStr.substring(posIp+1,posNmReq);
    var posDup = ofReqStr.indexOf(',', posNmReq+1);
    var dupPolicy = null;
```

```
    var reqAttributes = null;

    if (posDup===-1)
        dupPolicy = ofReqStr.substring(posNmReq+1);
    else
        dupPolicy = ofReqStr.substring(posNmReq+1,posDup);

    //check if request string has attributes
    var reqAttrPos = ofReqStr.search(/\(/g);
    if (reqAttrPos!==-1) {
        var reqAttributesStr = ofReqStr.substring(reqAttrPos);
        reqAttributesStr = trimString(reqAttributesStr);
        reqAttributesStr = removeOpenCloseBrackets(reqAttributesStr);
        reqAttributes = parseReqAttributes(reqAttributesStr);
    }

   return InteractAPI.GetOfferRequest.create(ip, parseInt(numReq),
                                parseInt(dupPolicy), reqAttributes);
}

//trim string
function trimString(strToTrim) {
    if (strToTrim)
        return strToTrim.replace(/^\s+|\s+$/g, "");
    else
        return null;
}

function trimStrArray(strArray) {
    if (!strArray) return ;
    for(var i = 0; i < strArray.length; i += 1) {
        strArray[i] = trimString(strArray[i]);
    }
}

//remove open and close brackets in the end
function removeOpenCloseBrackets(strToUpdate) {
    if (strToUpdate)
        return strToUpdate.replace(/^\(+|\)+$/g, "");
    else
        return null;
}

function parseReqAttributes(ofReqAttrStr) {

    //sanitize string
    ofReqAttrStr = trimString(ofReqAttrStr);
    ofReqAttrStr = removeOpenCloseBrackets(ofReqAttrStr);

    if (!ofReqAttrStr || ofReqAttrStr==="")
        return null;

    //get the number requested
    var pos = ofReqAttrStr.indexOf(",");
    var numRequested = ofReqAttrStr.substring(0,pos);
    ofReqAttrStr = ofReqAttrStr.substring(pos+1);

    //first part will be attribute and rest will be child attributes
    var parts = [];
    pos = ofReqAttrStr.indexOf(",");
    if (pos!==-1) {
        parts.push(ofReqAttrStr.substring(0,pos));
        parts.push(ofReqAttrStr.substring(pos+1));
    }
    else {
        parts.push(ofReqAttrStr);
    }
```

```
    for(var i = 0; i < parts.length; i += 1) {
        //sanitize string
        parts[i] = trimString(parts[i]);
        parts[i] = removeOpenCloseBrackets(parts[i]);
        parts[i] = trimString(parts[i]);
    }

    //build list of attributes
    var attributes = [];
    var idx = 0;
    if (parts[0]) {
        var attParts = parts[0].split(";");
        for (idx=0; idx<attParts.length; idx++) {
            attParts[idx] = trimString(attParts[idx]);
            attParts[idx] = removeOpenCloseBrackets(attParts[idx]);
            attParts[idx] = trimString(attParts[idx]);

            var atrObj = parseAttribute(attParts[idx]);
            if (atrObj) attributes.push(atrObj);
        }
    }

    //build list of child attributes
    var childAttributes = [];
    if (parts[1]) {
        var childAttParts = parts[1].split(")");
        for (idx=0; idx<childAttParts.length; idx++) {

            childAttParts[idx] = trimString(childAttParts[idx]);
            childAttParts[idx] = removeOpenCloseBrackets(childAttParts[idx]);
            childAttParts[idx] = trimString(childAttParts[idx]);


            //get the number requested
            var noReqPos = childAttParts[idx].indexOf(",");
            var numReqAt = childAttParts[idx].substring(0,noReqPos);
            childAttParts[idx] = childAttParts[idx].substring(noReqPos+1);
            childAttParts[idx] = trimString(childAttParts[idx]);


            var atrObjParsed = parseAttribute(childAttParts[idx]);
            if (atrObjParsed) {
                var childReq = InteractAPI.OfferAttributeRequirements.create
                (parseInt(numReqAt), [atrObjParsed], null);
                childAttributes.push(childReq);
            }
        }
    }

    return InteractAPI.OfferAttributeRequirements.create(parseInt(numRequested),
    attributes, childAttributes);
}

function parseAttribute(attStr) {

    attStr = trimString(attStr);

    if (!attStr || attStr==="")
        return null;

    var pos1 = attStr.indexOf("=");
    var pos2 = attStr.indexOf("|");
    var nvp = InteractAPI.NameValuePair.create
                          ( attStr.substring(0,pos1),
                            attStr.substring(pos1+1, pos2),
                            attStr.substring(pos2+1));
```

```
        return nvp;
    }
function callSetAudience(commandsToExecute, callback) {
        if (!document.getElementById('checkSetAudience').checked)
            return ;

        var ssId = document.getElementById('sa_sessionId').value;
        var audId = document.getElementById('sa_audienceId').value;
        var audLevel = document.getElementById('sa_audienceLevel').value;
        var params = document.getElementById('sa_parameters').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                            createSetAudienceCmd
            (getNameValuePairs(audId), audLevel, getNameValuePairs(params)));
        }
        else {
            InteractAPI.setAudience(ssId, getNameValuePairs(audId),
                                    audLevel, getNameValuePairs(params),
                                    callback);
        }
    }

    function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createGetProfileCmd());
        }
        else {
            InteractAPI.getProfile(ssId, callback);
        }
    }

    function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createEndSessionCmd());
        }
        else {
            InteractAPI.endSession(ssId, callback);
        }
    }

    function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
```

```
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createSetDebugCmd(isDebug));
    }
    else {
        InteractAPI.setDebug(ssId, isDebug, callback);
    }
}

function callGetVersion(commandsToExecute, callback) {

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createGetVersionCmd());
    }
    else {
        InteractAPI.getVersion(callback);
    }
}

function callExecuteBatch(commandsToExecute, callback) {

    if (!commandsToExecute)
        return ;

    InteractAPI.executeBatch(commandsToExecute.ssid,
    commandsToExecute.commands, callback);
}

function getNameValuePairs(parameters) {

    if (parameters === '')
        return null;

    var parts = parameters.split(';');
    var nvpArray = new Array(parts.length);

    for(i = 0; i < parts.length; i += 1) {
        var nvp = parts[i].split(',') ;
        var value = null;
        if (nvp[2]===InteractAPI.NameValuePair.prototype.TypeEnum.NUMERIC) {
            if (isNaN(nvp[1])) {
                value = nvp[1];  //a non number was provided as number,
                pass it to API as it is
            }
            else {
                value = Number(nvp[1]);
            }
        }
        else {
            value = nvp[1];
        }
        //special handling NULL value
        if (value && typeof value === 'string') {
            if (value.toUpperCase() === 'NULL') {
                value = null;
            }
        }
        nvpArray[i] = InteractAPI.NameValuePair.create(nvp[0], value, nvp[2]) ;
    }

    return nvpArray;
}
```

```
function showResponse(textDisplay) {
    var newWin = open('','Response','height=300,width=300,titlebar=no,
    scrollbars=yes,toolbar=no,
   resizable=yes,menubar=no,location=no,status=no');

    if (newWin.locationbar !== 'undefined' && newWin.locationbar
    && newWin.locationbar.visible)
        newWin.locationbar.visible = false;

    var displayHTML = '<META HTTP-EQUIV="Content-Type"
    CONTENT="text/html; charset=UTF-8">
    <html><head><style>TD { border-width : thin; border-style : solid }</style.'
                        + "<script language='Javascript'>"
                        + "var desiredDomain = 'unicacorp.com'; "
                        + "if (location.href.indexOf(desiredDomain)>=0) "
                        + "{ document.domain = desiredDomain;} "
                        + "</script></head><body> "
                        + textDisplay
                        + "</body></html>" ;
    newWin.document.body.innerHTML = displayHTML;
    newWin.focus() ;
}

function onSuccess(response) {
    showResponse("********Response********<br> " + JSON.stringify(response)) ;
}

function onError(response) {
    showResponse("********Error********<br> " + response) ;
}

function formatResoponse(response) {

}

function printBatchResponse(batResponse) {

}

function printResponse(response) {

}
```

Objeto JavaScript de resposta de exemplo onSuccesss

Este exemplo mostra as três variáveis para o objeto JavaScript de resposta; offerLists, messages e profile.

`offerList` retorna uma lista não nula se você chamar `getOffer` ou `getOffersForMultipleInteractionPoints` como uma API ou como parte de seus comandos em lote. Você deve sempre verificar o nulo neste antes de executar qualquer operação nesta variável.

Você deve sempre verificar o status da resposta `messages` do JavaScript.

`Profile` é um não nulo retornado se você usar `getProfile` como uma API ou parte de seus comandos em lote. Se você não usar `getProfile`, é possível ignorar esta variável. Você deve sempre verificar o nulo neste antes de executar qualquer operação nesta variável.

```
function onSuccess(response)
InteractAPI.ResponseTransUtil._buildResponse = function(response) {
        'use strict';
```

```
    if (!response) return null;

    var offerList = null;
    //transform offerLists to JS Objects
    if (response.offerLists) {
        offerList = [];
        for (var ofListCt=0; ofListCt<response.offerLists.length;ofListCt++) {
            var ofListObj = this._buildOfferList(response.offerLists[ofListCt]);
            if (ofListObj) offerList.push(ofListObj);
        }
    }

    var messages = null;
    //transform messages to JS Objects
    if (response.messages) {
        messages = [];
        for (var msgCt=0; msgCt<response.messages.length;msgCt++) {
            var msgObj = this._buildAdvisoryMessage(response.messages[msgCt]);
            if (msgObj) messages.push(msgObj);
        }
    }

    var profile = null;
    //transform profile nvps to JS Objects
    if (response.profile) {
        profile = [];
        for (var nvpCt=0; nvpCt<response.profile.length;nvpCt++) {
            var nvpObj = this._buildNameValuePair(response.profile[nvpCt]);
            if (nvpObj) profile.push(nvpObj);
        }
    }

    return InteractAPI.Response.create(response.sessionId,
                                       response.statusCode, offerList,
                                       profile, response.version,
                                       messages) ;
};
```

Capítulo 9. Sobre a API ExternalCallout

O Interact oferece uma macro extensível, a EXTERNALCALLOUT, para ser usada com os fluxogramas interativos. Essa macro permite executar a lógica customizada para comunicar-se com sistemas externos durante execuções de fluxograma. Por exemplo, se você desejar calcular a pontuação de crédito de um cliente durante uma execução de fluxograma, será possível criar uma classe Java (um texto explicativo) para fazer isso e, em seguida, usar a macro EXTERNALCALLOUT em um processo de seleção no fluxograma interativo para obter a pontuação de crédito a partir do texto explicativo.

A configuração de EXTERNALCALLOUT tem duas etapas principais. Primeiro, você deve criar uma classe Java que implemente a API ExternalCallout. Segundo, você deve configurar as propriedades de configuração necessárias do Marketing Platform no servidor de runtime na categoria Interact | fluxograma | ExternalCallouts.

Além das informações nesta seção, o JavaDoc para a API ExternalCallout está disponível em qualquer servidor de runtime do Interact no diretório Interact/docs/externalCalloutJavaDoc.

Interface IAffiniumExternalCallout

A API ExternalCallout está contida na interface IAffiniumExternalCallout. Você deve implementar a interface IAffiniumExternalCallout para usar a macro EXTERNALCALLOUT.

A classe que implementa o IAffiniumExternalCallout deve ter um construtor com o qual possa ser inicializada pelo servidor de runtime.

- Se não houver construtores na classe, o compilador Java criará um construtor padrão e isso será suficiente.
- Se houver construtores com argumentos, um construtor público sem argumento deverá ser fornecido, que será usado pelo servidor de runtime.

Ao desenvolver o texto explicativo externo, lembre-se do seguinte:

- Cada avaliação de expressão com um texto explicativo externo cria uma nova instância da classe. Você deve gerenciar problemas de segurança de encadeamento para membros estáticos na classe.
- Se o texto explicativo externo usar recursos do sistema, como arquivos ou uma conexão com o banco de dados, você deverá gerenciar as conexões. O servidor de runtime não possui um recurso para limpar conexões automaticamente.

Você deve compilar a implementação com relação ao interact_externalcallout.jar localizado no diretório lib da instalação do ambiente de tempo de execução do IBM Interact.

O IAffiniumExternalCallout permite que o servidor de runtime solicite dados a partir da classe Java. A interface consiste em quatro métodos:

- getNumberOfArguments
- getValue
- initialize

- shutdown

Incluindo um serviço da web a ser usado com a macro EXTERNALCALLOUT

Use este procedimento para incluir um serviço da Web a ser usado com a macro EXTERNALCALLOUT. A macro EXTERNALCALLOUT somente reconhecerá os textos explicativos se você tiver definido as propriedades de configuração apropriadas.

Procedimento

No Marketing Platform para o ambiente de tempo de execução, inclua ou defina as propriedades de configuração a seguir na categoria Interact > fluxograma > externalCallouts.

Propriedade de configuração	Configuração
externalCallouts category	Crie uma categoria para o texto explicativo externo
class	Os nomes de classes para o texto explicativo externo
classpath	O caminho de classe para os arquivos de classe de texto explicativo externo
Parameter Data category	Se o texto explicativo externo precisar de parâmetros, crie novas propriedades de configuração do parâmetro para eles e designe um valor a cada um

getNumberOfArguments

O método getNumberOfArguments retorna o número de argumentos esperados pela classe Java com a qual você está integrando.

```
getNumberOfArguments()
```

Valor de retorno

O método getNumberOfArguments retorna um número inteiro.

Exemplo

O exemplo a seguir mostra a impressão do número de argumentos.

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

O método getValue executa a funcionalidade principal do texto explicativo e retorna os resultados.

```
getValue(audienceID, configData, arguments)
```

O método getValue requer os parâmetros a seguir:

- **audienceID** - um valor que identifica o ID de público.
- **configData** - um mapa com pares de valores de chave de dados de configuração necessários pelo texto explicativo.
- **arguments** - os argumentos necessários pelo texto explicativo. Cada argumento pode ser uma Sequência, Duplo, Data ou uma Lista de um desses. Um argumento Lista pode conter valores nulos, no entanto, uma Lista não pode conter, por exemplo, uma Sequência e um Duplo.

  A verificação do tipo de argumento deve ser feita em sua implementação.

Se o método getValue falhar por qualquer motivo, ele retornará CalloutException.

Valor de retorno

O método getValue retorna uma lista de Sequências.

Exemplo

```
public List<String> getValue(AudienceId audienceId, Map<String,
  String> configurationData, Object... arguments) throws CalloutException
{
 Long customerId = (Long) audienceId.getComponentValue("Customer");
 // now query scoreQueryUtility for the credit score of customerId
 Double score = scoreQueryUtility.query(customerId);
 String str = Double.toString(score);
 List<String> list = new LinkedList<String>();
 list.add(str);
 return list;
}
```

Inicializar

O método `initialize` é chamado uma vez quando o servidor de runtime é iniciado. Se houver quaisquer operações que possam impedir o desempenho durante o tempo de execução, como carregar uma tabela de banco de dados, elas devem ser executadas por este método.

```
initialize(configData)
```

O método `initialize` requer o parâmetro a seguir:

- **configData** - um mapa com pares de valores de chave de dados de configuração necessários pelo texto explicativo.

  Interact lê estes valores a partir dos parâmetros External Callout definidos na categoria `Interact > Flowchart > External Callouts > [External Callout] > Parameter Data`.

Se o método `initialize` falhar por qualquer motivo, ele retornará `CalloutException`.

Valor de retorno

None.

Exemplo

```
public void initialize(Map<String, String> configurationData) throws CalloutException
{
    // configurationData has the key-value pairs specific to the environment
    // the server is running in
    // initialize scoreQueryUtility here
}
```

shutdown

O método `shutdown` é chamado uma vez quando o servidor de runtime é encerrado. Se houver quaisquer tarefas de limpeza necessárias a seu texto explicativo, elas devem ser executadas neste momento.

```
shutdown(configData)
```

O método `shutdown` requer o parâmetro a seguir:

- **configData** - um mapa com pares de valores de chave de dados de configuração necessários pelo texto explicativo.

Se o método shutdown falhar por qualquer motivo, ele retornará CalloutException.

Valor de retorno

None.

Exemplo

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
    // shutdown scoreQueryUtility here
}
```

Exemplo da API ExternalCallout

Este exemplo cria um texto explicativo externo que obtém uma pontuação de crédito.

Crie um texto explicativo externo que obtenha uma pontuação de crédito:

1. Crie um arquivo chamado GetCreditScore.java com o seguinte conteúdo. Este arquivo assume que haja uma classe chamada ScoreQueryUtility que busque uma pontuação a partir de um aplicativo de modelagem.

```
import java.util.Map;
import com.unicacorp.interact.session.AudienceId;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.IAffiniumExternalCallout;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.CalloutException;
import java.util.Random;

public class GetCreditScore implements IAffiniumExternalCallout
{
 // the class that has the logic to query an external system for a customer's credit score
 private static ScoreQueryUtility scoreQueryUtility;
 public void initialize(Map<String, String> configurationData) throws CalloutException
 {
  // configurationData has the key- value pairs specific to the environment the server is running in
  // initialize scoreQueryUtility here
 }

 public void shutdown(Map<String, String> configurationData) throws CalloutException
 {
 // shutdown scoreQueryUtility here
 }

 public int getNumberOfArguments()
 {
  // do not expect any additional arguments other than the customer's id
  return 0;
 }

 public List<String> getValue(AudienceId audienceId, Map<String, String> configurationData,
  Object... arguments) throws CalloutException
 {
  Long customerId = (Long) audienceId.getComponentValue("Customer");
  // now query scoreQueryUtility for the credit score of customerId
  Double score = scoreQueryUtility.query(customerId);
  String str = Double.toString(score);
  List<String> list = new LinkedList<String>();
  list.add(str);
  return list;
 }
}
```

2. Compile GetCreditScore.java para GetCreditScore.class.

3. Crie um arquivo JAR chamado `creditscore.jar` contendo `GetCreditScore.class` e os outros arquivos de classe que ele usa.
4. Copie o arquivo JAR em algum local no servidor de runtime, por exemplo `/data/interact/creditscore.jar`.
5. Crie um Texto explicativo externo com o nome `GetCreditScore` e o caminho de classe como `/data/interact/creditscore.jar` na categoria `externalCallouts` na página Gerenciar Configurações.
6. Em um fluxograma interativo, o texto explicativo pode ser usado como `EXTERNALCALLOUT('GetCreditScore')`.

Interface IInteractProfileDataService

A API de Serviços de dados do perfil está contida na interface do `iInteractProfileDataService`. Essa interface permite importar dados hierárquicos em uma sessão do Interact por meio de uma ou mais origens de dados externas (como um arquivo simples, serviço da web e assim por diante) no momento em que a sessão do Interact é iniciada ou o ID de público de uma sessão do Interact é alterado.

Para desenvolver a importação de dados hierárquica usando a API de Serviços de dados de perfil, você deve gravar uma classe Java que obtenha informações a partir de qualquer origem de dados, as mapeie para um objeto `ISessionDataRootNode` e, em seguida, fazer referência a esses dados mapeados usando a macro EXTERNALCALLOUT em um processo de seleção de um fluxograma interativo.

Você deve compilar a implementação com relação ao `interact_externalcallout.jar` localizado no diretório `lib` da instalação do ambiente de tempo de execução do IBM Interact.

Para obter um conjunto completo de documentação do Javadoc para usar esta interface, visualize os arquivos em *Interact_home*`/docs/externalCalloutJavaDoc` com qualquer navegador da web.

Para obter uma implementação de amostra de como usar o Serviço de dados de perfil, incluindo descrições comentadas de como o exemplo foi implementado, consulte *Interact_home*`/samples/externalcallout/XMLProfileDataService.java`.

Nota: A implementação de amostra deve ser usada somente como exemplo. Não é necessário usar essa amostra em sua implementação.

Incluindo uma origem de dados para ser usada com os Serviços de dados do perfil

Use este procedimento para incluir uma origem de dados a ser usada com os Serviços de dados de perfil.

Sobre Esta Tarefa

A macro `EXTERNALCALLOUT` somente reconhecerá uma origem de dados para a importação de dados hierárquicos dos Serviços de dados de perfil se você tiver definido as propriedades de configuração apropriadas.

Procedimento

No Marketing Platform para o ambiente de tempo de execução, inclua ou defina as seguintes propriedades de configuração na categoria `Interact > perfil > Níveis de público [AudienceLevelName] > Serviços de dados de perfil`.

Propriedade de configuração	Configuração
Categoria `Nome da nova categoria`	O nome da origem de dados que está sendo definida. O nome inserido aqui deve ser exclusivo entre as origens de dados do mesmo nível de público.
`enabled`	Indica se a origem de dados está ativada para o nível de público no qual ela está definida.
`className`	O nome completo da classe de origem de dados que implementa o `IInteractProfileDataService`
`classPath`	O caminho de classe para os arquivos de classe dos Serviços de dados de perfil. Se for omitido, o caminho de classe do servidor de aplicativos que o contém será usado por padrão.
`priority` category	A prioridade desta origem de dados neste nível de público. Deve ser um valor exclusivo entre todas as origens de dados para cada nível de público. (Ou seja, se uma prioridade for definida como 100 para uma origem de dados, nenhuma outra origem de dados no nível de público poderá ter a prioridade 100.)

Interface IParameterizableCallout

A API de Callout Parameterizable está contida na interface `IParameterizableCallout`.

Essa interface é a interface base das interfaces de API expostas que podem aceitar parâmetros da configuração por meio do Marketing Platform. Como esta é uma interface base, ela não deveria ser implementada diretamente. Os parâmetros são recuperados dos nós-filhos do nó `Parameter Data` sob a categoria que faz referência a esta implementação. No exemplo a seguir, ESB é uma implementação customizada do serviço de dados do perfil, que, por sua vez, implementa a interface `IParameterizableCallout`. Os parâmetros endPoint e login, juntamente com seus valores são transmitidos para esta classe de implementação quando o mecanismo do Interact tenta inicializar e finalizá-la.

```
Profile Data Services
...ESB
   ...Parameter Data
      ...endPoint
      ...login
```

A interface consiste em dois métodos:

- `initialize`
- `shutdown`

initialize

O método `initialize` inicializa esta classe de implementação.

```
void initialize(java.util.Map<java.lang.String,java.lang.String> configurationData)
               throws CalloutException
```

O método `initialize` requer o parâmetro a seguir:

- **configurationData** – um mapa com pares de nome e valor de parâmetros configurados pelos usuários

Lançamentos

`CalloutException`

shutdown

O método shutdown encerra esta classe de implementação.

```
void shutdown(java.util.Map<java.lang.String,java.lang.String> configurationData)
              throws CalloutException
```

O método shutdown requer o parâmetro a seguir:

- **configurationData** – um mapa com pares de nome e valor de parâmetros configurados pelos usuários

Lançamentos

CalloutException

Interface ITriggeredMessageAction

A API de Ação da Mensagem Acionada está contida na interface ITriggeredMessageAction. Esta interface permite que você obtenha e configure o nome dessa instância.

A interface ITriggeredMessageAction serve como uma interface base para outras interfaces e nunca deve ser implementada diretamente.

A interface consiste em dois métodos:

- getName
- setName

getName

O método getName retorna o nome da instância ITriggeredMessageAction.

```
java.lang.String getName()
```

setName

O método setName define o nome da instância ITriggeredMessageAction.

```
void setName(java.lang.String name)
```

Enquanto você inicializa a classe de implementação desta interface, o Interact configura o nome da interface com o nome fornecido na UI de configuração.

No exemplo a seguir, o nome desse gateway é InteractLog.

```
triggeredMessage
           ...gateways
              ...InteractLog
```

O método setName requer o parâmetro a seguir:

- name – o nome que você deseja configurar para a instância ITriggeredMessageAction.

Interface IChannelSelector

A API do Seletor de Canal é contida na interface IChannelSelector. Este interface permite que você selecione os canais de saída com base na oferta a ser enviada e nos atributos da sessão.

Para uma implementação de amostra de como usar a Ação da Mensagem Acionada, incluindo descrições comentadas de como o exemplo foi implementado, veja *Interact_home*/samples/triggeredmessage/SampleChannelSelector.java.

Nota: A implementação de amostra deve ser usada somente como exemplo. Não é necessário usar essa amostra em sua implementação.

Você deve tentar usar esta implementação em vez de gravar sua própria.

A interface consiste em um método:

- selectChannels

selectChannels

O método selectChannels seleciona os canais de saída que com os quais a oferta transmitida deve ser enviada com a interface IChannelSelector.

```
java.util.List<java.lang.String> selectChannels
                  (java.util.Map<java.lang.String,java.util.Map<java.lang.String,
                                 java.lang.Object>> availableChannels,
                                 com.unicacorp.interact.api.Offer offer,
                                 com.unicacorp.interact.treatment.
                                 optimization.IInteractSessionData sessionData)
```

O Interact tenta enviar esta oferta para todos aqueles canais retornados.

O método selectChannels requer os parâmetros a seguir:

- **availableChannels** - um mapa dos canais de saída disponíveis, que são configurados na UI da Mensagem Acionada nas configurações no tempo de design do Interact. Em cada entrada do mapa, a chave é o nome do canal e o valor são os parâmetros configurados para esse canal no tempo de design do Interact . A ordem de iteração deste mapa corresponde à ordem definida nessa UI. Se Canal Preferido do Perfil for usado na UI da Mensagem Acionada, ele será substituído pelo canal real antes do método ser chamado. Além disso, se o mesmo canal ocorrer diversas vezes na UI, apenas a ocorrência com a mais alta prioridade será mantida e todas as duplicatas serão removidas.
- **offer** - a oferta a ser entregue
- **sessionData** - os atributos atualmente armazenados na sessão associada do Interact

Interface IDispatcher

A API do Dispatcher está contida na interface IDispatcher. Esta interface envia ofertas para gateways de destino.

Como existe apenas uma instância dessa classe para cada dispatcher configurado, a implementação dessa interface deve ser stateless a partir da perspectiva do Interact.

Para uma implementação de amostra de como usar a Ação da Mensagem Acionada, incluindo descrições comentadas de como o exemplo foi implementado, veja *Interact_home*/samples/triggeredmessage/SampleDispatcher.java.

Nota: A implementação de amostra deve ser usada somente como exemplo. Não é necessário usar essa amostra em sua implementação.

Você deve tentar usar esta implementação em vez de gravar sua própria.

A interface consiste em um método:

- dispatch

dispatch

O método dispatch envia ofertas aos gateways de destino na interface IDispatcher.

```
boolean dispatch(java.lang.String channel,
            java.lang.String gatewayName,
            java.util.Collection<com.unicacorp.interact.api.Offer> offers,
            com.unicacorp.interact.api.NameValuePair[] profileData)
              throws com.unicacorp.interact.exceptions.InteractException
```

Depois que os canais de saída são selecionadas para uma oferta do candidato, o Interact tenta enviar as ofertas do candidato aos manipuladores associados ao canal. São feitas tentativas nos manipuladores com base em suas prioridades definidas, de alta para baixa. Para cada manipulador, o Interact chama este método do dispatcher configurado. A implementação desta instância do dispatcher decide como rotear a oferta para o gateway de destino, que está configurado no mesmo manipulador. Se houver diversas ofertas enviadas para o mesmo manipulador como resultado da mesma avaliação da mensagem acionada, o Interact tenta enviar todas essas ofertas em um lote.

O método dispatch requer os parâmetros a seguir:

- **channel** - o canal de saída para o qual estas ofertas são enviadas
- **gatewayName** - o nome do gateway de destino
- **offers** - as ofertas a serem enviadas para o gateway em um lote
- **profileData** - atributos de perfil preenchidos pelo IGateway.validate e passados para o IGateway.deliver

Valor de retorno

O método dispatch é retornado se o dispatch foi bem-sucedido ou falhou

Lançamentos

com.unicacorp.interact.exceptions.InteractException

Interface IGateway

A API do Gateway está contida na interface IGateway. Esta interface recebe ofertas do Interact e envia a ofertas para seu destino.

Cada implementação desta interface se comunica com um destino particular. O destino deve executar a transformação de dados necessária, o preenchimento de atributos e o trabalho relacionado ao destino semelhante.

Para uma implementação de amostra de como usar a Ação da Mensagem Acionada, incluindo descrições comentadas de como o exemplo foi implementado, veja *Interact_home*/samples/triggeredmessage/SampleOutboundGateway.java.

Nota: A implementação de amostra deve ser usada somente como exemplo. Não é necessário usar essa amostra em sua implementação.

A interface consiste em dois métodos:

- deliver

- validate

deliver

O método deliver é chamado para enviar a oferta ou as ofertas para um destino na interface IGateway.

```
void deliver(java.util.Collection<com.unicacorp.interact.api.Offer> offers,
                    com.unicacorp.interact.api.NameValuePair[] profileData,
                    java.lang.String channel)
```

O método deliver requer os parâmetros a seguir:

- **offers** - a oferta a ser enviada
- **profileData** - os atributos de perfil que o método validate preenche no parameterMap
- **channel** - o canal de saída para o qual estas ofertas serão enviadas

validate

O método validate valida ofertas do candidato na interface IGateway.

```
java.util.Collection<com.unicacorp.interact.api.Offer> validate
    (com.unicacorp.interact.treatment.optimization.
     IInteractSessionData sessionData,
           java.util.Collection<com.unicacorp.interact.api.Offer> candidateOffers,
           java.util.Map<java.lang.String,java.lang.Object> parameterMap,
           java.lang.String channel)
```

O mecanismo do Interact chama este método para validar as ofertas candidatas. A implementação deste método deve verificar as ofertas, os atributos da oferta e os atributos de sessão com relação aos requisitos do destino para determinar qual oferta ou quais ofertas podem ser enviadas por meio desse gateway. Além disso, ele pode incluir parâmetros necessários no mapa transmitido, que é transmitido de volta para o método deliver.

O método validate requer os parâmetros a seguir:

- **sessionData** - os atributos atualmente armazenados na sessão associada do Interact
- **candidateOffers** – a ofertas que o Interact selecionou com base no método de seleção de oferta, seus parâmetros e outros fatores. Essas ofertas são elegíveis para serem entregues a partir da perspectiva do Interact, mas ainda sujeitos ao gateway.
- **parameterMap** - um mapa que a implementação deste método deve usar para transmitir parâmetros para seu método de entrega
- **channel** - o canal de saída para o qual estas ofertas serão enviadas

Capítulo 10. Utilitários do IBM Interact

Esta seção descreve os utilitários administrativos disponíveis com o Interact.

Executar utilitário de implementação (runDeployment.sh/.bat)

A ferramenta de linha de comandos `runDeployment` permite implementar um canal interativo para um grupo de servidores específico da linha de comandos, usando as configurações fornecidas por um arquivo `deployment.properties` que descreve todos os parâmetros possíveis e está disponível no mesmo local que a própria ferramenta `runDeployment`. A capacidade de executar uma implementação de canal interativo a partir da linha de comandos é especificamente útil quando estiver usando o recurso `OffersBySQL`. Por exemplo, você pode configurar um fluxograma em lote do Campaign para executar periodicamente. Quando a execução do fluxograma for concluída, um acionador poderá ser chamado para inicializar a implementação das ofertas na tabela `OffersBySQL` usando esta ferramenta de linha de comandos.

Descrição

É possível localizar a ferramenta de linha de comandos `runDeployment` instalada automaticamente no servidor Interact Design Time, no local a seguir:

Interact_home`/interactDT/tools/deployment/runDeployment.sh` (ou `runDeployment.bat` em um servidor Windows)

O único argumento transmitido para o comando é o local de um arquivo chamado `deployment.properties` que descreve todos os possíveis parâmetros necessários para implementar a combinação canal interativo/grupo de servidores de tempo de execução. Um arquivo de amostra é fornecido para referência.

Nota: Antes de usar o utilitário `runDeployment`, você deve primeiro editá-lo com qualquer editor de texto para fornecer o local do Java Runtime Environment no servidor. Por exemplo, é possível especificar *Interact_home*`/jre` ou *Platform_home*`/jre` como o caminho, se um desses diretórios contiver o Java Runtime que deseja que o utilitário use. Como alternativa, você poderia fornecer o caminho para qualquer Java Runtime Environment que é fornecido para uso com esta liberação dos produtos IBM .

Usando o utilitário runDeployment em um ambiente seguro (SSL)

Para usar o utilitário runDeployment quando a segurança foi ativada no servidor Interact (e portanto, conectando por uma porta SSL), é necessário incluir a propriedade Java do armazenamento confiável como a seguir:

1. Quando estiver editando o arquivo `deployment.properties` para sua implementação de canal interativo, modifique a propriedade `deploymentURL` para usar a URL de SSL segura, como neste exemplo:

   `deploymentURL=https://<`*HOST*`>.<`*DOMAIN*`>:<`*PORT*`>/Campaign/interact/InvokeDeploymentServlet`
2. Edite o script `runDeployment.sh` ou `runDeployment.bat` usando qualquer editor de texto para incluir o argumento a seguir na linha que começa com ${JAVA_HOME}:

`-Djavax.net.ssl.trustStore=<TrustStorePath>`

Por exemplo, a linha pode ser semelhante com isto após incluir o argumento de armazenamento confiável:

```
${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
-cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
InvokeDeploymentClient $1
```

Substitua *<TrustStorePath>* pelo camnho para o armazenamento confiável de SSL real.

Executando o utilitário

Após ter editado o utilitário para fornecer o Java Runtime Environment e ter customizado uma cópia do arquivo `deployment.properties` para corresponder a seu ambiente, é possível executar o utilitário com este comando:

Interact_home`/interactDT/tools/deployment/runDeployment.sh` *deployment.properties*

Substitua *Interact_home* pelo valor real da instalação de tempo de design do Interact e substitua *deployment.properties* pelo caminho real e nome do arquivo de propriedades que você customizou para esta implementação.

Arquivo `deployment.properties` de amostra

O arquivo `deployment.properties` de amostra contém uma lista comentada de todos os parâmetros que você deve customizar para corresponder ao seu próprio ambiente. O arquivo de amostra também contém comentários que explicam o que cada parâmetro é e por que pode ser necessário customizar um determinado valor.

```
###############################################################################
#
# As propriedades a seguir são alimentadas no programa InvokeDeploymentClient.
# O programa procurará uma configuração deploymentURL.  O programa postará uma
# solicitação com relação a esta url; todas as outras configurações serão postadas como parâmetros no
# solicitação. O programa verificará o status da implementação e
# retornará novamente quando a implementação estiver em um estado terminal (ou se o
# waitTime especificado tiver sido atingido).
#
# A saída do programa será deste formato:
# <STATE> : <Misc Detail>
#
# em que o estado pode ser um dos seguintes:
# ERROR
# RUNNING
# SUCCESS
#
# Misc Detail são os dados que preencheriam normalmente a área da mensagem de status
# na gui de implementação da página de resumo de IC. NOTA: as marcas HTML podem existir
# no Misc Detail
#
###############################################################################

###############################################################################
# deploymentURL: url para o servlet InvokeDeployment que reside no
# tempo de design  deve estar no formato a seguir:
# http://dt_host:port/Campaign/interact/InvokeDeploymentServlet
###############################################################################
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet

###############################################################################
# dtLogin: este é o login que você usaria para efetuar login no
Tempo de Design se
```

```
# quisesse implementar o IC pela GUI de implementação dentro do
resumo da página de
# do IC.
################################################################################
dtLogin=asm_admin

################################################################################
# dtPW: este é o PW que vem junto com o dtLogin
################################################################################
dtPW=

################################################################################
# icName: este é o nome do Canal Interativo que deseja implementar
################################################################################
icName=ic1

################################################################################
# partição: este é o nome da partição
################################################################################
partition=partition1

################################################################################
# solicitação: este é o tipo de solicitação que deseja que esta
ferramenta execute
# atualmente, há dois comportamentos.  Se o valor for "deploy", a
implementação será
# executada.  Todos os outros valores fariam com que a ferramenta
simplesmente retornasse o
# status da última implementação do ID especificado.
################################################################################
request=deploy

################################################################################
# serverGroup: este é o nome do grupo de servidores que gostaria de
# implementar o IC.
################################################################################
serverGroup=defaultServerGroup

################################################################################
# serverGroupType: itso indicará se esta implementação vai contra ou
não
# o grupo de servidor de produção ou um grupo de servidor de teste. 1
denota a produção
# 2 denota o teste.
################################################################################
serverGroupType=1

################################################################################
# rtLogin: esta é a conta usada para autenticar com relação ao grupo
de servidores
# ao qual está implementando.
################################################################################
rtLogin=asm_admin

################################################################################
# rtPW: esta é a senha associada ao rtLogin
################################################################################
rtPW=

################################################################################
# waitTime: Quando a ferramenta envia a solicitação de implementação, a ferramenta verificará
# o status da implementação.  Se a implementação não tiver sido
concluída (ou
# com falha) a ferramenta continuará pesquisando o status no
sistema até que
# um estado concluído tenha sido atingido OU até o waitTime
especificado (em
```

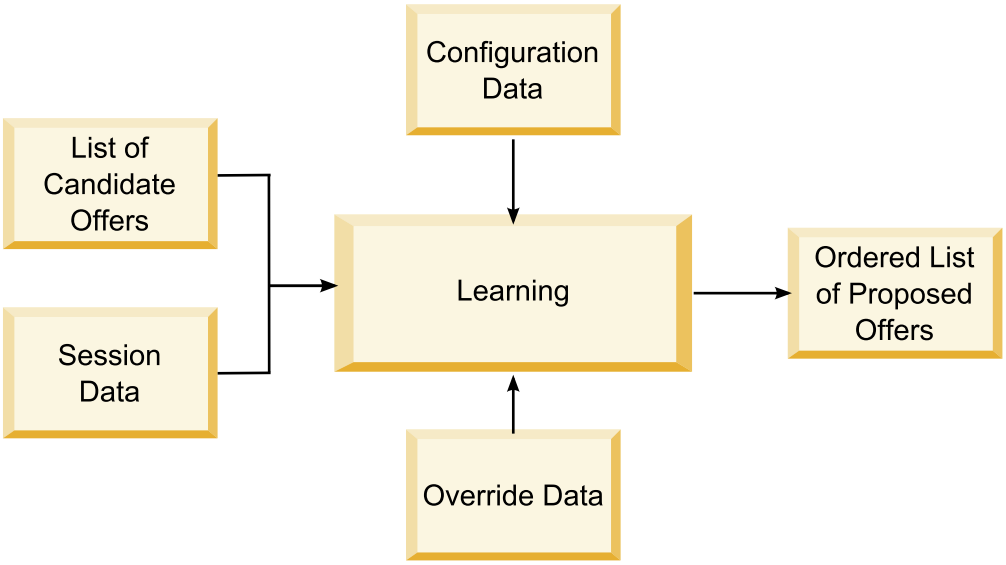
```
# segundos) tenha sido atingido.
################################################################################
waitTime=5

################################################################################
# pollTime: Se o status de uma implementação ainda estiver em estado
de execução, a
# ferramenta continuará a verificar o status.  Ela suspenderá
verificações de
# status um número de segundos baseado na configuração pollTime.
################################################################################
pollTime=3

################################################################################
# global: Configurar como false fará com que a ferramenta NÃO
implemente as configurações globais.
# A não disponibilidade da propriedade ainda implementará
as configurações globais.
################################################################################
global=true
```

Capítulo 11. Sobre a API de aprendizado

O Interact oferece um módulo de aprendizado que usa um algoritmo naive-bayesian para monitorar ações do visitante e propor ofertas ideais (em termos de aceitação). É possível implementar a mesma interface Java com seus próprios algoritmos usando a API de aprendizado para criar seu próprio módulo de aprendizado.

Nota: Se usar o Aprendizado externo, o exemplo relata sobre o aprendizado (relatórios Detalhes do aprendizado da oferta interativa e Análise de elevação do segmento interativo) não retornam dados válidos.

No nível mais simples, a API de aprendizado fornece métodos para coletar dados do ambiente de tempo de execução e retornar uma lista ordenada de ofertas recomendadas.



É possível coletar os dados a seguir do Interact

- Dados de contato da oferta
- Dados de aceitação da oferta
- Todos os dados da sessão
- Dados de oferta específicos do Campaign
- As propriedades de configuração definidas na categoria `learning` para o ambiente de design e a categoria `offerserving` para o ambiente de tempo de execução

É possível usar esses dados em seus algoritmos para cria uma lista de ofertas propostas. Você retorna uma lista de ofertas recomendadas, na ordem de recomendação mais alta para mais baixa.

Embora não mostrado no diagrama, também é possível usar a API de aprendizado para coletar dados para sua implementação de aprendizado. É possível manter esses dados na memória ou registrá-los em um arquivo ou banco de dados para análise adicional.

Após criar suas classes Java, é possível convertê-las para arquivos jar. Depois de criar seus arquivos jar, você também deve configurar o ambiente de tempo de execução para reconhecer seu módulo de aprendizado externo editando propriedades de configuração. Você deve copiar suas classes Java ou arquivos jar para cada servidor de runtime usando seu módulo de aprendizado externo.

Além das informações neste guia, o JavaDoc para a API do otimizador de aprendizado está disponível em qualquer servidor de runtime no diretório `Interact/docs/learningOptimizerJavaDoc`.

Você deve compilar sua implementação no `interact_learning.jar` localizado no diretório `lib` de sua instalação de ambiente de tempo de execução do Interact.

Ao gravar sua implementação de aprendizado customizada, você deve manter as diretrizes a seguir em mente.

- O desempenho é crítico.
- Deve-se trabalhar com multiencadeamento e ter segurança de encadeamento.
- Deve-se gerenciar todos os recursos externos com modos de falha e desempenho em mente.
- Use exceções, criação de log (log4j) e memória de forma apropriada.

Configurando o ambiente de tempo de execução para reconhecer módulos de aprendizado externo

É possível usar a API de aprendizado Java para gravar seu próprio módulo de aprendizado. Você deve configurar o ambiente de tempo de execução para reconhecer seu utilitário de aprendizado no Marketing Platform.

Sobre Esta Tarefa

Você deve reiniciar o servidor de runtime do Interact para que estas mudanças entrem em vigor.

Procedimento

1. No Marketing Platform para o ambiente de tempo de execução, edite as propriedades de configuração a seguir na categoria `Interact > offerserving`. As propriedades de configuração para a API de otimizador de aprendizado existem na categoria `Interact > offerserving > External Learning Config`.

Propriedade de configuração	Configuração
`optimizationType`	**ExternalLearning**
`externalLearningClass`	nome de classe para o aprendizado externo
`externalLearningClassPath`	O caminho para a classe ou arquivos JAR no servidor de runtime para o aprendizado externo. Se você estiver usando um grupo de servidores e todos os servidores de runtime fizerem referência à mesma instância do Marketing Platform, cada servidor deverá ter uma cópia da classe ou arquivos JAR no mesmo local.

2. Reinicie o servidor de runtime do Interact para que estas mudanças entrem em vigor.

Interface ILearning

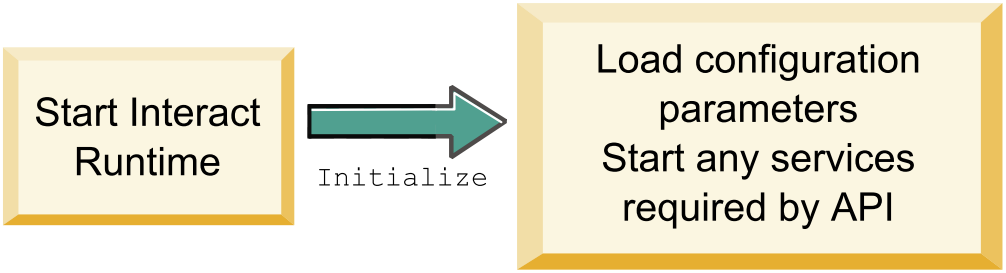
A API de aprendizado é construída em torno da interface `ILearning`. Você deve implementar a interface `ILearning` para suportar a lógica customizada de seu módulo de aprendizado.

Entre outras coisas, a interface `ILearning` permite coletar dados do ambiente de tempo de execução para sua classe Java e enviar uma lista de ofertas recomendadas novamente para o servidor de runtime.

initialize

O método `initialize` é chamado uma única vez quando o servidor de runtime é iniciado. Se houver operações que não precisem ser repetidas, mas que possam degradar desempenho durante o tempo de execução, como o carregamento de dados estáticos a partir de uma tabela de banco de dados, eles deverão ser executados por esse método.

```
initialize(ILearningConfig config, boolean debug)
```



- **config** - um objeto `ILearningConfig` define todas as propriedades de configuração relevantes ao aprendizado.
- **debug** - um booleano. Se for `true`, indicará o detalhamento do nível de criação de log para o ambiente de tempo de execução que o sistema está configurado para depurar. Para obter melhores resultados, selecione esse valor antes de gravar em um log.

Se o método `initialize` falhar por qualquer motivo, ele emitirá uma `LearningException`.

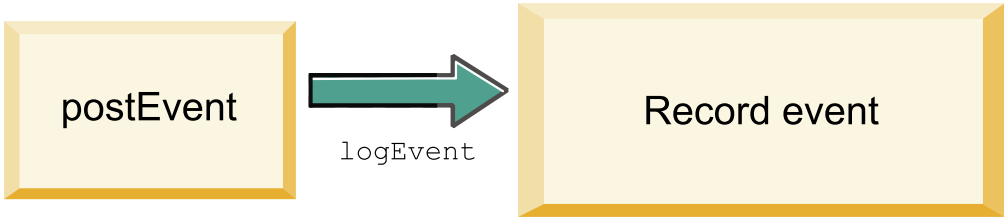
Valor de retorno

None.

logEvent

O método `logEvent` é chamado pleo servidor de runtime sempre que a API do Interact posta um evento configurado para registrar como um contato ou uma resposta. Use esse método para registrar dados de contatos e respostas em um banco de dados ou arquivo para propósito de relatório e aprendizado. Por exemplo, se você deseja usar algoritmos para determinar a probabilidade de um cliente de aceitar uma oferta com base em critérios, use esse método para registrar os dados.

```
logEvent(ILearningContext context,
             IOffer offer,
             IClientArgs clientArgs,
             IInteractSession session,
             boolean debug)
```



- **context** - objeto `ILearningContext` que define o contexto de aprendizado do evento, por exemplo, contato, aceitar ou rejeitar.
- **offer** - objeto `IOffer` que define o oferta sobre a qual este evento está sendo registrado.
- **clientArgs** - um objeto `IClientArgs` que define quaisquer parâmetros. Atualmente, o `logEvent` não requer nenhum `clientArgs`, portanto, esse parâmetro pode estar vazio.
- **session** - um objeto IInteractSession que define todos os dados da sessão.
- **debug** - um booleano. Se for `true`, indicará o detalhamento do nível de criação de log para o ambiente de tempo de execução que o sistema está configurado para depurar. Para obter melhores resultados, selecione esse valor antes de gravar em um log.

Se o método `AdministraçãoLog` falhar, ele emitirá uma `LearningException`.

Valor de retorno

None.

optimizeRecommendList

O método `optimizeRecommendList` deve obter a lista de ofertas recomendadas e os dados de sessão e retornar uma lista contendo o número solicitado de ofertas. O método `optimizeRecommendList` deve ordenar as ofertas de alguma maneira com seu próprio algoritmo de aprendizado. A lista de ofertas deve ser ordenada para que as ofertas que você deseja entregar primeiro fiquem no início da lista. Por exemplo, se o algoritmo de aprendizado fornecer uma pontuação baixa para as melhores ofertas, as ofertas deverão ser ordenadas como 1, 2, 3. Se o algoritmo de aprendizado fornecer uma pontuação alta para as melhores ofertas, elas deverão ser ordenadas como 100, 99, 98.

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

O método `optimizeRecommendList` requer os seguintes parâmetros:

- **recList** - uma lista dos objetos de tratamento (ofertas) recomendados pelo ambiente de tempo de execução.
- **clientArg** - um objeto IClientArgs contendo pelo menos o número de ofertas solicitado pelo ambiente de tempo de execução.
- **session** - an objeto IInteractSession que contém todos os dados da sessão.
- **debug** - um booleano. Se for `true`, indicará o detalhamento do nível de criação de log para o ambiente de tempo de execução que o sistema está configurado para depurar. Para obter melhores resultados, selecione esse valor antes de gravar em um log.

Se o método `optimizeRecommendList` falhar, ele lançará um `LearningException`.

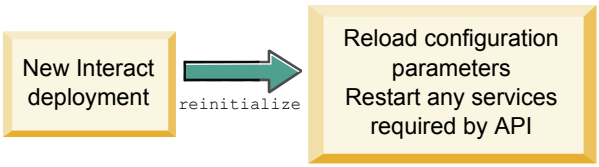
Valor de retorno

O método `optimizeRecommendList` retorna uma Lista dos objetos `ITreatment`.

reinitialize

O ambiente de tempo de execução chama o método `reinitialize` sempre que há uma nova implementação. Esse método transmite todos os dados de configuração de aprendizado. Se houver quaisquer serviços necessários para a API de aprendizado que leiam as propriedades de configuração, essa interface deverá reiniciá-los.

```
reinitialize(ILearningConfig config,
                 boolean debug)
```



- **config** - um objeto `ILearningConfig` que contém todas as propriedades de configuração.
- **debug** - um booleano. Se for `true`, indicará o detalhamento do nível de criação de log para o ambiente de tempo de execução que o sistema está configurado para depurar. Para obter melhores resultados, selecione esse valor antes de gravar em um log.

Se o método `AdministraçãoLog` falhar, ele emitirá uma `LearningException`.

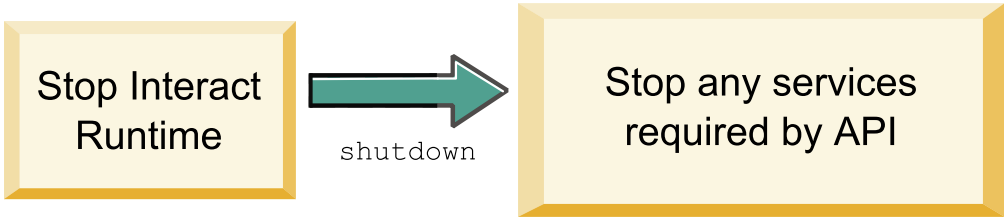
Valor de retorno

None.

shutdown

O ambiente de tempo de execução chama o método `shutdown` quando o servidor de runtime é encerrado. Se houver tarefas de limpeza necessárias para o módulo de aprendizado, elas deverão ser executadas neste momento.

```
shutdown(ILearningConfig config, boolean debug)
```



O método `shutdown` requer os seguintes parâmetros.

- **config** - um objeto `ILearningConfig` que define todas as propriedades de configuração.
- **debug** - um booleano. Se for `true`, indicará o detalhamento do nível de criação de log para o ambiente de tempo de execução que o sistema está configurado para depurar. Para obter melhores resultados, selecione esse valor antes de gravar em um log.

Se o método `shutdown` falhar por qualquer motivo, ele lançará uma `LearningException`.

Valor de retorno

None.

Interface IAudienceID

A interface `IAudienceID` suporta a interface `IInteractSession`. Esta é uma interface para o ID de público. Como seu ID de público pode ser feito de diversas partes, essa interface permite acessar todos os elementos do ID de público, bem como o nome do nível de público.

getAudienceLevel

O método `getAudienceLevel` retorna o nível de público.

```
getAudienceLevel()
```

Valor de retorno

O método `getAudienceLevel` retorna uma sequência que define o nível de público.

getComponentNames

O método `getComponentNames` obtém um conjunto dos nomes dos componentes que compõem o ID de público. Por exemplo, se seu ID de público consistir nos valores de `customerName` e `accountID`, `getComponentNames` retornariam um conjunto que contém as sequências customerName e accountID.

```
getComponentNames()
```

Valor de retorno

Um conjunto de sequências que contém os nomes dos componentes do ID de público.

getComponentValue

O método `getComponentValue` retorna o valor para o componente do nome.

```
getComponentValue(String componentName)
```

- **componentName**- uma sequência que define o nome do componente para o qual deseja recuperar o valor. Essa sequência não faz distinção entre maiúsculas e minúsculas.

Valor de retorno

O método `getComponentValue` retorna um objeto que define o valor do componente.

IClientArgs

A interface `IClientArgs` suporta a interface `ILearning`. Essa interface é uma abstração para cobrir quaisquer dados transmitidos para o servidor a partir do ponto de contato que ainda não foi abrangido pelos dados da sessão. Por exemplo, o número de ofertas solicitadas pela API Interact método `getOffers`. Esses dados são armazenados em um mapa.

getValue

O método `getValue` retorna o valor do elemento de mapa solicitado.

```
getValue(int clientArgKey)
```

Os elementos a seguir são necessários no mapa.

- **1** - NUMBER_OF_OFFERS_REQUESTED. O número de ofertas solicitadas pelo método `getOffers` da API do Interact. Esta constante retorna um número inteiro.

Valor de retorno

O método `getValue` retorna um objeto que define o valor da constante de mapa solicitada.

IInteractSession

A interface `IInteractSession` suporta a interface `ILearning`. Essa é uma interface para a sessão atual no ambiente de tempo de execução.

getAudienceId

O método `getAudienceId` retorna um objeto `AudienceID`. Use a interface `IAudienceID` para extrair os valores.

```
getAudienceId()
```

Valor de retorno

O método `getAudienceId` retorna um objeto `AudienceID`.

getSessionData

O método `getSessionData` retorna um mapa imodificável dos dados da sessão em que o nome da variável de sessão é a chave. O nome da variável de sessão é sempre maiúsculo. Use a interface `IInteractSessionData` para extrair valores.

```
getSessionData()
```

Valor de retorno

O método `getSessionData` retorna um objeto `IInteractSessionData`.

Interface IInteractSessionData

A interface `IInteractSessionData` suporta a interface `ILearning`. Essa é uma interface para os dados da sessão de tempo de execução para o visitante atual. Os dados da sessão são armazenados como uma lista de pares nome-valor. Também é possível usar essa interface para alterar o valor dos dados na sessão de tempo de execução.

getDataType

O método `getDataType` retorna o tipo de dados para o nome do parâmetro especificado.

```
getDataType(string parameterName)
```

Valor de retorno

O método `getDataType` retorna um objeto InteractDataType. IntearctDataType é uma enumeração Java representada por Desconhecido, Sequência, Duplo, Data ou Lista.

getParameterNames

O método `getParameterNames` retorna um conjunto de todos os nomes dos dados na sessão atual.

```
getParameterNames()
```

Valor de retorno

O método `getParameterNames` retorna um conjunto de nomes para os quais os valores foram configurados. Cada nome no conjunto pode ser passado para getValue(String) para retornar um valor.

getValue

O método `getValue` retorna o valor de objeto correspondente ao `parameterName` especificado. O objeto pode ser uma Sequência, Duplo ou Data.

```
getValue(parameterName)
```

O método `getValue` requer o parâmetro a seguir:

- **parameterName** - uma sequência que define o nome do par nome-valor de dados da sessão.

Valor de retorno

O método `getValue` retorna um objeto que contém o valor do parâmetro nomeado.

setValue

O método `setValue` permite configurar um valor para o `parameterName` especificado. O valor pode ser uma Sequência, Duplo ou Data.

```
setValue(string parameterName, object value)
```

O método `setValue` requer os parâmetros a seguir:

- **parameterName** - uma sequência que define o nome do par nome-valor de dados da sessão.
- **value** - um objeto que define o valor do parâmetro designado.

Valor de retorno

None.

ILearningAttribute

A interface `ILearningAttribute` suporta a interface `ILearningConfig`. Essa é uma interface para os atributos de aprendizado definidos nas propriedades de configuração na categoria `learningAttributes`.

getName

O método `getName` retorna o nome do atributo de aprendizado.

```
getName()
```

Valor de retorno

O método `getName` retorna uma sequência que define o nome do atributo de aprendizado.

ILearningConfig

A interface `ILearningConfig` suporta a interface `ILearning`. Essa é uma interface para as propriedades de configuração para aprendizado. Todos esses métodos retornam o valor da propriedade.

A interface consiste em 15 métodos:

- **getAdditionalParameters** - retorna um mapa de quaisquer propriedades adicionais definidas na categoria `Configuração de aprendizado externo`
- **getAggregateStatsIntervalInMinutes** - retorna um int
- **getConfidenceLevel** - retorna um int
- **getDataSourceName** - retorna uma sequência
- **getDataSourceType** - retorna uma sequência
- **getInsertRawStatsIntervalInMinutes** - retorna um int
- **getLearningAttributes** - retorna uma lista de objetos `ILearningAttribute`
- **getMaxAttributeNames** - retorna um int
- **getMaxAttributeValues** - retorna um int
- **getMinPresentCountThreshold** - retorna um int
- **getOtherAttributeValue** - retorna uma sequência
- **getPercentRandomSelection** - retorna um int
- **getRecencyWeightingFactor** - retorna um valor flutuante
- **getRecencyWeightingPeriod** - retorna um int
- **isPruningEnabled** - retorna um booleano

ILearningContext

A interface `ILearningContext` suporta a interface `ILearning`.

getLearningContext

O método `getLearningContext` retorna a constante que nos informa se é ou não um cenário de contato, aceitação ou rejeição.

```
getLearningContext()
```

- **1**-LOG_AS_CONTACT
- **2**-LOG_AS_ACCEPT
- **3**-LOG_AS_REJECT

4 e 5 são reservados para uso futuro.

Valor de retorno

O método `getLearningContext` retorna um número inteiro.

getResponseCode

O método `getResponseCode` retorna o código de resposta designado a esta oferta. Este valor deve existir na tabela `UA_UsrResponseType` nas tabelas de sistema do Campaign.

```
getResponseCode()
```

Valor de retorno

O método `getResponseCode` retorna uma sequência que define o código de resposta.

IOffer

A interface `IOffer` suporta a interface `ITreatment`. Essa é uma interface para o objeto de oferta definido no ambiente de design. Use a interface `IOffer` para coletar os detalhes da oferta do ambiente de tempo de execução.

getCreateDate

O método `getCreateDate` retorna a data em que a oferta foi criada.

```
getCreateDate()
```

Valor de retorno

O método `getCreateDate` retorna uma data que define a data em que a oferta foi criada.

getEffectiveDateFlag

O método `getEffectiveDateFlag` retorna um número que define a data efetiva da oferta.

```
getEffectiveDateFlag()
```

- **0**- a data efetiva é uma data absoluta, como 15 de março de 2010.
- **1**- a data efetiva é a data de recomendação.

Valor de retorno

O método `getEffectiveDateFlag` retorna um número inteiro que define a data efetiva da oferta.

getExpirationDateFlag

O método `getExpirationDateFlag` retorna um valor de número inteiro que descreve a data de expiração da oferta.

```
getExpirationDateFlag()
```

- **0**- uma data absoluta, por exemplo, 15 de março de 2010.
- **1**- algum número de dias após a recomendação, por exemplo, 14.
- **2**- fim do mês após a recomendação. Se uma oferta for apresentada em 31 de março, a oferta expirará esse dia.

Valor de retorno

O método `getExpirationDateFlag` retorna um número inteiro que descreve a data de expiração da oferta.

getOfferAttributes

O método `getOfferAttributes` retorna atributos de oferta definidos para a oferta como um objeto `IOfferAttributes`.

```
getOfferAttributes()
```

Valor de retorno

O método `getOfferAttributes` retorna um objeto `IOfferAttributes`.

getOfferCode

O método `getOfferCode` retorna o código de oferta da oferta conforme definido no Campaign.

```
getOfferCode()
```

Valor de retorno

O método `getOfferCode` retorna um objeto `IOfferCode`.

getOfferDescription

O método `getOfferDescription` retorna a descrição da oferta definida em Campaign.

```
getOfferDescription()
```

Valor de retorno

O método `getOfferDescription` retorna uma sequência.

getOfferID

O método `getOfferID` retorna o ID da oferta conforme definido em Campaign.

```
getOfferID()
```

Valor de retorno

O método `getOfferID` retorna um item longo que define o ID da oferta.

getOfferName

O método getOfferName retorna o nome da oferta conforme definido no Campaign.

```
getOfferName()
```

Valor de retorno

O método getOfferName retorna uma sequência.

getUpdateDate

O método getUpdateDate retorna uma data de quando a oferta foi atualizada pela última vez.

```
getUpdateDate()
```

Valor de retorno

O método getUpdateDate retorna uma data que define quando a oferta foi atualizada pela última vez.

IOfferAttributes

A interface IOfferAttributes suporta a interface IOffer. Essa é uma interface para os atributos de oferta definidos para uma oferta no ambiente de design. Use a interface IOfferAttributes para coletar os atributos de oferta do ambiente de tempo de execução.

getParameterNames

O método getParameterNames retorna uma lista dos nomes de parâmetro de oferta.

```
getParameterNames()
```

Valor de retorno

O método getParameterNames retorna um conjunto que define a lista de nomes de parâmetro de oferta.

getValue

O método getValue retorna um objeto que define o valor do atributo de oferta.

```
getValue(String parameterName)
```

O método getValue retorna um valor do determinado atributo de oferta.

Valor de retorno

Interface IOfferCode

A interface IOfferCode suporta a interface ILearning. Essa é uma interface para o código de oferta que foi definido para uma oferta no ambiente de design. Um código de oferta pode ser feito de uma ou várias Sequências. Use a interface IOfferCode para coletar o código de oferta do ambiente de tempo de execução.

getPartCount

O método getPartCount retorna o número de partes que compõem um código de oferta.

getPartCount()

Valor de retorno

O método getPartCount retorna um número inteiro que define o número de partes do código de oferta.

getParts

O método getParts obtém uma lista não modificável das partes do código de oferta.

getParts()

Valor de retorno

O método getParts retorna uma lista não modificável das partes do código de oferta.

LearningException

A classe LearningException suporta a interface ILearning. Alguns métodos na interface irão requerer implementações para lançar um LearningException que é uma subclasse simples de java.lang.Exception. É altamente recomendado para fins de depuração que LearningException seja construída com a exceção raiz se uma exceção raiz existir.

IScoreOverride

A interface IScoreOverride suporta a interface ITreatment. Essa interface permite ler os dados definidos na substituição de pontuação ou tabela de ofertas padrão.

getOfferCode

O método getOfferCode retorna o valor das colunas de código de oferta na tabela de substituição de pontuação para este membro do público.

getOfferCode()

Valor de retorno

O método getOfferCode retorna um objeto IOfferCode que define o valor das colunas de código de oferta na tabela de substituição de pontuação.

getParameterNames

O método getParameterNames retorna a lista de parâmetros.

getParameterNames()

Valor de retorno

O método getParameterNames retorna um conjunto que define a lista de parâmetros.

O método IScoreOverride contém os parâmetros a seguir. A menos que indicado de outra forma, esses parâmetros são os mesmos que a tabela de substituição de pontuação.

- ADJ_EXPLORE_SCORE_COLUMN

- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT - Para substituir a contagem presente estimada (durante o cálculo de peso da oferta)
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN - Para criar uma expressão booleana para determinar a elegibilidade da oferta
- PREDICATE_SCORE - Para criar uma expressão que resulta em uma pontuação numérica
- SCORE_COLUMN
- ZONE_COLUMN

Também é possível referenciar qualquer coluna que incluir na substituição de pontuação ou tabela de ofertas padrão usando o mesmo nome que a coluna.

getValue

O método `getValue` retorna o valor da coluna zona na tabela de substituição de pontuação para este membro do público.

`getValue(String parameterName)`

- **parameterName**- uma sequência que define o nome do parâmetro para o qual deseja o valor.

Valor de retorno

O método `getValue` retorna um objeto que define o valor do parâmetro solicitado.

ISelectionMethod

A interface `ISelection` indica o método usado para produzir a lista recomendada. O valor padrão para o objeto Treatment é EXTERNAL_LEARNING, portanto, não é necessário configurar esse valor. O valor é finalmente armazenado no Histórico de contato detalhado para fins de relatório.

É possível estender esta interface além das constantes existentes se você desejar armazenar os dados para análise posterior. Por exemplo, você poderia criar dois módulos de aprendizado diferentes e implementá-los em grupos de servidores separados. Você poderia estender a interface `ISelection` para incluir SERVER_GROUP_1 e SERVER_GROUP_2. Você poderia comparar os resultados de seus dois módulos de aprendizado.

Interface ITreatment

A interface `ITreatment` suporta a interface `ILearning` como uma interface para as informações do Tratamento. Um tratamento representa a oferta designada a uma determinada célula conforme definido no ambiente de design. A partir dessa interface, é possível obter informações de célula e oferta, bem como a pontuação de marketing designada.

getCellCode

O método `getCellCode` retorna o código de célula conforme definido no Campaign. A célula é a célula designada ao segmento inteligente associado a esta oferta.

```
getCellCode()
```

Valor de retorno

O método `getCellCode` retorna uma sequência que define o código de célula.

getCellId

O método `getCellId` retorna o ID interno da célula conforme definido no Campaign. A célula é a célula designada ao segmento inteligente associado a esta oferta.

```
getOfferName()
```

Valor de retorno

O método `getCellId` retorna um item longo que define o ID de célula.

getCellName

O método `getCellName` retorna o nome da célula conforme definido no Campaign. A célula é a célula designada ao segmento inteligente associado a esta oferta.

```
getCellName()
```

Valor de retorno

O método `getCellName` retorna uma sequência que define o nome da célula.

getLearningScore

O método `getLearningScore` retorna a pontuação para este tratamento.

```
getLearningScore()
```

A precedência é como a seguir.

1. Retornar o valor de substituição, se presente no mapa Substituir valores entre chaves por `IScoreoveride.PREDICATE_SCORE_COLUMN`
2. Retornar a pontuação prevista se o valor não for nulo
3. Retornar a pontuação de comerciante, se presente no mapa Substituir valores entre chaves por `IScoreoveride.SCORE`
4. Retornar uma pontuação de comerciante

Valor de retorno

O método `getLearningScore` retorna um número inteiro que define a pontuação determinada pelo algoritmo de aprendizado.

getMarketerScore

O método `getMarketerScore` retorna a pontuação do comerciante definida pela régua de controle na guia Estratégia de interação para a oferta.

```
getMarketerScore()
```

Para recuperar a pontuação de um comerciante definida pelas opções avançadas da guia Estratégia de interação, use `getPredicateScore`.

Para recuperar a pontuação do comerciante realmente usada pelo tratamento, use `getLearningScore`.

Valor de retorno

O método `getMarketerScore` retorna um número inteiro que define a pontuação do comerciante.

getOffer

O método `getOffer` retorna a oferta para o tratamento.

```
getOffer()
```

Valor de retorno

O método `getOffer` retorna um objeto `IOffer` que define a oferta para este tratamento.

getOverrideValues

O método `getOverrideValues` retorna substituições definidas nas ofertas padrão ou tabela de substituição de pontuação.

```
getOverrideValues()
```

Valor de retorno

O método `getOverrideValues` retorna um objeto `IScoreOverride`.

getPredicate

O método `getPredicate` retorna o predicado definido pela coluna do predicado da tabela de ofertas padrão, tabela de substituição de pontuação ou opções avançadas de regras de tratamento.

```
getPredicate()
```

Valor de retorno

O método `getPredicate` retorna uma sequência que define o predicado definido pela coluna do predicado da tabela de ofertas padrão, tabela de substituição de pontuação ou opções avançadas de regras de tratamento.

getPredicateScore

O método `getPredicateScore` retorna a pontuação configurada pela coluna do predicado da tabela de ofertas padrão, tabela de substituição de pontuação ou as opções avançadas das regras de tratamento.

```
getPredicateScore()
```

Valor de retorno

O método `getPredicateScore` retorna um dupl que define a pontuação configurada pela coluna de predicados da tabela de ofertas padrão, tabela de substituição de pontuação ou as opções avançadas das regras de tratamento.

getScore

O método `getScore` retorna a pontuação de marketing que é definida pela estratégia de interação no Campaign ou pela tabela de substituição de pontuação.

```
getScore()
```

O método `getScore` retorna um dos seguintes:

- A pontuação de marketing da oferta conforme definido na guia Estratégia de interação no Campaign se a propriedade `enableScoreOverrideLookup` estiver configurada para false.
- A pontuação da oferta conforme definido pelo `scoreOverrideTable` se a propriedade `enableScoreOverrideLookup` estiver configurada para true.

Valor de retorno

O método `getScore` retorna um número inteiro que representa a pontuação da oferta.

getTreatmentCode

O método `getTreatmentCode` retorna o código de tratamento.

```
getTreatmentCode()
```

Valor de retorno

O método `getTreatmentCode` retorna uma sequência que define o código de tratamento.

setActualValueUsed

Use o método `setActualValueUsed` para definir que valores são usados em vários estágios na execução do algoritmo de aprendizado.

```
setActualValueUsed(string parmName, object value)
```

Por exemplo, se você usar este método para gravar nas tabelas de histórico de resposta e contato e modificar os relatórios de amostra existentes, é possível incluir dados de seu algoritmo de aprendizado no relatório.

- **parmName**- uma sequência que define o nome do parâmetro que você está configurando.
- **value**- um objeto que define o valor do parâmetro que você está configurando.

Valor de retorno

None.

Exemplo de API de aprendizado

Esta seção contém uma implementação de amostra de ILearningInterface. Observe que essa implementação é apenas uma amostra e não foi projetada para ser usada em um ambiente de produção.

Este exempl mantém o controle de contagens de aceitação e contato e usa a proporção de aceitação para contatos para uma determinada oferta como a taxa de probabilidade de aceitação para a oferta. Ofertas não apresentadas obtêm maior prioridade para recomendação. Ofertas com pelo menos um contato serão ordenadas com base na taxa de probabilidade de aceitação decrescente.

Neste exemplo, todas as contagens serão mantidas na memória. Este não é um cenário realista, pois o servidor de runtime ficará sem memória. Em um cenário de produção real, as contagens devem persistir em um banco de dados.

```
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;

/**
 * Esta é uma implementação de amostra do otimizador de aprendizado.
 * A interface ILearning pode ser localizada na biblioteca interact.jar.
 *
 * Para realmente usar esta implementação, selecione ExternalLearning como o optimizationType no nó offerServing
 * do aplicativo Interact na configuração da plataforma.  No nó offerserving, também existe uma
 * categoria de configuração Aprendizado externo - nela, você deve configurar o nome da classe para este:
 * com.unicacorp.interact.samples.learning.v2.SampleLearning.  Observe, no entanto, que esta implementação é apenas uma amostra
 * e não foi projetada para ser usada em um ambiente de produção.
 *
 *
 * Este exemplo mantém o controle de contagens de aceitação e contato e usa a proporção de aceitação para contatos
 * para uma determinada oferta, como a taxa de probabilidade de aceitação para a oferta.
 *
 *
 * Ofertas não apresentadas obterão maior prioridade para
recomendação.
 * Ofertas com pelo menos um contato serão ordenadas com base na
taxa de probabilidade de aceitação decrescente.
 *
 * Nota: todas as contagens são mantidas na memória.  Este não é um
cenário realista, pois você ficaria sem memória mais cedo ou mais
 * tarde.  Em
um cenário de produção real, as contagens devem persistir em um banco de dados.
 *
 */

public class SampleLearning implements ILearning
{

    // A map of offer ids to contact count for the offer id
    private Map<Long,Integer> _offerToContactCount = new HashMap<Long, Integer>();

    // A map of offer ids to contact count for the offer id
    private Map<Long,Integer> _offerToAcceptCount = new HashMap<Long, Integer>();


    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#initialize
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void initialize(ILearningConfig config, boolean debug) throws LearningException
    {
        // Se alguma conexão remota for necessária, este será um bom
lugar para inicializar essas conexões, pois este
        // método é chamado uma vez no início do aplicativo da
web de tempo de execução do Interact.
        // Este exemplo não tem conexões remotas e impressões para
fins de depuração que este método será
        // chamado
        System.out.println("Calling initialize for SampleLearning");
```

```
    }

    /* (não 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
    {
        // Se um IC for implementado, este método de reinicialização
será chamado para permitir que a implementação atualize
        // quaisquer definições de configuração atualizadas
        System.out.println("Calling reinitialize for SampleLearning");
    }


    /* (não Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#logEvent
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningContext,
     *   com.unicacorp.interact.treatment.optimization.v2.IOffer,
     *   com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
     *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
     */
    public void logEvent(ILearningContext context, IOffer offer, IClientArgs clientArgs,
   IInteractSession session, boolean debug) throws LearningException
    {
        System.out.println("Calling logEvent for SampleLearning");


        if(context.getLearningContext()==ILearningContext.LOG_AS_CONTACT)
        {
            System.out.println("adding contact");

            // Keep track of all contacts in memory
            synchronized(_offerToAcceptCount)
            {
               Integer count = _offerToAcceptCount.get(offer.getOfferId());
               if(count == null)
                   count = new Integer(1);
               else
                   count++;
               _offerToAcceptCount.put(offer.getOfferId(), ++count);
            }

        }
        else if(context.getLearningContext()==ILearningContext.LOG_AS_ACCEPT)
        {
            System.out.println("adding accept");
            // Mantenha controle de todas as contagens de aceitação
na memória incluindo no mapa
            synchronized(_offerToAcceptCount)
            {
               Integer count = _offerToAcceptCount.get(offer.getOfferId());
               if(count == null)
                   count = new Integer(1);
               else
                   count++;
               _offerToAcceptCount.put(offer.getOfferId(), ++count);
            }
            }

    }

    /* (não Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#optimizeRecommendList
     *   (java.util.List, com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
     *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
     */
    public List<ITreatment> optimizeRecommendList(List<ITreatment> recList,
        IClientArgs clientArgs, IInteractSession session, boolean debug)
        throws LearningException
```

```
    {
        System.out.println("Calling optimizeRecommendList for SampleLearning");

        // Classifique os tratamentos de candidato chamando o
classificador definido nesta classe e retorne a lista classificada
        Collections.sort(recList,new MyOfferSorter());

        // agora apenas retorne o que foi solicitado pela variável "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;
    }

    /* (não 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
    {
        // Se quaisquer conexões remotas existirem, seria um bom
lugar para
        // desconectar graciosamente delas, pois este método é
chamado no encerramento do aplicativo da web de tempo de execução
        // do Interact.   Para este exemplo, não há nada realmente
a se fazer,
        // exceto imprimir uma instrução para depuração.
        System.out.println("Calling shutdown for SampleLearning");

    }
    // Classificar por:
    // 1. ofertas com contatos zero - para camadas, a ordem é
baseda na entrada original
    // 2. taxa de probabilidade de aceitação decrescente -
para camadas, a ordem é baseada na entrada original

    public class MyOfferSorter implements Comparator<ITreatment>
    {
        private static final long serialVersionUID = 1L;

        /* (não Javadoc)
         * @see java.lang.Comparable#compareTo(java.lang.Object)
         */
        public int compare(ITreatment treatment1, ITreatment treatment2)
        {

            // obter contagem de contato para ambos os tratamentos
            Integer contactCount1 = _offerToContactCount.get(treatment1.getOffer().getOfferId());
            Integer contactCount2 = _offerToContactCount.get(treatment2.getOffer().getOfferId());

            // se o tratamento não tiver sido contactado, ele vencerá
            if(contactCount1 == null || contactCount1 == 0)
                return -1;

            if(contactCount2 == null || contactCount2 == 0)
                return 1;

            // obter contagens de aceitação
            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;

            // ordem decrescente
            return (int) (acceptProbability2 - acceptProbability1);
```

```
            }
        }
    }
```

Apêndice A. WSDL do IBM Interact

A instalação do Interact inclui dois arquivos XML WDSL (Web Services Description Language) que descrevem os serviços da web disponíveis e como acessá-los. É possível visualizar esses arquivos no diretório inicial do Interact e um exemplo é mostrado aqui.

Depois de instalar o Interact, será possível localizar os arquivos WSDL do Interact no seguinte local:

- <*Interact_home*>/conf/InteractService.wsdl
- <*Interact_home*>/conf/InteractAdminService.wsdl

Com cada liberação de software ou fix pack, poderá haver mudanças no WDSL do Interact. Consulte as *Notas sobre a liberação do Interact* ou os arquivos leia-me com a liberação para obter detalhes.

Uma cópia do `InteractService.wsdl` é mostrada aqui para referência. Para assegurar-se de que você esteja usando as informações mais recentes, consulte os arquivos WSDL instalados com o 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>
```

Apêndice B. Propriedades de configuração do ambiente de tempo de execução do Interact

Esta seção descreve todas as propriedades de configuração para o ambiente de tempo de execução do Interact.

Interact | geral

Estas propriedades de configuração definem as configurações gerais para o ambiente de tempo de execução, incluindo o nível de criação de log padrão e a configuração do código de idioma.

log4jConfig

Descrição

A localização do arquivo contendo as propriedades log4j. Esse caminho deve ser relativo à variável de ambiente `INTERACT_HOME`. `INTERACT_HOME` é a localização do diretório de instalação do Interact.

Valor padrão

`./conf/interact_log4j.properties`

asmUserForDefaultLocale

Descrição

A propriedade `asmUserForDefaultLocale` define o usuário do IBM EMM a partir do qual o Interact deriva suas configurações do código de idioma.

As configurações do código de idioma definem que idioma é exibido no tempo de design e em qual idioma estão as mensagens de recomendação da API do Interact. Se as configurações do código de idioma não corresponderem às configurações do sistema operacional das máquinas, o Interact continuará a funcionar, no entanto, a exibição do tempo de design e as mensagens de recomendação poderão aparecer em um idioma diferente.

Valor padrão

`asm_admin`

Interact | geral | learningTablesDataSource

Estas propriedades de configuração definem as configurações de origens de dados para as tabelas de aprendizado integrado. Você deverá definir essas origens de dados se estiver usando o aprendizado integrado do Interact.

Se você criar sua própria implementação de aprendizado usando a API de Aprendizado, será possível configurar a implementação de aprendizado customizada para ler esses valores usando a interface `ILearningConfig`.

jndiName

Descrição

Use esta propriedade jndiName para identificar a origem de dados Java Naming and Directory Interface (JNDI) definida no servidor de aplicativos (Websphere ou WebLogic) para as tabelas de aprendizado acessadas pelos servidores de runtime do Interact.

As tabelas de aprendizado são criadas pelo arquivo DDL aci_lrntab e contêm as seguintes tabelas (entre outras): UACI_AttributeValue e UACI_OfferStats.

Valor padrão

Nenhum valor padrão definido.

type

Descrição

O tipo de banco de dados para a origem de dados usada pelas tabelas de aprendizado acessadas pelos servidores de runtime do Interact.

As tabelas de aprendizado são criadas pelo arquivo DDL aci_lrntab e contêm as seguintes tabelas (entre outras): UACI_AttributeValue e UACI_OfferStats.

Valor padrão

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descrição

A propriedade ConnectionRetryPeriod especifica o tempo em segundos em que o Interact tenta novamente automaticamente a solicitação de conexão com o banco de dados no caso de falha, para as tabelas de aprendizado. O Interact tenta reconectar-se automaticamente ao banco de dados durante esse período de tempo antes de relatar um erro ou uma falha do banco de dados. Se o valor for configurado como 0, o Interact tentará novamente e indefinidamente. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

As tabelas de aprendizado são criadas pelo arquivo DDL aci_lrntab e contêm as seguintes tabelas (entre outras): UACI_AttributeValue e UACI_OfferStats.

Valor padrão

-1

connectionRetryDelay

Descrição

A propriedade ConnectionRetryDelay especifica o tempo em segundos que o Interact aguarda antes de tentar se reconectar ao banco de dados após uma falha para as tabelas de aprendizado. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

As tabelas de aprendizado são criadas pelo arquivo DDL aci_lrntab e contêm as seguintes tabelas (entre outras): UACI_AttributeValue e UACI_OfferStats.

Valor padrão

-1

esquema

Descrição

O nome do esquema contendo as tabelas para o módulo de aprendizado integrado. O Interact insere o valor dessa propriedade antes de todos os nomes de tabela, por exemplo, `UACI_IntChannel` torna-se `schema.UACI_IntChannel`.

Não é necessário definir um esquema. Se você não definir um esquema, o Interact assumirá que o proprietário das tabelas é o mesmo que o do esquema. Você deve configurar esse valor para remover a ambiguidade.

Valor padrão

Nenhum valor padrão definido.

Interact | geral | prodUserDataSource

Essas propriedades de configuração definem as configurações de origem de dados para as tabelas de perfis de produção. Você deve definir essa origem de dados. Essa é a origem de dados que o ambiente de tempo de execução referencia ao executar fluxogramas interativos após a implementação.

jndiName

Descrição

Use esta propriedade `jndiName` para identificar a origem de dados Java Naming and Directory Interface (JNDI) que é definida no servidor de aplicativos (Websphere ou WebLogic) para as tabelas do cliente acessadas pelos servidores de runtime do Interact.

Valor padrão

Nenhum valor padrão definido.

type

Descrição

O tipo de banco de dados para as tabelas do cliente acessadas pelos servidores de runtime do Interact.

Valor padrão

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

aliasPrefix

Descrição

A propriedade `AliasPrefix` especifica a forma como o Interact forma o nome alternativo que o Interact cria automaticamente ao usar uma tabela de dimensões e gravar em uma nova tabela nas tabelas do cliente acessadas pelos servidores de runtime do Interact.

Observe que cada banco de dados tem um comprimento máximo do identificador. Verifique a documentação do banco de dados que está sendo usada para assegurar-se de que o valor configurado não exceda ao comprimento máximo do identificador para o banco de dados.

Valor padrão

A

connectionRetryPeriod

Descrição

A propriedade `ConnectionRetryPeriod` especifica a quantia de tempo, em segundos, em que o Interact tenta automaticamente novamente a solicitação de conexão com o banco de dados no caso de falha para as tabelas do cliente de tempo de execução. O Interact tenta reconectar-se automaticamente ao banco de dados durante esse período de tempo antes de relatar um erro ou uma falha do banco de dados. Se o valor for configurado como 0, o Interact tentará novamente e indefinidamente. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

Valor padrão

-1

connectionRetryDelay

Descrição

A propriedade `ConnectionRetryDelay` especifica o tempo em segundos que o Interact aguarda antes de tentar se reconectar ao banco de dados após uma falha nas tabelas do cliente de tempo de execução do Interact. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

Valor padrão

-1

esquema

Descrição

O nome do esquema contendo as suas tabelas de dados de perfil. O Interact insere o valor dessa propriedade antes de todos os nomes de tabela, por exemplo, `UACI_IntChannel` torna-se `schema.UACI_IntChannel`.

Não é necessário definir um esquema. Se você não definir um esquema, o Interact assumirá que o proprietário das tabelas é o mesmo que o do esquema. Você deve configurar esse valor para remover a ambiguidade.

Ao usar um banco de dados DB2, o nome do esquema deve estar em letras maiúsculas.

Valor padrão

Nenhum valor padrão definido.

Interact | geral | systemTablesDataSource

Estas propriedades de configuração definem as configurações de origens de dados para as tabelas de sistema do ambiente de tempo de execução. Você deve definir essa origem de dados.

jndiName

Descrição

Use esta propriedade jndiName para identificar a origem de dados Java Naming and Directory Interface (JNDI) definida no servidor de aplicativos (Websphere ou WebLogic) para as tabelas do ambiente de tempo de execução.

O banco de dados do ambiente de tempo de execução é o banco de dados preenchido com os scripts DLL aci_runtime e aci_populate_runtime e, por exemplo, contém as tabelas a seguir (entre outras): UACI_CHOfferAttrib e UACI_DefaultedStat.

Valor padrão

Nenhum valor padrão definido.

type

Descrição

O tipo de banco de dados para as tabelas de sistema do ambiente de tempo de execução.

O banco de dados do ambiente de tempo de execução é o banco de dados preenchido com os scripts DLL aci_runtime e aci_populate_runtime e, por exemplo, contém as tabelas a seguir (entre outras): UACI_CHOfferAttrib e UACI_DefaultedStat.

Valor padrão

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descrição

A propriedade ConnectionRetryPeriod especifica o período de tempo, em segundos, em que o Interact tenta novamente automaticamente a solicitação de conexão com o banco de dados no caso de falha para as tabelas do sistema de Runtime. O Interact tenta reconectar-se automaticamente ao banco de dados durante esse período de tempo antes de relatar um erro ou uma falha do banco de dados. Se o valor for configurado como 0, o Interact tentará novamente e indefinidamente. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

O banco de dados do ambiente de tempo de execução é o banco de dados preenchido com os scripts DLL aci_runtime e aci_populate_runtime e, por exemplo, contém as tabelas a seguir (entre outras): UACI_CHOfferAttrib e UACI_DefaultedStat.

Valor padrão

-1

connectionRetryDelay

Descrição

A propriedade ConnectionRetryDelay especifica o período de tempo, em segundos, que o Interact aguarda antes de tentar se reconectar ao banco de dados após uma falha para as tabelas do sistema de Runtime do Interact. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

O banco de dados do ambiente de tempo de execução é o banco de dados preenchido com os scripts DLL aci_runtime e aci_populate_runtime e, por exemplo, contém as tabelas a seguir (entre outras): UACI_CHOfferAttrib e UACI_DefaultedStat.

Valor padrão

-1

esquema

Descrição

O nome do esquema contendo as tabelas para o ambiente de tempo de execução. O Interact insere o valor dessa propriedade antes de todos os nomes de tabela, por exemplo, UACI_IntChannel torna-se schema.UACI_IntChannel.

Não é necessário definir um esquema. Se você não definir um esquema, o Interact assumirá que o proprietário das tabelas é o mesmo que o do esquema. Você deve configurar esse valor para remover a ambiguidade.

Valor padrão

Nenhum valor padrão definido.

Interact | geral | systemTablesDataSource | loaderProperties

Estas propriedades de configuração definem as configurações de um utilitário de carregador de banco de dados para as tabelas de sistema do ambiente de tempo de execução. Será necessário definir essas propriedades se você somente estiver usando um utilitário de carregador de banco de dados.

databaseName

Descrição

O nome do banco de dados ao qual o carregador do banco de dados se conecta.

Valor padrão

Nenhum valor padrão definido.

LoaderCommandForAppend

Descrição

O parâmetro LoaderCommandForAppend especifica o comando emitido para chamar o utilitário de carregamento do banco de dados para anexação de registros nas tabelas do banco de dados de migração de histórico de contatos e respostas no Interact. É necessário configurar esse parâmetro para ativar o utilitário de carregador de banco de dados para os dados do histórico de contatos e respostas.

Esse parâmetro é especificado como um nome de caminho completo para o executável do utilitário de carregamento do banco de dados ou para um script que ative o utilitário de carregamento do banco de dados. O uso de um script permite executar uma configuração adicional antes de chamar o utilitário de carregamento.

A maioria dos utilitários de carregamento de banco de dados requer diversos argumentos para ser ativada com sucesso. Esses argumentos podem incluir a especificação do arquivo de dados e do arquivo de controle a partir dos quais carregar e o banco de dados e a tabela nos quais carregar. Os tokens são substituídos pelos elementos especificados quando o comando é executado.

Consulte a documentação do utilitário de carregamento de banco de dados para verificar a sintaxe correta a ser usada ao chamar o utilitário de carregamento de banco de dados.

Este parâmetro é indefinido, por padrão.

Os tokens disponíveis para `LoaderCommandForAppend` são descritos na tabela a seguir.

Token	Descrição
<CONTROLFILE>	Este token é substituído pelo caminho completo e o nome do arquivo para o arquivo de controle temporário que o Interact gera de acordo com o modelo especificado no parâmetro `LoaderControlFileTemplate`.
<DATABASE>	Este token é substituído pelo nome da origem de dados na qual o Interact está carregando dados. Esse é o mesmo nome de origem de dados usado no nome da categoria para essa origem de dados.
<DATAFILE>	Este token é substituído pelo caminho completo e o nome do arquivo para o arquivo de dados temporários criado pelo Interact durante o processo de carregamento. Esse arquivo está no diretório temporário Interact, `UNICA_ACTMPDIR`.
<DBCOLUMNNUMBER>	Este token é substituído pelo ordinal da coluna no banco de dados.
<FIELDLENGTH>	Este token é substituído pelo comprimento do campo que está sendo carregado no banco de dados.
<FIELDNAME>	Este token é substituído pelo nome do campo que está sendo carregado no banco de dados.
<FIELDNUMBER>	Este token é substituído pelo número do campo que está sendo carregado no banco de dados.
<FIELDTYPE>	Este token é substituído pelo literal "CHAR( )". O comprimento desse campo é especificado entre (). Se o banco de dados não entender o tipo de campo CHAR, será possível especificar manualmente o texto adequado para o tipo de campo e usar o token <FIELDLENGTH>. Por exemplo, para SQLSVR e SQL2000 deve ser usado "SQLCHAR(<FIELDLENGTH>)"
<NATIVETYPE>	Esta token é substituído pelo tipo de banco de dados no qual esse campo é carregado.
<NUMFIELDS>	Este token é substituído pelo número de campos na tabela.
<PASSWORD>	Este token é substituído pela senha do banco de dados a partir da conexão do fluxograma atual com a origem de dados.
<TABLENAME>	Este token é substituído pelo nome da tabela de banco de dados na qual o Interact está carregando dados.
<USER>	Este token é substituído pelo usuário do banco de dados a partir da conexão atual do fluxograma com a origem de dados.

Valor padrão

Nenhum valor padrão definido.

LoaderControlFileTemplateForAppend

Descrição

A propriedade LoaderControlFileTemplateForAppend especifica o caminho completo e o nome do arquivo para o modelo de arquivo de controle que foi configurado anteriormente no Interact. Quando esse parâmetro é configurado, o Interact constrói dinamicamente um arquivo de controle temporário com base no modelo especificado aqui. O caminho e o nome desse arquivo de controle temporário estão disponíveis para o token <CONTROLFILE> que está disponível para a propriedade LoaderCommandForAppend.

Antes de usar o Interact no modo do utilitário do carregador do banco de dados, você deve configurar o modelo de arquivo de controle especificado por esse parâmetro. O modelo do arquivo de controle suporta os tokens a seguir, que são substituídos dinamicamente quando o arquivo de controle temporário é criado pelo Interact.

Consulte a documentação do utilitário de carregador de banco de dados para obter a sintaxe correta para o arquivo de controle. Os tokens disponíveis para o modelo de arquivo de controle são os mesmos que os da propriedade LoaderControlFileTemplate.

Este parâmetro é indefinido, por padrão.

Valor padrão

Nenhum valor padrão definido.

LoaderDelimiterForAppend

Descrição

A propriedade LoaderDelimiterForAppend especifica se o arquivo de dados temporários do Interact é um arquivo simples de largura fixa ou delimitado e, se for delimitado, o caractere ou conjunto de caracteres usado como delimitador.

Se o valor for indefinido, o Interact criará o arquivo de dados temporários como um arquivo simples de largura fixa.

Se você especificar um valor, ele será usado quando o carregador for chamado para preencher uma tabela que não se sabe se está vazia. O Interact cria o arquivo de dados temporários como um arquivo simples delimitado, usando o valor dessa propriedade como o delimitador.

Essa propriedade é indefinida, por padrão.

Valor padrão

Valores válidos

Caracteres, que podem estar entre aspas duplas, se desejado.

LoaderDelimiterAtEndForAppend

Descrição

Alguns utilitários de carregamento externos requerem que o arquivo de dados seja delimitado e que cada linha termine com o delimitador. Para cumprir esse requisito, configure o valor de LoaderDelimiterAtEndForAppend como TRUE, para que quando o carregador

for chamado para preencher uma tabela que não se sabe se está vazia, o Interact use delimitadores no final de cada linha.

Valor padrão

FALSE

Valores válidos

TRUE | FALSE

LoaderUseLocaleDP

Descrição

A propriedade LoaderUseLocaleDP especifica quando o Interact grava valores numéricos nos arquivos a serem carregados por um utilitário de carregamento do banco de dados, se o símbolo específico do código de idioma for usado como ponto decimal.

Configure esse valor como FALSE para especificar que o ponto (.) é usado como ponto decimal.

Configure esse valor como TRUE para especificar que o símbolo de ponto decimal adequado ao código de idioma seja usado.

Valor padrão

FALSE

Valores válidos

TRUE | FALSE

Interação | geral | testRunDataSource

Essas propriedades de configuração definem as configurações de origem de dados das tabelas de execução de teste para o ambiente de design do Interact. Você deve definir essa origem de dados para no mínimo um dos ambientes de tempo de execução. Essas são as tabelas usadas ao realizar uma execução de teste do fluxograma interativo.

jndiName

Descrição

Use esta propriedade jndiName para identificar a origem de dados Java Naming and Directory Interface (JNDI) definida no servidor de aplicativos (Websphere ou WebLogic) para as tabelas do cliente acessadas pelo ambiente de design ao realizar execuções de teste de fluxogramas interativos.

Valor padrão

Nenhum valor padrão definido.

type

Descrição

O tipo de banco de dados para as tabelas do cliente acessadas pelo ambiente de design ao realizar execuções de teste de fluxogramas interativos.

Valor padrão

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

aliasPrefix

Descrição

A propriedade `AliasPrefix` especifica a maneira que o Interact forma o nome alternativo que o Interact cria automaticamente ao usar uma tabela de dimensões e gravar em uma nova tabela nas tabelas do cliente acessadas pelo ambiente de design ao realizar execuções de teste de fluxogramas interativos.

Observe que cada banco de dados tem um comprimento máximo do identificador. Verifique a documentação do banco de dados que está sendo usada para assegurar-se de que o valor configurado não exceda ao comprimento máximo do identificador para o banco de dados.

Valor padrão

A

connectionRetryPeriod

Descrição

A propriedade `ConnectionRetryPeriod` especifica o período de tempo, em segundos, em que o Interact tenta automaticamente novamente a solicitação de conexão com o banco de dados no caso de falha para as tabelas de execução de teste. O Interact tenta reconectar-se automaticamente ao banco de dados durante esse período de tempo antes de relatar um erro ou uma falha do banco de dados. Se o valor for configurado como 0, o Interact tentará novamente e indefinidamente. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

Valor padrão

-1

connectionRetryDelay

Descrição

A propriedade `ConnectionRetryDelay` especifica o período de tempo, em segundos, em que o Interact aguarda antes de tentar se reconectar ao banco de dados após uma falha nas tabelas de execução de teste. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

Valor padrão

-1

esquema

Descrição

O nome do esquema contendo as tabelas para execuções de teste de fluxograma interativo. O Interact insere o valor dessa propriedade antes de todos os nomes de tabela, por exemplo, `UACI_IntChannel` torna-se `schema.UACI_IntChannel`.

Não é necessário definir um esquema. Se você não definir um esquema, o Interact assumirá que o proprietário das tabelas é o mesmo que o do esquema. Você deve configurar esse valor para remover a ambiguidade.

Valor padrão

Nenhum valor padrão definido.

Interact | geral | contactAndResponseHistoryDataSource

Essas propriedades de configuração definem as configurações de conexão para a origem de dados de histórico de contatos e respostas necessárias para o rastreamento de resposta de sessão cruzada do Interact. Essas configurações não estão relacionadas ao módulo de histórico de contatos e respostas.

jndiName

Descrição

Use essa propriedade `jndiName` para identificar a origem de dados Java Naming and Directory Interface (JNDI) definida no servidor de aplicativos (WebSphere ou WebLogic) para a origem de dados do histórico de respostas e contatos necessária para o rastreamento de respostas de sessão cruzada do Interact.

Valor padrão

type

Descrição

O tipo de banco de dados para a origem de dados usado pela origem de dados do histórico de contatos e respostas necessária para o rastreamento de resposta de sessão cruzada do Interact.

Valor padrão

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descrição

A propriedade `ConnectionRetryPeriod` especifica o tempo em segundos em que o Interact tenta novamente automaticamente a solicitação de conexão com o banco de dados no caso de falha para o rastreamento de resposta de sessão cruzada do Interact. O Interact tenta reconectar-se automaticamente ao banco de dados durante esse período de tempo antes de relatar um erro ou uma falha do banco de dados. Se o valor for configurado como 0, o Interact tentará novamente e indefinidamente. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

Valor padrão

-1

connectionRetryDelay

Descrição

A propriedade ConnectionRetryDelay especifica o tempo em segundos que o Interact espera antes de tentar se reconectar ao banco de dados após uma falha para o rastreamento de resposta de sessão cruzada do Interact. Se o valor for configurado como -1, nenhuma nova tentativa será feita.

Valor padrão

-1

esquema

Descrição

O nome do esquema contendo as tabelas para o rastreamento de resposta de sessão cruzada do Interact. O Interact insere o valor dessa propriedade antes de todos os nomes de tabela, por exemplo, UACI_IntChannel torna-se schema.UACI_IntChannel.

Não é necessário definir um esquema. Se você não definir um esquema, o Interact assumirá que o proprietário das tabelas é o mesmo que o do esquema. Você deve configurar esse valor para remover a ambiguidade.

Valor padrão

Nenhum valor padrão definido.

Interact | geral | idsByType

Essas propriedades de configuração definem as configurações para números de ID usados pelo módulo de histórico de contatos e respostas.

initialValue

Descrição

O valor de ID inicial usado na geração de IDs usando a tabela UACI_IDsByType.

Valor padrão

1

Valores válidos

Qualquer valor maior que 0.

retries

Descrição

O número de novas tentativas antes da geração de uma exceção ao gerar IDs usando a tabela UACI_IDsByType.

Valor padrão

20

Valores válidos

Qualquer número inteiro maior que 0.

Interact | fluxograma

Esta seção define as definições de configuração para fluxogramas interativos.

defaultDateFormat

Descrição

O formato de data padrão usado pelo Interact para converter Data em Sequência e Sequência em Data.

Valor padrão

MM/dd/aa

idleFlowchartThreadTimeoutInMinutes

Descrição

O número de minutos que o Interact permite que um encadeamento dedicado de um fluxograma interativo fique inativo antes de liberar o encadeamento.

Valor padrão

5

idleProcessBoxThreadTimeoutInMinutes

Descrição

O número de minutos que o Interact permite que um encadeamento dedicado de um processo de fluxograma interativo fique inativo antes de liberar o encadeamento.

Valor padrão

5

maxSizeOfFlowchartEngineInboundQueue

Descrição

O número máximo de solicitações de execução de fluxograma que o Interact mantém na fila. Se esse número de solicitações for atingido, o Interact parará de aceitar solicitações.

Valor padrão

1000

maxNumberOfFlowchartThreads

Descrição

O número máximo de encadeamentos dedicados a solicitações de fluxograma interativo.

Valor padrão

25

maxNumberOfProcessBoxThreads

Descrição

O número máximo de encadeamentos dedicados a processos de fluxograma interativo.

Valor padrão

50

maxNumberOfProcessBoxThreadsPerFlowchart

Descrição

O número máximo de encadeamentos dedicados a processos de fluxograma interativo por instância de fluxograma.

Valor padrão

3

minNumberOfFlowchartThreads

Descrição

O número mínimo de encadeamentos dedicados a solicitações de fluxograma interativo.

Valor padrão

10

minNumberOfProcessBoxThreads

Descrição

O número mínimo de encadeamentos dedicados a processos de fluxograma interativo.

Valor padrão

20

sessionVarPrefix

Descrição

O prefixo para variáveis de sessão.

Valor padrão

SessionVar

Interact | fluxograma | ExternalCallouts | [ExternalCalloutName]

Esta seção define as configurações de classe para textos explicativos externos customizados gravados com a API de texto explicativo externa.

class

Descrição

O nome da classe Java representada por esse texto explicativo externo.

Essa é a classe Java que pode ser acessada com a Macro `EXTERNALCALLOUT` do IBM .

Valor padrão

Nenhum valor padrão definido.

classpath

Descrição

O caminho de classe para a classe Java representada por esse texto explicativo externo. O caminho de classe deve referenciar arquivos JAR no

servidor do ambiente de tempo de execução. Se estiver usando um grupo de servidores e todos os servidores de runtime estiverem usando o mesmo Marketing Platform, cada servidor deverá ter uma cópia do arquivo JAR no mesmo local. O caminho de classe deve consistir em locais absolutos de arquivos JAR, separados pelo delimitador de caminho do sistema operacional do servidor do ambiente de tempo de execução, por exemplo, ponto e vírgula (;) nos sistemas Windows e dois pontos (:) nos sistemas UNIX. Diretórios contendo arquivos de classe não são aceitos. Por exemplo, em um sistema Unix: `/path1/file1.jar:/path2/file2.jar`.

Este caminho de classe deve ter menos de 1024 caracteres. É possível usar o arquivo de manifesto em um arquivo .jar para especificar outros arquivos .jar, para que somente um arquivo .jar apareça no caminho da classe

Essa é a classe Java que pode ser acessada com a Macro `EXTERNALCALLOUT` do IBM .

Valor padrão

Nenhum valor padrão definido.

Interact | flowchart | ExternalCallouts | [ExternalCalloutName] | Dados do Parâmetro | [parameterName]

Esta seção define as configurações de parâmetro para um texto explicativo externo customizado gravado com a API de texto explicativo externo.

value

Descrição

O valor para qualquer parâmetro necessário para a classe para o texto explicativo externo.

Valor padrão

Nenhum valor padrão definido.

Exemplo

Se o texto explicativo externo texto exigir o nome do host de um servidor externo, crie uma categoria de parâmetro chamada `host` e defina a propriedade `value` como o nome do servidor.

Interact | monitoramento

Este conjunto de propriedades de configuração permite definir as configurações de monitoramento JMX. Somente será necessário configurar estas propriedades se você estiver usando o monitoramento JMX. Há propriedades de monitoramento JMX separadas a serem definidas para o módulo de histórico de contatos e respostas nas propriedades de configuração para o ambiente de design do Interact.

protocol

Descrição

Defina o protocolo para o serviço de sistema de mensagens do Interact.

Se escolher JMXMP, você deverá incluir os seguintes arquivos JAR no caminho da classe em ordem:

`Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar`

Valor padrão

JMXMP

Valores válidos

JMXMP | RMI

port

Descrição

O número da porta para o serviço do sistema de mensagens.

Valor padrão

9998

enableSecurity

Descrição

Um booleano que ativa ou desativa a segurança do serviço do sistema de mensagens JMXMP para o servidor de runtime do Interact. Se for configurado como `true`, você deverá fornecer um nome de usuário e uma senha para acessar o serviço JMX do tempo de execução do Interact. Essa credencial do usuário será autenticada pelo Marketing Platform para o servidor de runtime. O Jconsole não permite efetuar login com a senha vazia.

Essa propriedade não terá efeito se o protocolo for RMI. Esta propriedade não tem efeito sobre o JMX para Campaign (o tempo de design do Interact).

Valor padrão

Verdadeiro

Valores válidos

True | False

Interact | perfil

Este conjunto de propriedades de configuração controla vários dos recursos de entrega de oferta opcionais, incluindo a supressão de oferta e a substituição de pontuação.

enableScoreOverrideLookup

Descrição

Se for configurado como `True`, o Interact carregará os dados de substituição de pontuação da `scoreOverrideTable` ao criar uma sessão. Se for `False`, o Interact não carregará os dados de substituição de pontuação de marketing ao criar uma sessão.

Se for true, você também deverá configurar a propriedade `IBM EMM > Interact > perfil > níveis de público > (nível de público) > scoreOverrideTable`. Somente será necessário definir a propriedade `scoreOverrideTable` para os níveis de público que forem necessários. Deixar o `scoreOverrideTable` em branco para um nível de público desativará a tabela de substituição de pontuação para o nível de público.

Valor padrão

`False`

Valores válidos

True | False

enableOfferSuppressionLookup

Descrição

Se for configurado como `True`, o Interact carregará os dados de supressão de oferta da `offerSuppressionTable` ao criar uma sessão. Se for `False`, o Interact não carregará os dados de supressão de oferta ao criar uma sessão.

Se for true, você também deverá configurar a propriedade `IBM EMM > Interact > perfil > níveis de público > (nível de público) > offerSuppressionTable`. Somente será necessário definir a propriedade `enableOfferSuppressionLookup` para os níveis de público que forem necessários.

Valor padrão

`False`

Valores válidos

True | False

enableProfileLookup

Descrição

Para novas instalações do Interact, essa propriedade foi descontinuada. Em uma instalação atualizada do Interact, essa propriedade será válida até a primeira implementação.

O comportamento de carregamento de uma tabela usada em um fluxograma interativo, mas não mapeada no canal interativo. Se for configurado como `True`, o Interact carregará os dados de perfil da `profileTable` ao criar uma sessão.

Se for true, você também deverá configurar a propriedade `IBM EMM > Interact > perfil > níveis de público > (nível de público) > profileTable`.

A configuração **Carregar estes dados na memória quando uma sessão de visita for iniciada** no assistente de mapeamento de tabela do canal interativo substituirá essa propriedade de configuração.

Valor padrão

`False`

Valores válidos

True | False

defaultOfferUpdatePollPeriod

Descrição

O número de segundos que o sistema aguarda antes de atualizar as ofertas padrão no cache da tabela de ofertas padrão. Se for configurado como -1, o sistema não atualizará as ofertas padrão no cache depois que a lista inicial for carregada no cache quando o servidor de runtime for iniciado.

Valor padrão

-1

Interact | perfil | níveis de público | [AudienceLevelName]

Este conjunto de propriedades de configuração permite definir os nomes de tabela necessários para recursos adicionais do Interact. Somente será necessário definir o nome de tabela se você estiver usando o recurso associado.

scoreOverrideTable

Descrição

O nome da tabela contendo as informações de substituição de pontuação para esse nível de público. Essa propriedade será aplicável se você tiver configurado `enableScoreOverrideLookup` como true. É necessário definir essa propriedade para os níveis de público para os quais deseja ativar uma tabela de substituição de pontuação. Se não houver uma tabela de substituição de pontuação para esse nível de público, será possível deixar essa propriedade indefinida, mesmo se `enableScoreOverrideLookup` estiver configurado como true.

O Interact procura essa tabela nas tabelas do cliente acessadas pelos servidores de runtime do Interact, definidas pelas propriedades `prodUserDataSource`.

Se a propriedade `schema` tiver sido definida para essa origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, `schema.UACI_ScoreOverride`. Se você inserir um nome completo, por exemplo,`mySchema.UACI_ScoreOverride`, o Interact não prefixará o nome do esquema.

Valor padrão

UACI_ScoreOverride

offerSuppressionTable

Descrição

O nome da tabela contendo as informações de supressão de oferta para esse nível de público. Você deve definir essa propriedade para os níveis de público para os quais deseja ativar uma tabela de supressão de oferta. Se não houver uma tabela de supressão de oferta para esse nível de público, será possível deixar essa propriedade indefinida. Se `enableOfferSuppressionLookup` estiver configurado como true, essa propriedade deverá ser configurada como uma tabela válida.

O Interact procura por essa tabela nas tabelas do cliente acessadas pelos servidores de runtime, definidos pelas propriedades `prodUserDataSource`.

Valor padrão

UACI_BlackList

profileTable

Descrição

Para novas instalações do Interact, essa propriedade foi descontinuada. Em uma instalação atualizada do Interact, essa propriedade será válida até a primeira implementação.

O nome da tabela contendo os dados do perfil para esse nível de público.

O Interact procura por essa tabela nas tabelas do cliente acessadas pelos servidores de runtime, definidos pelas propriedades prodUserDataSource.

Se a propriedade schema tiver sido definida para esta origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, schema.UACI_usrProd. Se você inserir um nome completo, por exemplo, mySchema.UACI_usrProd, o Interact não prefixará o nome do esquema.

Valor padrão

Nenhum valor padrão definido.

contactHistoryTable

Descrição

O nome da tabela de migração para dados de histórico de contatos para esse nível de público.

Essa tabela é armazenada nas tabelas do ambiente de tempo de execução (systemTablesDataSource).

Se a propriedade schema tiver sido definida para essa origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, schema.UACI_CHStaging. Se você inserir um nome completo, por exemplo, mySchema.UACI_CHStaging, o Interact não prefixará o nome do esquema.

Se a criação de log do histórico de contatos estiver desativada, essa propriedade não precisará ser configurada.

Valor padrão

UACI_CHStaging

chOfferAttribTable

Descrição

O nome da tabela de atributos de oferta do histórico de contatos para esse nível de público.

Essa tabela é armazenada nas tabelas do ambiente de tempo de execução (systemTablesDataSource).

Se a propriedade schema tiver sido definida para essa origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, schema.UACI_CHOfferAttrib. Se você inserir um nome completo, por exemplo, mySchema.UACI_CHOfferAttrib, o Interact não prefixará o nome do esquema.

Se a criação de log do histórico de contatos estiver desativada, essa propriedade não precisará ser configurada.

Valor padrão

UACI_CHOfferAttrib

responseHistoryTable

Descrição

O nome da tabela de migração de histórico de respostas para este nível de púlico.

Essa tabela é armazenada nas tabelas do ambiente de tempo de execução (systemTablesDataSource).

Se a propriedade schema tiver sido definida para essa origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, schema.UACI_RHStaging. Se você inserir um nome completo, por exemplo, mySchema.UACI_RHStaging, o Interact não prefixará o nome do esquema.

Se a criação de log do histórico de respostas estiver desativada, essa propriedade não precisará ser configurada.

Valor padrão

UACI_RHStaging

crossSessionResponseTable

Descrição

O nome da tabela para este nível de público necessário para o rastreamento de resposta de sessão cruzada nas tabelas de histórico de contatos e respostas acessível para o recurso de rastreamento de resposta.

Se a propriedade schema tiver sido definida para essa origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, schema.UACI_XSessResponse. Se você inserir um nome completo, por exemplo, mySchema.UACI_XSessResponse, o Interact não prefixará o nome do esquema.

Se a criação de log de resposta de sessão cruzada estiver desativada, essa propriedade não precisará ser configurada.

Valor padrão

UACI_XSessResponse

userEventLoggingTable

Descrição

Este é o nome da tabela de banco de dados usada para a criação de log das atividades de eventos definidos pelo usuário. Os eventos definidos pelo usuário na guia Eventos das páginas Resumo do canal interativo na interface do Interact. A tabela de banco de dados especificada aqui armazena informações como ID de evento, nome, quantas vezes esse evento ocorreu para esse nível de público desde a última vez que o cache de atividade de evento foi limpo e assim por diante.

Se a propriedade schema tiver sido definida para essa origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, schema.UACI_UserEventActivity. Se você inserir um nome completo, por exemplo, mySchema.UACI_UserEventActivity, o Interact não prefixará o nome do esquema.

Valor padrão

UACI_UserEventActivity

patternStateTable

Descrição

Este é o nome da tabela de banco de dados usada para a criação de log de estados de padrão do evento, como se a condição do padrão foi ou não atendida, se o padrão expirou ou foi desativado e assim por diante.

Se a propriedade schema tiver sido definida para essa origem de dados, o Interact prefixará o nome dessa tabela com o esquema, por exemplo, schema.UACI_EventPatternState. Se você inserir um nome completo, por exemplo, mySchema.UACI_EventPatternState, o Interact não prefixará o nome do esquema.

Um patternStateTable será necessário para cada nível de público mesmo se você não usar os padrões de evento. O patternStateTable será baseado na DDL do UACI_EventPatternState incluído. A seguir está um exemplo no qual o ID de público tem dois componentes: ComponentNum e 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)
)
```

Valor padrão

UACI_EventPatternState

Interagir | perfil | Níveis de Público | [AudienceLevelName] | Ofertas por SQL Bruto

Este conjunto de propriedades de configuração permite definir os nomes de tabela necessários para recursos adicionais do Interact. Somente será necessário definir o nome de tabela se você estiver usando o recurso associado.

enableOffersByRawSQL

Descrição

Se for configurado como True, o Interact ativará o recurso offersBySQL para esse nível de público que permitirá configurar o código SQL a ser executado para criar um conjunto desejado de ofertas candidatas no tempo de execução. Se for False, o Interact não usará o recurso offersBySQL.

Ao configurar essa propriedade como true, você também poderá configurar a propriedade Interact | perfil | níveis de público | (Audience Level) | Ofertas por SQL bruto | Modelo de SQL para definir um ou mais modelos de SQL.

Valor padrão

Falso

Valores válidos

True | False

cacheSize

Descrição

Tamanho do cache usado para armazenar resultados das consultas OfferBySQL. Observe que o uso de um cache poderá ter impacto negativo se os resultados da consulta forem exclusivos para a maioria das sessões.

Valor padrão

`-1 (desligado)`

Valores válidos

-1 | Valor

cacheLifeInMinutes

Descrição

Se o cache estiver ativado, indicará o número de minutos antes que o sistema limpe o cache para evitar conteúdo antigo.

Valor padrão

`-1 (desligado)`

Valores válidos

-1 | Valor

defaultSQLTemplate

Descrição

O nome do modelo SQL a ser usado se não for especificado um por meio das chamadas de API.

Valor padrão

`None`

Valores válidos

Nome do modelo SQL

Interagir | perfil | Níveis de Público | [AudienceLevelName] | Modelo SQL

Essas propriedades de configuração permitem definir um ou mais modelos de consulta SQL usados pelo recurso offersBySQL do Interact.

name

Descrição

O nome que você deseja designar a esse modelo de consulta SQL. Insira um nome descritivo que será significativo quando você usar este modelo SQL em chamadas de API. Observe que se você usar um nome aqui que seja *idêntico* a um nome definido na caixa do processo Lista de interação para um tratamento offerBySQL, o SQL na caixa de processo será usado no lugar do SQL inserido aqui.

Valor padrão

None

SQL

Descrição

Contém a consulta SQL a ser chamada por este modelo. A consulta SQL pode conter referências a nomes de variável que façam parte dos dados da sessão do visitante (perfil). Por exemplo, `select * from MyOffers where category = ${preferredCategory}` dependeria da sessão que contivesse uma variável chamada `preferredCategory`.

Você deve configurar o SQL para consultar as tabelas de ofertas específica criadas durante o tempo de design para serem usadas por este recurso. Observe que os procedimentos armazenados não são suportados aqui.

Valor padrão

None

Interact | perfil | níveis de público | [AudienceLevelName | serviços de dados de perfil | [DataSource]

Este conjunto de propriedades de configuração permite definir os nomes de tabela necessários para recursos adicionais do Interact. Somente será necessário definir o nome de tabela se você estiver usando o recurso associado. A categoria Serviços de dados de perfil fornece informações sobre uma origem de dados integrada (chamada Banco de dados) que é criada para todos os níveis de público e que é pré-configurada com a prioridade 100. No entanto, é possível escolher modificá-la ou desativá-la. Esta categoria também contém um modelo para as origens de dados externas adicionais. Ao clicar no modelo chamado **Serviços de dados externos** será possível concluir as definições de configuração descritas aqui.

Nome da nova categoria

Descrição

(Não disponível para a entrada de banco de dados padrão.) O nome da origem de dados que está sendo definida. O nome inserido aqui deve ser exclusivo entre as origens de dados do mesmo nível de público.

Valor padrão

None

Valores válidos

Qualquer sequência de texto é permitida.

enabled

Descrição

Se for configurado como `True`, essa origem de dados será ativada para o nível de público ao qual ela foi designada. Se for `False`, o Interact não usará essa origem de dados para esse nível de público.

Valor padrão

`Verdadeiro`

Valores válidos

True | False

className

Descrição

(Não disponível para a entrada de banco de dados padrão.) O nome completo da classe da origem de dados que implementa `IInteractProfileDataService`.

Valor padrão

None.

Valores válidos

Uma sequência que fornece um nome completo da classe.

classPath

Descrição

(Não disponível para a entrada de banco de dados padrão.) Uma definição de configuração opcional que fornece o caminho para carregar esta classe de implementação da origem de dados. Se for omitido, o caminho de classe do servidor de aplicativos que o contém será usado por padrão.

Valor padrão

Não mostrado, mas o caminho de classe do servidor de aplicativos que o contém será usado por padrão se nenhum valor for fornecido aqui.

Valores válidos

Uma sequência que fornece o caminho da classe.

priority

Descrição

A prioridade desta origem de dados neste nível de público. Deve ser um valor exclusivo entre todas as origens de dados para cada nível de público. (Ou seja, se uma prioridade for definida como 100 para uma origem de dados, nenhuma outra origem de dados no nível de público poderá ter a prioridade 100.)

Valor padrão

`100` para o Banco de dados padrão, `200` para origem de dados definida pelo usuário

Valores válidos

Qualquer número inteiro não negativo é permitido.

Interact | offerserving

Essas propriedades de configuração definem as propriedades de configuração de aprendizado genéricas. Se estiver usando o aprendizado integrado, para ajustar a implementação de aprendizado, use as propriedades de configuração para o ambiente de design.

offerTieBreakMethod

Descrição

A propriedade `offerTieBreakMethod` define o comportamento da oferta entregue quando duas ofertas possuem pontuações equivalentes (empatadas). Se você configurar essa propriedade para o valor padrão Aleatório, o Interact apresentará uma opção aleatória entre as ofertas com pontuações equivalentes. Se você definir essa configuração como Oferta mais recente, o Interact entregará a oferta mais recente (com base no maior ID de oferta) antes da oferta mais antiga (menor ID de oferta) caso as pontuações das ofertas sejam iguais.

Nota:

O Interact tem um recurso opcional que permite que o administrador configure o sistema para retornar as ofertas em ordem aleatória

independentemente da pontuação configurando a opção percentRandomSelection (Campaign | partitions | [partition_number] | Interact | learning | percentRandomSelection). A propriedade offerTieBreakMethod descrita aqui somente é usada quando percentRandomSelection está configurado como zero (desativado).

Valor padrão

Aleatório

Valores válidos

Aleatório | Oferta mais recente

optimizationType

Descrição

A propriedade optimizationType define se o Interact usa um mecanismo de aprendizado para ajudar nas designações de oferta. Se for configurado como NoLearning, o Interact não usará aprendizado. Se for configurado como BuiltInLearning, o Interact usará o mecanismo de aprendizado bayesiano construído com o Interact. Se configurado como ExternalLearning, o Interact usará um mecanismo de aprendizado fornecido. Ao selecionar ExternalLearning, você deverá definir as propriedades externalLearningClass e externalLearningClassPath.

Valor padrão

NoLearning

Valores válidos

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

Descrição

O número máximo de milissegundos que o servidor de runtime aguarda para que um fluxograma interativo seja concluído antes de obter ofertas.

Valor padrão

5000

treatmentCodePrefix

Descrição

O prefixo anexado aos códigos de tratamento.

Valor padrão

Nenhum valor padrão definido.

effectiveDateBehavior

Descrição

Determina se o Interact deve usar a data efetiva de uma oferta na filtragem das ofertas que são apresentadas a um visitante. Os valores incluem:

- -1 informa a Interact que a data efetiva na oferta deve ser ignorada.

0 informa o Interact que a data efetiva deve ser usada para filtrar a oferta, de modo que se a data efetiva da oferta for anterior ou igual à data atual, a data efetiva da oferta, a oferta será entregue aos visitantes.

Se houver um valor **effectiveDateGracePeriod** configurado, o período de carência também será aplicado para determinar se a oferta deve ser entregue.

- Qualquer número inteiro positivo informa o Interact que deve ser usada a data atual mais o valor dessa propriedade para determinar se a oferta deverá ser entregue aos visitantes, de forma que, se a data efetiva da oferta for anterior à data atual mais o valor desta propriedade, a oferta será entregue aos visitantes.

  Se houver um valor **effectiveDateGracePeriod** configurado, o período de carência também será aplicado para determinar se a oferta deve ser entregue.

Valor padrão

-1

effectiveDateGracePeriodOfferAttr

Descrição

Especifica o nome do atributo customizado em uma definição de oferta que indica o período de carência da data efetiva. Por exemplo, é possível configurar essa propriedade com o valor `AltGracePeriod`. Em seguida, será possível definir as ofertas com um atributo customizado chamado `AltGracePeriod` que será usado para especificar o número de dias a ser usado como um período de carência com a propriedade **effectiveDateBehavior**.

Suponha que você crie um novo modelo de oferta com uma data efetiva de 10 dias a partir da data atual e inclua um atributo customizado chamado `AltGracePeriod`. Ao criar uma oferta usando o modelo, se você configurar o valor de `AltGracePeriod` como 14 dias, a oferta será entregue aos visitantes, porque a data atual estará dentro do período de carência da data efetiva da oferta.

Valor padrão

Em branco

alwaysLogLearningAttributes

Descrição

Indica se o Interact deve gravar informações sobre os atributos do visitante usados pelo módulo de aprendizado para os arquivos de log. Observe que configurar esse valor como true pode afetar o desempenho do aprendizado e o tamanho dos arquivos de log.

Valor padrão

Falso

Interact | offerserving | configuração de aprendizado integrado

Essas propriedades de configuração definem as configurações de gravação do banco de dados para aprendizado integrado. Para ajustar a implementação de aprendizado, use as propriedades de configuração para o ambiente de design.

version

Descrição

É possível selecionar 1 ou 2. Versão 1 é a versão de configuração básica que não usa parâmetros para configurar limites de encadeamento e registro. Versão 2 é a versão de configuração aprimorada que permite configurar parâmetros de encadeamento e registro para melhorar o desempenho. Esses parâmetros executam agregação e exclusão quando esses limites de parâmetro são atingidos.

Valor padrão

1

insertRawStatsIntervalInMinutes

Descrição

O número de minutos que o módulo de aprendizado do Interact aguarda antes de inserir mais linhas nas tabelas de migração de aprendizado. Poderá ser necessário modificar esse tempo com base na quantia de dados que o módulo de aprendizado estiver processando no ambiente.

Valor padrão

5

Valores válidos

Um número inteiro positivo

aggregateStatsIntervalInMinutes

Descrição

O número de minutos que o módulo de aprendizado do Interact aguarda antes de agregar dados nas tabelas de estatísticas de aprendizado. Poderá ser necessário modificar esse tempo com base na quantia de dados que o módulo de aprendizado estiver processando no ambiente.

Valor padrão

15

Valores válidos

Um número inteiro maior que zero

autoAdjustPercentage

Descrição

O valor que determina a porcentagem de dados que a execução de agregação tenta processar com base nas métricas da execução anterior. Por padrão, esse valor é configurado como zero, o que significa que o agregador processa todos os registros de temporariedade, e essa funcionalidade de ajuste automático é desativada.

Valor padrão

0

Valores válidos

Um número entre 0 e 100.

enableObservationModeOnly

Descrição

Se for configurado como `True`, ativará um modo de aprendizado no qual o Interact coleta dados para aprendizado sem usar esses dados para recomendações ou arbitragem de oferta. Isso permite operar o autoaprendizado em um modo de inicialização até determinar que dados suficientes foram coletados para recomendações.

Valor padrão

Falso

Valores válidos

True | False

excludeAbnormalAttribute

Descrição

A configuração que determina se esses atributos devem ser marcados como inválido. Se configurado como `IncludeAttribute`, os atributos anormais serão incluídos e não marcados como inválidos. Se for configurado como `ExcludeAttribute`, os atributos anormais serão excluídos e marcados como inválidos.

Valor padrão

IncludeAttribute

Valores válidos

IncludeAttribute | ExcludeAttribute

Interact | offerserving | configuração de aprendizado integrado | dados de parâmetro | [parameterName]

Essas propriedades de configuração definem todos os parâmetros para o módulo de aprendizado externo.

numberOfThreads

Descrição

O número máximo de encadeamentos que o agregador de aprendizado usa para processar os dados. Um valor válido é um número inteiro positivo e não deve ser maior do que o número máximo de conexões configurado na origem de dados de aprendizado. Esse parâmetro somente é usado pela versão do agregador 2.

Valor padrão

10

maxLogTimeSpanInMin

Descrição

Se a versão do agregador 1 for selecionada, será possível processar os registros de temporariedade em iterações para evitar lotes de banco de dados demasiadamente grandes. Nesse caso, esses registros de temporariedade serão processadas por chunks, iteração por iteração, em um único ciclo de agregação. O valor desse parâmetro especifica o período

máximo de tempo de registros de temporariedade que o agregador tenta processar em cada iteração. Esse período de tempo é baseado no campo LogTime associado a cada registro de temporariedade e somente os registros cujos LogTime estiverem na primeira janela de tempo serão processados. Um valor válido é um número inteiro não negativo. Se o valor for 0, não haverá limite, o que significa que todos os registros de temporariedade serão processados em uma única iteração.

Valor padrão

0

maxRecords

Descrição

Se a versão do agregador 2 for selecionada, será possível processar os registros de temporariedade em iterações para evitar lotes de banco de dados demasiadamente grandes. Nesse caso, esses registros de temporariedade serão processados por chunks, iteração por iteração, em um único ciclo de agregação. O valor deste parâmetro especifica o número máximo de registros de temporariedade que o agregador tenta processar em cada iteração. Um valor válido é um número inteiro não negativo. Se o valor for 0, não haverá limite, o que significa que todos os registros de temporariedade serão processados em uma única iteração.

Valor padrão

0

value

Descrição

O valor para qualquer parâmetro necessário para a classe de um módulo de aprendizado integrado.

Valor padrão

Nenhum valor padrão definido.

Interact | offerserving | configuração de aprendizado externa

Essas propriedades de configuração definem as configurações de classe para um módulo de aprendizado externo que foi gravado usando a API de aprendizado.

class

Descrição

Se `optimizationType` estiver configurado como `ExternalLearning`, configure `externalLearningClass` com o nome da classe para o mecanismo de aprendizado externo.

Valor padrão

Nenhum valor padrão definido.

Disponibilidade

Essa propriedade semente será aplicável se `optimizationType` estiver configurado como `ExternalLearning`.

classPath

Descrição

Se `optimizationType` estiver configurado como `ExternalLearning`, configure `externalLearningClass` com o caminho de classe do mecanismo de aprendizado externo.

O caminho de classe deve referenciar arquivos JAR no servidor do ambiente de tempo de execução. Se estiver usando um grupo de servidores e todos os servidores de runtime estiverem usando o mesmo Marketing Platform, cada servidor deverá ter uma cópia do arquivo JAR no mesmo local. O caminho de classe deve consistir em locais absolutos de arquivos JAR, separados pelo delimitador de caminho do sistema operacional do servidor do ambiente de tempo de execução, por exemplo, ponto e vírgula (;) nos sistemas Windows e dois pontos (:) nos sistemas UNIX. Diretórios contendo arquivos de classe não são aceitos. Por exemplo, em um sistema Unix: `/path1/file1.jar:/path2/file2.jar`.

Este caminho de classe deve ter menos de 1024 caracteres. É possível usar o arquivo de manifesto em um arquivo .jar para especificar outros arquivos .jar, para que somente um arquivo .jar apareça no caminho da classe

Valor padrão

Nenhum valor padrão definido.

Disponibilidade

Essa propriedade semente será aplicável se `optimizationType` estiver configurado como `ExternalLearning`.

Interact | offerserving | configuração de aprendizado externa | dados de parâmetro | [parameterName]

Essas propriedades de configuração definem todos os parâmetros para o módulo de aprendizado externo.

value

Descrição

O valor de qualquer parâmetro necessário para classe para o módulo de aprendizado externo.

Valor padrão

Nenhum valor padrão definido.

Exemplo

Se o módulo de aprendizado externo exigir um caminho para um aplicativo solucionador de algoritmo, você deverá criar uma categoria de parâmetro chamada `solverPath` e definir a propriedade `value` como o caminho para o aplicativo.

Interact | serviços

As propriedades de configuração nesta categoria definem as configurações para todos os serviços que gerenciam a coleta de dados e estatísticas de histórico de contatos e respostas para relatório e gravação nas tabelas de sistema do ambiente de tempo de execução.

externalLoaderStagingDirectory

Descrição

Esta propriedade define a localização do diretório de temporariedade para um utilitário de carregamento do banco de dados.

Valor padrão

Nenhum valor padrão definido.

Valores válidos

Um caminho relativo para o diretório de instalação do Interact ou um caminho absoluto para um diretório de temporariedade.

Se você ativar um utilitário de carregamento de banco de dados, você deverá configurar a propriedade `cacheType` nas categorias `contactHist` e `responstHist` como `Arquivo do carregador externo`.

Interact | serviços | contactHist

As propriedades de configuração nesta categoria definem as configurações do serviço que coleta dados para as tabelas de migração de históricos de contatos.

enableLog

Descrição

Se for `true`, ativará o serviço que coleta dados para gravação de dados de históricos de contatos. Se for `false`, nenhum dado será coletado.

Valor padrão

Verdadeiro

Valores válidos

True | False

cacheType

Descrição

Define se os dados coletados para o histórico de contatos são mantidos na memória (`Cache de memória`) ou em um arquivo (`Arquivo do carregador externo`). Somente será possível usar o `Arquivo do carregador externo` se você tiver configurado o Interact para usar um utilitário de carregador de banco de dadoss.

Se você selecionar `Cache de memória`, use as configurações de categoria `cache`. Se você selecionar `Arquivo do carregador externo`, use as configurações de categoria `fileCache`.

Valor padrão

Cache de memória

Valores válidos

Memory Cache | External Loader File

Interact | serviços | contactHist | cache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coleta os dados para a tabela de migração do histórico de contatos.

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar os dados do histórico de contatos coletados para o banco de dados.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | contactHist | fileCache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coletará dados do histórico de contatos se você estiver usando um utilitário de carregador de banco de dados.

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar os dados do histórico de contatos coletados para o banco de dados.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | defaultedStats

As propriedades de configuração nesta categoria definem as configurações do serviço que coleta as estatísticas relativas ao número de vezes que a sequência padrão do ponto de interação foi usada.

enableLog

Descrição

Se for `true`, ativará o serviço que coleta as estatísticas relativas ao número de vezes que a sequência padrão do ponto de interação foi usada para a tabela `UACI_DefaultedStat`. Se for `false`, nenhuma estatística de sequência padrão será coletada.

Se você não estiver usando o relatório do IBM, será possível configurar essa propriedade como `false` pois a coleta de dados não será necessária.

Valor padrão

Verdadeiro

Valores válidos

True | False

Interact | serviços | defaultedStats | cache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coleta as estatísticas relativas ao número de vezes que a sequência padrão para o ponto de interação é usada.

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar as estatísticas de sequência padrão coletadas no banco de dados.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | eligOpsStats

As propriedades de configuração nesta categoria definem as configurações do serviço que grava as estatísticas para ofertas elegíveis.

enableLog

Descrição

Se for `true`, ativará o serviço que coleta as estatísticas de ofertas elegíveis. Se for `false`, nenhuma estatística de oferta elegíveis será coletada.

Se você não estiver usando o relatório do IBM, será possível configurar essa propriedade como `false` pois a coleta de dados não será necessária.

Valor padrão

Verdadeiro

Valores válidos

True | False

Interact | serviços | eligOpsStats | cache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coleta as estatísticas de ofertas elegíveis.

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar no banco de dados as estatísticas de ofertas elegíveis coletadas.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | eventActivity

As propriedades de configuração nesta categoria definem as configurações para o serviço que coleta as estatísticas de atividade do evento.

enableLog

Descrição

Se for `true`, ativará o serviço que coleta as estatísticas de atividade de evento. Se for `false`, nenhuma estatística de evento será coletada.

Se você não estiver usando o relatório do IBM, será possível configurar essa propriedade como `false` pois a coleta de dados não será necessária.

Valor padrão

Verdadeiro

Valores válidos

True | False

Interact | serviços | eventActivity | cache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coleta as estatísticas de atividade do evento.

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar no banco de dados as estatísticas de atividade de evento coletadas.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | eventPattern

As propriedades de configuração na categoria `eventPattern` definem as configurações para o serviço que coleta estatísticas de atividade de padrão de evento.

persistUnknownUserStates

Descrição

Determina se os estados de padrão de evento para um ID de público desconhecido (visitante) ficam retidos no banco de dados. Por padrão, quando a sessão termina, os status de todos os padrões de evento atualizados associados ao ID de público do visitante serão armazenados no banco de dados, desde que o ID de público seja conhecido (ou seja, o perfil do visitante possa ser localizado na origem de dados do perfil).

A propriedade `persistUnknownUserStates` determina o que acontece quando o ID de público não é conhecido. Por padrão, essa propriedade é configurada como `False` e para IDs de público desconhecidos, os estados de padrão de evento são descartados no final da sessão.

Se você configurar essa propriedade como `True`, os estados de padrão de evento de usuários desconhecidos (cujos perfis não possam ser localizados no serviço de dados de perfil configurado) persistirão.

Valor padrão

Falso

Valores válidos

True | False

mergeUnknowUserInSessionStates

Descrição

Determina como os estados de padrão de evento para IDs de público desconhecidos (visitantes) são retidos. Se o ID de público mudar no meio de uma sessão, o Interact tentará carregar os estados de padrão de evento salvos para o novo ID de público da tabela de banco de dados. Quando o ID de público for desconhecido inicialmente e a propriedade `mergeUnknowUserInSessionStates` estiver configurada como `True`, as atividades de evento do usuário pertencentes ao ID de público anterior na mesma sessão serão mescladas no novo ID de público.

Valor padrão

Falso

Valores válidos

True | False

enableUserEventLog

Descrição

Determina se atividades de evento do usuário são registradas no banco de dados.

Valor padrão

Falso

Valores válidos

True | False

Interact | serviços | eventPattern | userEventCache

As propriedades de configuração na categoria `userEventCache` definem as configurações que determinam quando a atividade de evento é movida do cache para persistir no banco de dados.

Limite

Descrição

Determina o número máximo de estados de padrão de evento que podem ser armazenados no cache de estado de padrão de evento. Quando o limite é atingido, os estados menos usados recentemente são esvaziados do cache.

Valor padrão

100

Valores válidos

O número desejado de estados de padrão de evento a serem retidos no cache.

insertPeriodInSecs

Descrição

Determina o período máximo de tempo, em segundos, em que as atividades de evento do usuário são enfileiradas na memória. Quando o limite de tempo especificado por esta propriedade é atingido, essas atividades são persistidas no banco de dados.

Valor padrão

3600 (60 minutos)

Valores válidos

O número desejado de segundos.

Interact | serviços | eventPattern | advancedPatterns

As propriedades de configuração nesta categoria controlam se a integração com o Interact Opportunity Detection é ativada e definem os intervalos de tempo limite para conexões com o Interact Opportunity Detection.

enableAdvancedPatterns

Descrição

Se for `true`, ativará a integração com o Interact Opportunity Detection. Se for `false`, a integração não será ativada. Se a integração tiver sido ativada anteriormente, o Interact usará os estados de padrão mais recentes recebidos a partir do Interact Opportunity Detection.

Valor padrão

Verdadeiro

Valores válidos

True | False

connectionTimeoutInMilliseconds

Descrição

O tempo máximo que pode ser gasto para executar uma conexão HTTP a partir do ambiente de tempo real do Interact com o Interact Opportunity Detection. Se a solicitação atingir o tempo limite, o Interact usará os últimos dados salvos a partir dos padrões.

Valor padrão

30

readTimeoutInMilliseconds

Descrição

Depois que uma conexão HTTP for estabelecida entre o ambiente de tempo real do Interact e o Interact Opportunity Detection e uma solicitação for enviada ao Interact Opportunity Detection para obter o status de um padrão de evento, o tempo máximo que poderá levar para receber dados. Se a solicitação atingir o tempo limite, o Interact usará os últimos dados salvos a partir dos padrões.

Valor padrão

100

connectionPoolSize

Descrição

O tamanho do conjunto de conexões HTTP para comunicação entre o ambiente de tempo real do Interact e o Interact Opportunity Detection.

Valor padrão

10

Interact | serviços | eventPattern | advancedPatterns | autoReconnect

As propriedades de configuração nesta categoria especificam os parâmetros para o recurso de reconexão automática na integração com o Interact Opportunity Detection.

ativar

Descrição

Determina se o sistema se reconectará automaticamente se ocorrerem problemas de conexão entre o ambiente de tempo real do Interact e o Interact Opportunity Detection. O valor padrão **True** ativa este recurso.

Valor padrão

Verdadeiro

Valores válidos

True | False

durationInMinutes

Descrição

Essa propriedade especifica o intervalo de tempo, em minutos, durante o qual o sistema avalia problemas de conexão repetidos ocorrendo entre o ambiente de tempo real do Interact e o Interact Opportunity Detection.

Valor padrão

10

numberOfFailuresBeforeDisconnect

Descrição

Esta propriedade especifica o número de falhas de conexão permitidas durante o período de tempo especificado antes que o sistema seja desconectado automaticamente do Interact Opportunity Detection.

Valor padrão

3

consecutiveFailuresBeforeDisconnect

Descrição

Determina se o recurso de nova conexão automática somente avalia falhas de conexão consecutivas entre o ambiente em tempo real Interact e o Interact Opportunity Detection. Se você configurar esse valor como **False**, todas as falhas no intervalo de tempo especificado serão avaliadas.

Valor padrão

Verdadeiro

sleepBeforeReconnectDurationInMinutes

Descrição

O sistema aguarda o número de minutos especificado nesta propriedade antes de reconectar depois que o sistema desconecta devido a falhas repetidas, conforme a especificação nas outras propriedades nesta categoria.

Valor padrão

5

sendNotificationAfterDisconnect

Descrição

Esta propriedade determina se o sistema envia uma notificação por email quando ocorre uma falha de conexão. A mensagem de notificação inclui o nome da instância de tempo real do Interact para a qual a falha ocorreu e o período de tempo antes da ocorrência da nova conexão, da maneira especificada na propriedade **sleepBeforeReconnectDurationInMinutes**. O valor padrão **True** significa que as notificações são enviadas.

Valor padrão

Verdadeiro

Interact | serviços | customLogger

As propriedades de configuração nesta categoria definem as configurações para o serviço que coleta dados customizados para gravar em uma tabela (um evento que usa o parâmetro de evento UACICustomLoggerTableName).

enableLog

Descrição

Se for true, ativa o recurso de registro em tabela customizado. Se for false, o parâmetro de evento UACICustomLoggerTableName não terá efeito.

Valor padrão

Verdadeiro

Valores válidos

True | False

Interact | serviços | customLogger | cache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coleta dados customizados para uma tabela (um evento que usa o parâmetro de evento UACICustomLoggerTableName).

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar os dados customizados coletados para o banco de dados.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | responseHist

As propriedades de configuração nesta categoria definem as configurações para o serviço que grava nas tabelas de migração do histórico de respostas.

enableLog

Descrição

Se for true, ativará o serviço que grava nas tabelas de migração do histórico de respostas. Se for false, nenhum dado será gravado nas tabelas de migração do histórico de respostas.

A tabela de migração do histórico de respostas é definida pela propriedade responseHistoryTable para o nível de público. O padrão é UACI_RHStaging.

Valor padrão

Verdadeiro

Valores válidos

True | False

cacheType

Descrição

Define se o cache deve ser mantido na memória ou em um arquivo. Somente será possível usar o `Arquivo do carregador externo` se o Interact estiver configurado para usar um utilitário de carregador de banco de dados.

Se você selecionar `Cache de memória`, use as configurações de categoria `cache`. Se você selecionar `Arquivo do carregador externo`, use as configurações de categoria `fileCache`.

Valor padrão

Cache de memória

Valores válidos

Memory Cache | External Loader File

actionOnOrphan

Descrição

Esta configuração determina o que fazer com os eventos de resposta que não possuem eventos de contato correspondentes. Se for definida como `NoAction`, o evento de resposta será processado como se o evento de contato correspondente tivesse sido postado. Se for definida como `Aviso`, o evento de resposta será processado como se o evento de contato correspondente tivesse sido postado, mas uma mensagem de aviso será gravada no `interact.log`. Se for definida como `Ignorar`, o evento de resposta não será processado e uma mensagem de erro será gravada no `interact.log`. A configuração escolhida aqui ficará em vigor, independentemente da ativação da criação de log do histórico de respostas.

Valor padrão

NoAction

Valores válidos

NoAction | Warning | Skip

Interact | serviços | responseHist | cache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coleta os dados do histórico de respostas.

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar os dados do histórico de respostas coletados no banco de dados.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | responseHist | fileCache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coletará os dados de histórico de respostas se um utilitário de carregador de banco de dados estiver em uso.

Limite

Descrição

O número de registros acumulados antes do Interact gravá-los no banco de dados.

`responseHist` - A tabela definida pela propriedade `responseHistoryTable` para o nível de público. O padrão é `UACI_RHStaging`.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas no banco de dados.

Valor padrão

3600

Interact | serviços | crossSessionResponse

As propriedades de configuração nesta categoria definem as configurações gerais para o serviço crossSessionResponse e o processo xsession. Somente será necessário definir essas configurações se você estiver usando o rastreamento de resposta de sessão cruzada do Interact.

enableLog

Descrição

Se for `true`, ativará o serviço crossSessionResponse e o Interact gravará dados nas tabelas de migração de rastreamento de resposta de sessão cruzada. Se for `false`, desativará o serviço crossSessionResponse.

Valor padrão

Falso

xsessionProcessIntervalInSecs

Descrição

O número de segundos entre execuções do processo xsession. Esse processo move dados das tabelas de migração de rastreamento de resposta de sessão cruzada para a tabela de migração de histórico de respostas e o módulo de aprendizado integrado.

Valor padrão

180

Valores válidos

Um número inteiro maior que zero.

purgeOrphanResponseThresholdInMinutes

Descrição

O número de minutos que o serviço crossSessionResponse aguarda antes de marcar todas as respostas que não correspondem aos contatos nas tabelas de históricos de contatos e respostas.

Se uma resposta não tiver correspondência nas tabelas de históricos de contatos e respostas, após `purgeOrphanResponseThresholdInMinutes` minutos, o Interact marcará a resposta com o valor -1 na coluna `Marca` da tabela de migração xSessResponse. Em seguida, será possível corresponder manualmente ou excluir essas respostas.

Valor padrão

180

Interact | serviços | crossSessionResponse | cache

As propriedades de configuração nesta categoria definem as configurações de cache para o serviço que coleta dados de resposta de sessão cruzada.

Limite

Descrição

O número de registros acumulados antes do serviço flushCacheToDB gravar os dados de respostas de sessão cruzada coletados no banco de dados.

Valor padrão

100

insertPeriodInSecs

Descrição

O número de segundos entre gravações forçadas na tabela `XSessResponse`.

Valor padrão

3600

Interact | serviços | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

As propriedades nesta seção definem como o rastreamento de resposta de sessão cruzada corresponde códigos de tratamento ao histórico de contatos e respostas.

SQL

Descrição

Esta propriedade define se o Interact usa o SQL gerado pelo sistema ou SQL customizado definido na propriedade `OverrideSQL`.

Valor padrão

Usar SQL gerado pelo sistema

Valores válidos

Usar SQL gerado pelo sistema | Substituir SQL

OverrideSQL

Descrição

Se você não usar o comando SQL padrão para corresponder ao código de tratamento com o histórico de contatos e respostas, insira aqui o SQL ou procedimento armazenado.

Esse valor será ignorado se `SQL` estiver configurado como `Usar SQL gerado pelo sistema`.

Valor padrão

useStoredProcedure

Descrição

Se for configurado como true, o `OverrideSQL` deverá conter uma referência a um procedimento armazenado que corresponda o código de tratamento com o histórico de contatos e respostas.

Se for configurado como false, o `OverrideSQL`, caso seja usado, deverá ser uma consulta SQL.

Valor padrão

false

Valores válidos

true | false

Tipo

Descrição

O TrackingCodeType associado definido na tabela `UACI_TrackingType` nas tabelas do ambiente de tempo de execução. A menos que você revise a tabela `UACI_TrackingType`, o `Tipo` deve ser 1.

Valor padrão

1

Valores válidos

Um número inteiro definido na tabela `UACI_TrackingType`.

Interact | serviços | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

As propriedades nesta seção definem como o rastreamento de resposta de sessão cruzada corresponde os códigos de oferta ao histórico de contatos e respostas.

SQL

Descrição

Esta propriedade define se o Interact usa o SQL gerado pelo sistema ou SQL customizado definido na propriedade `OverrideSQL`.

Valor padrão

Usar SQL gerado pelo sistema

Valores válidos

Usar SQL gerado pelo sistema | Substituir SQL

OverrideSQL

Descrição

Se você não usar o comando SQL padrão para corresponder o código da oferta ao histórico de contatos e respostas, insira aqui o SQL ou procedimento armazenado.

Esse valor será ignorado se `SQL` estiver configurado como `Usar SQL gerado pelo sistema`.

Valor padrão

useStoredProcedure

Descrição

Se for configurado como true, o `OverrideSQL` deverá conter uma referência a um procedimento armazenado que corresponda o código de oferta ao histórico de contatos e respostas.

Se for configurado como false, o `OverrideSQL`, caso seja usado, deverá ser uma consulta SQL.

Valor padrão

false

Valores válidos

true | false

Tipo

Descrição

O TrackingCodeType associado definido na tabela `UACI_TrackingType` nas tabelas do ambiente de tempo de execução. A menos que você revise a tabela `UACI_TrackingType`, o `Tipo` deve ser 2.

Valor padrão

2

Valores válidos

Um número inteiro definido na tabela UACI_TrackingType.

Interact | serviços | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

As propriedades nesta seção definem como o rastreamento de resposta de sessão cruzada corresponde um código alternativo definido pelo usuário ao histórico de contatos e respostas.

Nome

Descrição

Esta propriedade define o nome para o código alternativo. Ele deve corresponder o valor de Nome na tabela UACI_TrackingType nas tabelas de ambiente de tempo de execução.

Valor padrão

OverrideSQL

Descrição

O comando SQL ou procedimento armazenado para corresponder o código alternativo ao histórico de contatos e respostas por código de oferta ou código de tratamento.

Valor padrão

useStoredProcedure

Descrição

Se configurado como true, o OverrideSQL deverá conter uma referência a um procedimento armazenado que corresponda o código alternativo para o histórico de contatos e respostas.

Se for configurado como false, o OverrideSQL, caso seja usado, deverá ser uma consulta SQL.

Valor padrão

false

Valores válidos

true | false

Tipo

Descrição

O TrackingCodeType associado definido na tabela UACI_TrackingType nas tabelas do ambiente de tempo de execução.

Valor padrão

3

Valores válidos

Um número inteiro definido na tabela UACI_TrackingType.

Interact | serviços | threadManagement | contactAndResponseHist

As propriedades de configuração nesta categoria definem as configurações de gerenciamento de encadeamento para os serviços que coletam dados para as tabelas de migração de histórico de contatos e respostas.

corePoolSize

Descrição

O número de encadeamentos a serem mantidos no conjunto, mesmo se eles estiverem inativos, para coleta de dados do histórico de contatos e respostas.

Valor padrão

5

maxPoolSize

Descrição

O número máximo de encadeamentos a serem mantidos no conjunto para coleta de dados do histórico de contatos e respostas.

Valor padrão

5

keepAliveTimeSecs

Descrição

Quando o número de encadeamentos é maior que o principal, esse é o tempo máximo que os encadeamentos inativos em excesso aguardam novas tarefas antes de terminarem para coletas de dados de histórico de contatos e respostas.

Valor padrão

5

queueCapacity

Descrição

O tamanho da fila usada pelo conjunto de encadeamentos para coleta de dados do histórico de contatos e respostas.

Valor padrão

1000

termWaitSecs

Descrição

No encerramento do servidor de runtime, esse é o número de segundos que deve ser aguardado para que os encadeamentos de serviço concluam a coleta de dados do histórico de contatos e respostas.

Valor padrão

5

Interact | serviços | threadManagement | allOtherServices

As propriedades de configuração nesta categoria definem as configurações de gerenciamento de encadeamento para os serviços que coletam as estatísticas de elegibilidade da oferta, as estatísticas de atividades de evento, as estatísticas de uso de sequência padrão e o log customizado para dados da tabela.

corePoolSize

Descrição

O número de encadeamentos a serem mantidos no conjunto, mesmo se estiverem inativos, para os serviços que coletam as estatísticas de elegibilidade de oferta, as estatísticas de atividades de evento, as estatísticas de uso de sequência padrão e o log customizado para dados da tabela.

Valor padrão

5

maxPoolSize

Descrição

O número máximo de encadeamentos a serem mantidos no conjunto para os serviços que coletam as estatísticas de elegibilidade da oferta, as estatísticas de atividades de evento, as estatísticas de uso de sequência padrão e o log customizado para dados da tabela.

Valor padrão

5

keepAliveTimeSecs

Descrição

Quando o número de encadeamentos for maior que o principal, esse será o tempo máximo que os encadeamentos inativos em excesso aguardam novas tarefas antes de terminarem para os serviços que coletam as estatísticas de elegibilidade da oferta, as estatísticas de atividade de evento, as estatísticas de uso de sequência padrão e log customizado para dados da tabela.

Valor padrão

5

queueCapacity

Descrição

O tamanho da fila usada pelo conjunto de encadeamentos para os serviços que coletam as estatísticas de elegibilidade da oferta, as estatísticas de atividade de evento, as estatísticas de uso de sequência padrão e o log customizado para dados da tabela.

Valor padrão

1000

termWaitSecs

Descrição

No encerramento do servidor de runtime, este é o número de segundos que devem ser aguardados até que os encadeamentos de serviço sejam concluídos para os serviços que coletam as estatísticas de elegibilidade de oferta, as estatísticas de atividade de evento, as estatísticas de uso de sequência padrão e o log customizado para dados da tabela.

Valor padrão

5

Interact | serviços | threadManagement | flushCacheToDB

As propriedades de configuração nesta categoria definem as configurações de gerenciamento de encadeamento que gravam os dados coletados em cache nas tabelas de banco de dados do ambiente de tempo de execução.

corePoolSize

Descrição

O número de encadeamentos a serem mantidos no conjunto para os encadeamentos planejados que gravam dados em cache no armazenamento de dados.

Valor padrão

5

maxPoolSize

Descrição

O número máximo de encadeamentos a serem mantidos no conjunto para os encadeamentos planejados que gravam dados em cache no armazenamento de dados.

Valor padrão

5

keepAliveTimeSecs

Descrição

Quando o número de encadeamentos for maior que o principal, será o tempo máximo que os encadeamentos inativos em excesso aguardarão novas tarefas antes de terminarem para encadeamentos planejados que gravam dados em cache no armazenamento de dados.

Valor padrão

5

queueCapacity

Descrição

O tamanho da fila usada pelo conjunto de encadeamentos para encadeamentos planejados que grava dados em cache no armazenamento de dados.

Valor padrão

1000

termWaitSecs

Descrição

No encerramento do servidor de runtime, esse é o número de segundos que deve ser aguardado para que os encadeamentos de serviço sejam concluídos para encadeamentos planejados que gravam dados em cache no armazenamento de dados.

Valor padrão

5

Interact | serviços | configurationMonitor

As propriedades de configuração nesta categoria permitem ativar ou desativar a integração com o Interact Opportunity Detection sem precisar reiniciar o tempo real do Interact e elas definem o intervalo de pesquisa do valor da propriedade que ativa a integração.

ativar

Descrição

Se for `true`, ativará o serviço que atualiza o valor da propriedade **Interact | serviços | eventPattern | advancedPatterns enableAdvancedPatterns**. Se for `false`, você deverá reiniciar o tempo real do Interact ao alterar o valor da propriedade **Interact | serviços | eventPattern | advancedPatterns | eventPattern enableAdvancedPatterns**.

Valor padrão

Falso

Valores válidos

True | False

refreshIntervalInMinutes

Descrição

Define o intervalo de tempo para pesquisar o valor da propriedade **Interact | serviços | eventPattern | advancedPatterns enableAdvancedPatternss**.

Valor padrão

5

Interact | cacheManagement

Este conjunto de propriedades de configuração define as configurações para selecionar e configurar cada um dos gerenciadores de cache suportados que podem ser usados para melhorar o desempenho do Interact, como o EHCache, que é integrado no armazenamento em cache WebSphere eXtreme Scale da instalação do Interact, que é um complemento opcional ou outro sistema de armazenamento em cache externo.

Use as propriedades de configuração **Interact | cacheManagement | Gerenciadores de cache** para configurar o gerenciador de cache que deseja usar.

Use as propriedades de configuração **Interact | cacheManagement | caches** para especificar qual gerenciador de cache o Interact deve usar para melhorar o desempenho.

Interact | cacheManagement | Gerenciadores de cache

A categoria Gerenciador de cache especifica os parâmetros para as soluções de gerenciamento de cache que você planeja usar com o Interact.

Interact | cacheManagement | Gerenciadores de cache | EHCache

A categoria EHCache especifica os parâmetros para a solução de gerenciamento de cache EHCache, para que seja possível customizá-la para melhorar o desempenho do Interact.

Interact | Gerenciadores de cache | EHCache Scale | Dados de parâmetro

As propriedades de configuração nesta categoria controlam como o sistema de gerenciamento de cache EHCache trabalha para melhorar o desempenho do Interact.

cacheType

Descrição

É possível configurar os servidores de runtime do Interact em um grupo de servidores para usar um endereço multicast para compartilhamento de dados em cache. Isso é mencionado como *cache distribuído*. O parâmetro cacheType especifica se você está usado o mecanismo de armazenamento em cache EHCache integrado no modo **local** (independente) ou **distribuído** (como em um grupo de servidores de runtime).

Nota:

Se você selecionar **Distribuído** como o cacheType, todos os servidores que compartilharem o cache deverão fazer parte do mesmo grupo de servidores. Você também deve ativar o multicast para trabalhar entre todos os membros de um grupo de servidores.

Valor padrão

Local

Valores válidos

Local | Distributed

multicastIPAddress

Descrição

Se você especificar o parâmetro **cacheType** como "distribuído," você estará configurando o cache para operar por multicast entre todos os membros de um grupo de servidores de runtime do Interact. O valor de multicastIPAddress é o endereço IP que todos os servidores do Interact do grupo de servidores usam para receber.

O endereço IP deve ser exclusivo entre seus grupos de servidores.

Valor padrão

230.0.0.1

multicastPort

Descrição

Se você especificar o parâmetro **cacheType** como "distribuído," o parâmetro **multicastPort** indicará a porta que todos os servidores do Interact do grupo de servidores usam para receber.

Valor padrão

6363

overflowToDisk

Descrição

O gerenciador de cache EHCache gerencia as informações da sessão usando a memória disponível. Para ambientes em que o tamanho da sessão é grande devido a um perfil grande, o número de sessões a serem suportadas na memória poderá não ser grande o suficiente para suportar o cenário do cliente. Para situações nesse caso, o EHCache possui um recurso opcional para permitir que informações de cache maiores do que a quantia que pode ser mantida na memória sejam gravadas temporariamente no disco rígido.

Se você configurar a propriedade **overflowToDisk** como "sim," cada Java virtual machine (JVM) poderá manipular mais sessões simultâneas do que a memória sozinha permitiria.

Valor padrão

No

Valores válidos

Não | Sim

diskStore

Descrição

Quando a propriedade de configuração **overflowToDisk** é configurada como `Sim`, essa propriedade de configuração especifica o diretório de disco que conterá as entradas de cache que estiverem com excesso de memória. Se essa propriedade de configuração não existe ou seu valor não é válido, o diretório de disco é criado automaticamente no diretório temporário padrão do sistema operacional.

Valor padrão

None

Valores válidos

Um diretório para o qual o aplicativo da web que hospeda o tempo de execução do Interact tem privilégios de gravação.

(Parameter)

Descrição

Um modelo que pode ser usado para criar um parâmetro customizado a ser usado com o gerenciador de cache. É possível configurar qualquer nome de parâmetro e o valor que ele deve ter.

Para criar um parâmetro customizado, clique em ***(Parameter)*** e preencha o nome e o valor que deseja designar ao parâmetro. Ao clicar em **Salvar mudanças**, o parâmetro criado será incluído na lista, na categoria Dados do parâmetro.

Valor padrão

None

Interact | cacheManagement | Gerenciadores de cache | Extreme Scale

A categoria Extreme Scale especifica os parâmetros para que o adaptador use a solução de gerenciamento de cache WebSphere eXtreme Scale, para que seja possível customizá-lo para melhorar o desempenho do Interact.

ClassName

Descrição

O nome completo da classe que conecta o Interact ao servidor do WebSphere eXtreme Scale. Ele deve ser `com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`.

Valor padrão

`com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`

ClassPath

Descrição

O URI da localização do arquivo `interact_wxs_adapter.jar`, como `file:///IBM/EMM/Interact/lib/interact_wxs_adapter.jar` ou `file:///C:/IBM/EMM/Interact/lib/interact_wxs_adapter.jar`. No entanto, se esse arquivo JAR já estiver incluído no caminho de classe do servidor de aplicativos de hospedagem, este campo deverá ser deixado em branco.

Valor padrão

Em branco

Interact | Gerenciadores de cache | Extreme Scale | Dados de parâmetro

As propriedades de configuração nesta categoria controlam o adaptador WebSphere eXtreme Scale que é incluído opcionalmente com a instalação do Interact. Essas configurações devem ser definidas para cada servidor de runtime do Interact que esteja agindo como um cliente para a grade do servidor WebSphere eXtreme Scale.

catalogPropertyFile

Descrição

O URI do local do arquivo de propriedade usado para iniciar o servidor de catálogos do WebSphere eXtreme Scale. Se o Adaptador Extreme Scale for usado para iniciar o servidor de catálogo, essa propriedade deverá ser configurada. Caso contrário, ele não será usado.

Valor padrão

`file:///C:/depot/Interact/dev/main/extremescale/config/catalogServer.props`

containerPropertyFile

Descrição

O URI do local do arquivo de propriedade usado para iniciar as instâncias do contêiner do WebSphere eXtreme Scale. Se o componente do servidor incluído for usado para iniciar os servidores de contêiner do WebSphere eXtreme Scale, essa propriedade deverá ser configurada. Caso contrário, ele não será usado.

Valor padrão

```
file:///C:/depot/Interact/dev/main/extremescale/config/
containerServer.props
```

deploymentPolicyFile

Descrição

O URI do local do arquivo de políticas de implementação usado para iniciar o servidor de catálogos do WebSphere eXtreme Scale. Se o componente do servidor incluído for usado para iniciar o servidor de catálogos do WebSphere eXtreme Scale, essa propriedade deverá ser configurada. Caso contrário, ele não será usado.

Valor padrão

```
file:///C:/depot/Interact/dev/main/extremescale/config/
deployment.xml
```

objectGridConfigFile

Descrição

O URI do local do arquivo de configuração da grade do objeto usado para iniciar o servidor de catálogo do WebSphere eXtreme Scale e também o componente de cache próximo que executa juntamente com o servidor de runtime do Interact, na mesma Java Virtual Machine (JVM).

Valor padrão

```
file:///C:/depot/Interact/dev/main/extremescale/config/
objectgrid.xml
```

gridName

Descrição

O nome da grade do WebSphere eXtreme Scale que contém todos os caches do Interact.

Valor padrão

InteractGrid

catalogURLs

Descrição

Uma URL contendo o nome do host ou o endereço IP e a porta nos quais o servidor de catálogos do WebSphere eXtreme Scale está recebendo conexões.

Valor padrão

None

(Parameter)

Descrição

Um modelo que pode ser usado para criar um parâmetro customizado a ser usado com o gerenciador de cache. É possível configurar qualquer nome de parâmetro e o valor que ele deve ter.

Para criar um parâmetro customizado, clique em ***(Parameter)*** e preencha o nome e o valor que deseja designar ao parâmetro. Ao clicar em **Salvar mudanças**, o parâmetro criado será incluído na lista, na categoria Dados do parâmetro.

Valor padrão

None

Interact | caches

Use este conjunto de propriedades de configuração para especificar qual gerenciador de cache suportado deseja usar para melhorar o desempenho do Interact, como o armazenamento em cache Ehcache ou WebSphere eXtreme Scale e para configurar propriedades de cache específicas para o servidor de runtime que estiver sendo configurado.

Isso inclui os caches de armazenamento de dados de sessão, os estados de padrão de evento e os resultados de segmentação. Ao ajustar essas configurações, é possível especificar qual solução de cache é usada para cada tipo de armazenamento em cache e especificar configurações individuais para controlar como o cache funciona.

Interact | cacheManagement | caches | InteractCache

A categoria InteractCache configura o armazenamento em cache para todos os objetos de sessão, incluindo os dados do perfil, os resultados da segmentação, os tratamentos entregues mais recentemente, os parâmetros transmitidos por meio dos métodos da API e outros objetos usados pelo tempo de execução do Interact.

A categoria InteractCache é necessária para que o Interact funcione adequadamente.

A categoria InteractCache também pode ser configurada por meio de uma configuração de EHCache externa para definições que não são suportadas no **Interact | cacheManagement | Caches**. Se o EHCache for usado, você deverá assegurar-se de que o InteractCache esteja configurado adequadamente.

CacheManagerName

Descrição

O nome do gerenciador de cache que manipula o cache do Interact. O valor que você digitar aqui deverá ser um dos gerenciadores de cache definidos nas propriedades de configuração **Interact | cacheManagement | Gerenciadores de cache**, como `EHCache` ou `Extreme Scale`.

Valor padrão

EHCache

Valores válidos

Qualquer gerenciador de cache definido na propriedade de configuração **Interact | cacheManagement | Gerenciadores de cache**.

maxEntriesInCache

Descrição

O número máximo de objetos de dados da sessão para armazenar nesse cache. Quando o número máximo de objetos de dados da sessão for atingido e dados para uma sessão adicionais precisarem ser armazenados, o objeto usado menos recentemente será excluído.

Valor padrão

100000

Valores válidos

Número inteiro maior que 0.

timeoutInSecs

Descrição

O tempo em segundos que decorreram desde que um objeto de dados de sessão foi usado ou atualizado, usados para determinar quando o objeto é removido da cache.

Valor padrão

300

Valores válidos

Número inteiro maior que 0.

Interact | caches | cache do Interact | dados de parâmetro

As propriedades de configuração nesta categoria controlam o Cache do Interact que é usado automaticamente pela instalação do Interact. Essas configurações devem ser configuradas individualmente para cada servidor de runtime do Interact.

asyncIntervalMillis

Descrição

O tempo em milissegundos que o gerenciador de cache EHCache deve aguardar antes de replicar mudanças para outras instâncias de tempo de execução do Interact. Se o valor não for positivo, essas mudanças serão replicadas de forma síncrona.

Esta propriedade de configuração não é criada, por padrão. Se você criar essa propriedade, ela somente será usada quando o EHCache for o gerenciador de cache e quando a propriedade **cacheType** do ehCache for configurada como `distribuído`.

Valor padrão

None.

(Parameter)

Descrição

Um modelo que pode ser usado para criar um parâmetro customizado a ser usado com o cache do Interact. É possível configurar qualquer nome de parâmetro e o valor que ele deve ter.

Para criar um parâmetro customizado, clique em ***(Parameter)*** e preencha o nome e o valor que deseja designar ao parâmetro. Ao clicar em **Salvar mudanças**, o parâmetro criado será incluído na lista, na categoria Dados do parâmetro.

Valor padrão

None

Interact | cacheManagement | caches | PatternStateCache

A categoria PatternStateCache é usada para hospedar os estados de padrões de eventos e as regras de supressão de oferta em tempo real. Por padrão, este cache é configurado como cache read-through e write-through, para que o Interact tente usar o primeiro padrão de evento do cache e os dados de supressão de oferta. Se a entrada solicitada não existir no cache, a implementação de cache a carregará a partir da origem de dados, por meio da configuração de JNDI ou diretamente usando uma conexão JDBC.

Para usar uma conexão JNDI, o Interact conecta-se a um provedor de origem de dados existente que tenha sido definido por meio do servidor especificado usando o nome JNDI, a URL e assim por diante. Para uma conexão JDBC, você deve fornecer um conjunto de configurações JDBC que incluam o nome de classe do driver JDBC, a URL do banco de dados e informações sobre autenticação.

Observe que se você definir várias origens JNDI e JDBC, será usada a primeira origem JNDI ativada e se não houver origens JNDI ativadas, será usada a primeira origem JDBC ativada.

A categoria PatternStateCache é necessária para que o Interact funcione adequadamente.

A categoria PatternStateCache também pode ser configurada por meio de uma configuração do EHCache externa para definições que não são suportadas no **Interact | cacheManagement | Caches**. Se o EHCache for usado, você deverá assegurar-se de que o PatternStateCache esteja configurado adequadamente.

CacheManagerName

Descrição

O nome do gerenciador de cache que manipula o cache de estado de padrão do Interact. O valor que você digitar aqui deverá ser um dos gerenciadores de cache definidos nas propriedades de configuração **Interact | cacheManagement | Gerenciadores de cache**, como `EHCache` ou `Extreme Scale`.

Valor padrão

EHCache

Valores válidos

Qualquer gerenciador de cache definido na propriedade de configuração **Interact | cacheManagement | Gerenciadores de cache**.

maxEntriesInCache

Descrição

O número máximo de estados de padrão de evento para armazenar neste cache. Quando o número máximo de estados de padrão de evento for

atingido e os dados de um estado de padrão de evento adicional precisarem ser armazenados, o objeto usado menos recentemente será excluído.

Valor padrão

100000

Valores válidos

Número inteiro maior que 0.

timeoutInSecs

Descrição

Especifica o período de tempo, em segundos, para um objeto de estado de padrão de evento atingir o tempo limite no cache de estado de padrão de evento. Quando esse objeto de estado ficar inativo no cache pelo número de segundos de `timeoutInSecs`, ele poderá ser ejetado do cache com base na regra usada menos recentemente. Observe que o valor dessa propriedade deve ser maior que o definido na propriedade `sessionTimeoutInSecs`.

Valor padrão

300

Valores válidos

Número inteiro maior que 0.

Interact | caches | PatternStateCache | dados de parâmetro:

As propriedades de configuração nesta categoria controlam o Cache de estado de padrão usado para hospedar os estados de padrões de evento e as regras de supressão de oferta em tempo real.

(Parameter)

Descrição

Um modelo que pode ser usado para criar um parâmetro customizado a ser usado com o Cache de estado de padrão. É possível configurar qualquer nome de parâmetro e o valor que ele deve ter.

Para criar um parâmetro customizado, clique em ***(Parameter)*** e preencha o nome e o valor que deseja designar ao parâmetro. Ao clicar em **Salvar mudanças**, o parâmetro criado será incluído na lista, na categoria Dados do parâmetro.

Valor padrão

None

Interact | cacheManagement | caches | PatternStateCache | loaderWriter:

A categoria **loaderWriter** contém a configuração do carregador que interage com repositórios externos para a recuperação e persistência de padrões de evento.

className

Descrição

O nome da classe completo para esse carregador. Essa classe deve estar em conformidade com o requisito do gerenciador de cache escolhido.

Valor padrão

`com.unicacorp.interact.cache.ehcache.loaderwriter.PatternStateEHCacheLoaderWriter`

Valores válidos

Um nome da classe completo.

classPath

Descrição

O caminho para o arquivo de classe do carregador. Se você deixar este valor em branco ou se a entrada for inválida, o caminho de classe usado para executar o Interact será usado.

Valor padrão

None

Valores válidos

Um caminho de classe válido.

writeMode

Descrição

Especifica o modo para o gravador persistir os estados de padrão de evento novos ou atualizados no cache. Os valores válidos são:

- WRITE_THROUGH. Sempre que houver uma nova entrada ou que uma entrada existente for atualizada, essa entrada será gravada nos repositórios imediatamente.
- WRITE_BEHIND. O gerenciador de cache aguarda um tempo para coletar algumas mudanças e, em seguida, as persiste nos repositórios em um lote.

Valor padrão

WRITE_THROUGH

Valores válidos

WRITE_THROUGH ou WRITE_BEHIND.

batchSize

Descrição

O número máximo de objetos de estado do padrão de evento que o gravador persistirá em um lote. Essa propriedade somente é usada quando o **writeMode** é configurado como `WRITE_BEHIND`.

Valor padrão

100

Valores válidos

Valor de número inteiro.

maxDelayInSecs

Descrição

O tempo máximo em segundos que o gerenciador de cache aguarda antes que um objeto de estado de padrão de evento seja persistido. Essa propriedade somente é usada quando o **writeMode** é configurado como `WRITE_BEHIND`.

Valor padrão

5

Valores válidos

Valor de número inteiro.

Interact | Caches | PatternStateCache | loaderWriter | dados de parâmetro:

As propriedades de configuração nesta categoria controlam o carregador de Cache de estado de padrão.

(Parameter)

Descrição

Um modelo que pode ser usado para criar um parâmetro customizado a ser usado com o carregador de Cache de estado de padrão. É possível configurar qualquer nome de parâmetro e o valor que ele deve ter.

Para criar um parâmetro customizado, clique em ***(Parameter)*** e preencha o nome e o valor que deseja designar ao parâmetro. Ao clicar em **Salvar mudanças**, o parâmetro criado será incluído na lista, na categoria Dados do parâmetro.

Valor padrão

None

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jndiSettings:

A categoria **jndiSettings** contém a configuração da origem de dados JNDI que o carregador usará para se comunicar com o banco de dados de suporte. Para criar um novo conjunto de configurações de JNDI, expanda a categoria **jdniSettings** e clique na propriedade ***(jndiSetting)***.

(jndiSettings)

Nota: Quando o WebSphere Application Server é usado, o loaderWriter não é conectado ao **jndiSettings**.

Descrição

Ao clicar nesta categoria, um formulário aparecerá. Para definir uma origem de dados JNDI, preencha os seguintes valores:

- **Nome da nova categoria** é o nome que deseja usar para identificar esta conexão JNDI.
- **enabled** permite indicar se deseja que essa conexão JNDI seja disponibilizada para uso ou não. Configure como `True` para novas conexões.

- **jdniName** é o nome JNDI que já foi definido na origem de dados quando ela foi configurada.
- **providerUrl** é a URL para localizar essa origem de dados JNDI. Se você deixar esse campo em branco, o URL do aplicativo da web que hospeda o tempo de execução do Interact será usada.
- **Factory de contexto inicial** é o nome completo da classe do factory de contexto inicial para conexão com o provedor JNDI. Se o aplicativo da web que hospeda o tempo de execução do Interact for usado para o **providerUrl**, deixe esse campo em branco.

Valor padrão

None.

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jdbcSettings:

A categoria **jdbcSettings** contém a configuração das conexões JDBC que o carregador usará para se comunicar com o banco de dados de suporte. Para criar um novo conjunto de configurações de JDBC, expanda a categoria **jdbcSettings** e clique na propriedade ***(jdbcSetting)***.

(jdbcSettings)

Descrição

Ao clicar nesta categoria, um formulário aparecerá. Para definir uma origem de dados JDBC, preencha os seguintes valores:

- **Nome da nova categoria** é o nome que deseja usar para identificar esta conexão JDBC.
- **enabled** permite indicar se deseja que essa conexão JDBC seja disponibilizada para uso ou não. Configure como `True` para novas conexões.
- **driverClassName** é o nome completo da classe do driver JDBC. Esta classe deve existir no caminho de classe configurado para iniciar o servidor de cache de hosting.
- **databaseUrl** é a URL para localizar esta origem de dados JDBC.
- **asmUser** é o nome do usuário do IBM EMM que foi configurado com as credenciais para se reconectar ao banco de dados nesta conexão JDBC.
- **asmDataSource** é o nome da origem de dados do IBM EMM que foi configurada com as credenciais para se reconectar ao banco de dados nesta conexão JDBC.
- **maxConnection** é o número máximo de conexões simultâneas que podem ser feitas com o banco de dados nesta conexão JDBC.

Valor padrão

None.

Interact | triggeredMessage

As propriedades de configuração nesta categoria definem as configurações para todas as mensagens acionadas e para o canal de entrega da oferta.

backendProcessIntervalMin

Descrição

Esta propriedade define o período de tempo, em minutos, que o encadeamento de backend carrega e processos entregas de oferta atrasadas. Esse valor deve ser um número inteiro. Se o valor for zero ou negativo, o processo de backend estará desativado.

Valores válidos

Um número inteiro positivo

autoLogContactAfterDelivery

Descrição

Se esta propriedade for configurada como verdadeira, um evento de contato será postado automaticamente assim que essa oferta for enfileirada para entrega atrasada ou se essa propriedade for configurada como false, nenhum evento de contato será automaticamente postado para as ofertas de saída. Este é o comportamento padrão.

Valores válidos

True | False

waitForFlowchart

Descrição

Esta propriedade determina se o fluxograma deve aguardar a conclusão da segmentação atualmente em execução e o comportamento se essa espera atingir o tempo limite.

DoNotWait: o processamento de uma mensagem acionada é iniciado independente de a segmentação estar em execução atualmente ou não. No entanto, se os segmentos são usados na regra de elegibilidade e/ou NextBestOffer for selecionado como o método de seleção de oferta, a execução da TM ainda aguarda.

OptionalWait : o processamento de uma mensagem acionada aguarda até que a segmentação atualmente em execução seja concluída ou atinja o tempo limite. Se a espera atingir o tempo limite, um aviso será registrado e o processamento dessa mensagem acionada continuará. Este é o padrão.

MandatoryWait: o processamento de uma mensagem acionada aguarda até que a segmentação atualmente em execução seja concluída ou atinja o tempo limite. Se a espera atingir o tempo limite, um erro será registrado e o processamento desta mensagem acionada será interrompido.

Valores válidos

DoNotWait | OptionalWait | MandatoryWait

Interact | triggeredMessage | offerSelection

As propriedades de configuração nesta categoria definem as configurações para a seleção da oferta nas mensagens acionadas.

maxCandidateOffers

Descrição

Esta propriedade define o número máximo de ofertas elegíveis que o mecanismo retorna para obter a melhor oferta para entrega. Há uma chance de que nenhuma dessas ofertas elegíveis retornada possa ser enviada com base no canal selecionado. Quanto mais ofertas candidatas

existirem, menos esse caso ocorre. Entretanto, muitas ofertas candidatas pode aumentar o tempo de processamento.

Valores válidos

Um número inteiro positivo

defaultCellCode

Descrição

Se a oferta entregue for o resultado da avaliação de uma regra estratégica ou de um registro orientado a tabela, há uma célula de destino associada a ele e as informações desta célula são usadas em toda a criação de log relacionada. No entanto, se uma lista de ofertas específicas for usada como a entrada para a seleção da oferta, nenhuma célula de destino estará disponível. Nesse caso, o valor dessa definição de configuração será usado. Você deve certificar-se de que essa célula de destino e sua campanha estejam incluídos na implementação. O método mais fácil de fazer isso é incluir a célula em uma estratégia implementada.

Interact | triggeredMessage | dispatchers

As propriedades de configuração nesta categoria definem as configurações para todos os dispatchers nas mensagens acionadas.

dispatchingThreads

Descrição

Esta propriedade define o número de encadeamentos que o mecanismo usa para chamar de forma assíncrona os dispatchers. Se o valor for 0 ou um número negativo, a chamada de dispatchers será síncrona. O valor padrão é 0.

Valores válidos

Um número inteiro

Interact | triggeredMessage | dispatchers | <dispatcherName>

As propriedades de configuração nesta categoria definem as configurações para um dispatcher específico nas mensagens acionadas.

category name

Descrição

Esta propriedade define o nome deste dispatcher. O nome deve ser exclusivo entre todos os dispatchers.

type

Descrição

Esta propriedade define o tipo dispatcher.

Valores válidos

InMemoryQueue | JMSQueue | Custom

Nota: Se você usar JMSQueue ou Custom, para integrar o Interact com o IBM MQ, o tempo de execução do Interact deve estar no servidor de aplicativos com o JDK 1.7. Para o WebSphere e WebLogic, recomenda-se usar a versão de fix pack do JDK fornecida mais recentemente.

O JMSQueue suporta apenas o WebLogic. Não é possível usar o JMSQueue se você usar o WebSphere Application Server.

className

Descrição

Esta propriedade define o nome completo de classe desta implementação do dispatcher. Se o tipo for InMemoryQueue, o valor deverá estar vazio. Se o tipo for custom, essa configuração deverá ter o valor `com.unicacorp.interact.eventhandler.triggeredmessage.dispatchers.IBMMQDispatcher`.

classPath

Descrição

Esta propriedade define a URL para o arquivo JAR que inclui a implementação desse dispatcher.

Se o tipo for custom, essa configuração deverá ter o valor `file://<Interact_HOME>/lib/interact_ibmmqdispatcher.jar;file://<Interact_HOME>/lib/com.ibm.mq.allclient.jar;file://<Interact_HOME>/lib/jms.jar`

Interact | triggeredMessage | dispatchers | <dispatcherName> | Dados do Parâmetro

As propriedades de configuração nesta categoria definem parâmetros para um dispatcher específico nas mensagens acionadas.

É possível escolher entre três tipos de dispatchers. InMemoryQueue é o dispatcher interno para o Interact. Custom é usado para o IBM MQ. JMSQueue é usado para se conectar a um provedor JMS por meio do JNDI.

category name

Descrição

Esta propriedade define o nome deste parâmetro. O nome deve ser exclusivo entre todos os parâmetros para esse dispatcher.

value

Descrição

Esta propriedade define os parâmetros, no formato de pares de nome e valor, necessários para este dispatcher.

Nota: Todos os parâmetros para mensagens do acionador fazem distinção entre maiúsculas e minúsculas e devem ser inseridos conforme mostrado aqui.

Se o tipo for InMemoryQueue, o parâmetro a seguir é suportado.

- queueCapacity: opcional. O máximo de ofertas que podem estar aguardando na fila para serem despachadas. Quando especificada, esta propriedade deve ser um número inteiro positivo. Se não especificado ou inválido, o valor padrão (1000) é usado.

Se o tipo for Custom, os parâmetros a seguir são suportados.

- providerUrl: <hostname>:port (distinção entre maiúsculas e minúsculas)

- queueManager: o nome do gerenciador de filas que foi criado no servidor IBM MQ.
- messageQueueName: o nome da fila de mensagem que foi criada no servidor IBM MQ.
- enableConsumer: esta propriedade deve ser configurada como verdadeira.
- asmUserforMQAuth: o nome de usuário para efetuar login no servidor. É necessário(a) quando o servidor força a autenticação. Caso contrário, não deve ser especificado(a).
- authDS: a senha associada com o nome de usuário para efetuar login no servidor. É necessário(a) quando o servidor força a autenticação. Caso contrário, não deve ser especificado(a).

Se o tipo for JMSQueue, o parâmetro a seguir é suportado.

- providerUrl: a URL para o provedor JNDI (distinção entre maiúsculas e minúsculas).
- connectionFactoryJNDI: o nome JNDI do connection factory de JMS.
- messageQueueJNDI: o nome JNDI da fila JMS para a qual as mensagens acionadas são enviadas e recuperadas.
- enableConsumer: se um consumidor destas mensagens acionadas deve ser iniciado no Interact. Esta propriedade deve estar configurada como verdadeira. Se não especificada, o valor padrão (falso) será usado.
- initialContextFactory: o nome completo da classe de factory de contexto inicial da JNDI. SE você usar o WebLogic, o valor deste parâmetro deverá ser `weblogic.jndi.WLInitialContextFactory`.

Interact | triggeredMessage | gateways | <gatewayName>

As propriedades de configuração esta categoria definem as configurações para um gateway específico nas mensagens acionadas.

O Interact não suporta várias instâncias do mesmo gateway. Todos os arquivos de configuração de gateway devem estar acessíveis a partir de cada nó de tempo de execução do Interact. No caso de uma configuração distribuída, certifique-se de que os arquivos de gateway sejam mantidos em um local compartilhado.

category name

Descrição

Esta propriedade define o nome deste gateway. Ela deve ser exclusiva entre todos os gateways.

className

Descrição

Esta propriedade define o nome completo de classe desta implementação do gateway.

classPath

Descrição

Essa propriedade define o URI do arquivo JAR que inclui a implementação desse gateway. Se deixada vazia, o caminho de classe do aplicativo Interact de hosting será usado.

Por exemplo, em um sistema Windows, se o arquivo JAR do gateway estiver disponível no diretório `C:\IBM\EMM\EmailGateway\IBM_Interact_OMO_OutboundGateway_Silverpop_1.0\lib\OMO_OutboundGateway_Silverpop.jar`, o classPath deverá ser `file:///C:/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`. Em um sistema Unix, se o arquivo JAR do gateway estiver disponível no diretório `/opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`, o caminho de classe deverá ser `file:///opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`.

Interact | triggeredMessage | gateways | <gatewayName> | Dados do Parâmetro

As propriedades de configuração nesta categoria definem parâmetros para um gateway específico nas mensagens acionadas.

category name

Descrição

Esta propriedade define o nome deste parâmetro. O nome deve ser exclusivo entre todos os parâmetros para esse gateway.

value

Descrição

Esta propriedade define os parâmetros, no formato de pares de nome e valor, necessários para este gateway. Para todos os gateways, os parâmetros a seguir são suportados.

Nota: Todos os parâmetros para mensagens do acionador fazem distinção entre maiúsculas e minúsculas e devem ser inseridos conforme mostrado aqui.

- validationTimeoutMillis: a duração, em milissegundos, que a validação de uma oferta por meio desse gateway atinge o tempo limite. O valor padrão é 500.
- deliveryTimeoutMillis: a duração, em milissegundos, que a entrega de uma oferta usando esse gateway atinge o tempo limite. O valor padrão é 1000.

Interact | triggeredMessage | canais

As propriedades de configuração nesta categoria definem as configurações para todos os canais nas mensagens acionadas.

type

Descrição

Esta propriedade define o nó raiz para configurações relacionadas a um gateway específico. O padrão usar o seletor de canais integrado, que é baseado na lista de canais definida na UI das mensagens acionadas. Se Default for selecionado, os valores className e classPath deverão ser deixados em branco. O cliente usa a implementação do cliente do IChannelSelector.

Valores válidos

Default | Custom

className

Descrição

Esta propriedade define o nome completo de classe da implementação do cliente do seletor de canais. Essa configuração é necessária se o tipo for Custom.

classPath

Descrição

Esta propriedade define a URL para o arquivo JAR que inclui a implementação do cliente do seletor de canais. Se deixada vazia, o caminho de classe do aplicativo Interact de hosting será usado.

Interact | triggeredMessage | canais | Dados do Parâmetro

As propriedades de configuração nesta categoria definem parâmetros para um canal específico nas mensagens acionadas.

category name

Descrição

Esta propriedade define o nome deste parâmetro. O nome deve ser exclusivo entre todos os parâmetros para esse canal.

value

Descrição

Esta propriedade define os parâmetros, no formato de pares de nome e valor, necessários para o seletor de canal.

Se você usar **Canais Preferidos do Cliente** para o seu canal, deve-se criar

Interact | triggeredMessage | canais | <channelName>

As propriedades de configuração nesta categoria definem parâmetros para um canal específico nas mensagens acionadas.

category name

Descrição

Esta propriedade define o nome do canal por meio do qual as ofertas são enviadas. Ela deve corresponder àquelas definidas no tempo de design sob **Campaign | partições | <partição[N]> | Interact | outboundChannels**.

Interact | triggeredMessage | canais | <channelName> | <handlerName>

As propriedades de configuração nesta categoria definem as configurações para um manipulador específico nas mensagens acionadas que é usado para ofertas enviadas.

category name

Descrição

Esta propriedade define o nome do manipulador que o canal usará para enviar ofertas.

dispatcher

Descrição

Esta propriedade define o nome do dispatcher através do qual este manipulador usa ofertas enviadas para o gateway. Ela deve ser uma daquelas definidas sob **interact | triggeredMessage | dispatchers**.

gateway

Descrição

Esta propriedade define o nome do gateway para o qual esse manipulador envia ofertas no final. Ela deve ser uma daquelas definidas sob **interact | triggeredMessage | gateways**.

mode

Descrição

Esta propriedade define o modo de uso deste manipulador. Se Failover for selecionado, esse manipulador será usado apenas quando todos os manipuladores com prioridade mais alta definidos dentro desse canal falharem ao enviar ofertas. Se Complemento for selecionado, esse manipulador será usado, não importando se outros manipuladores tenham enviado ofertas com sucesso.

priority

Descrição

Esta propriedade define a prioridade deste manipulador. Primeiro, o mecanismo tenta usar o manipulador com a mais alta prioridade para o envio de ofertas.

Valores válidos

Qualquer número inteiro

Padrão

100

Interact | ETL | patternStateETL

As propriedades de configuração nesta categoria definem as configurações para o processo ETL.

Nome da nova categoria

Descrição

Forneça um nome que identifique exclusivamente essa configuração. Observe que você deverá fornecer esse nome exato quando executar o processo ETL independente. Por conveniência em especificar esse nome na linha de comandos, você pode evitar um nome que contenha espaços ou pontuação, como `ETLProfile1`.

runOnceADay

Descrição

Determina se o processo ETL independente nessa configuração deve ser executado uma vez por dia. As respostas válidas são **Sim** ou **Não**. Se você responder **Não** aqui, o **processSleepIntervalInMinutes** determinará o planejamento de execução para o processo.

preferredStartTime

Descrição

O horário preferencial no qual o processo ETL independente deve ser iniciado. Especifique o horário no formato HH:MM:SS AM/PM, como em `01:00:00 AM`.

preferredEndTime

Descrição

O horário preferencial no qual o processo ETL independente deve ser parado. Especifique o horário no formato HH:MM:SS AM/PM, como em `08:00:00 AM`.

processSleepIntervalInMinutes

Descrição

Se você não tiver configurado o processo ETL independente para ser executado uma vez ao dia (conforme especificado na propriedade **runOnceADay**), essa propriedade especificará o intervalo entre as execuções do processo ETL. Por exemplo, se você especificar 15 aqui, o processo ETL independente aguardará 15 minutos após a parada da execução antes de iniciar o processo novamente.

maxJDBCInsertBatchSize

Descrição

O número máximo de registros de um lote de JDBC antes da confirmação da consulta. Por padrão, é configurado como 5000. Observe que esse não é o número máximo de registros dos processos ETL em uma iteração. Durante cada iteração, o ETL processa todos os registros disponíveis na tabela UACI_EVENTPATTERNSTATE. No entanto, todos esses registros são divididos em chunks do **maxJDBCInsertSize**.

maxJDBCFetchBatchSize

Descrição

O número máximo de registros de um lote de JDBC a serem buscados no banco de dados temporário.

Você pode precisar aumentar esse valor para ajustar o desempenho do ETL.

communicationPort

Descrição

A porta de rede na qual o processo ETL independente atende uma solicitação de parada. Sob circunstâncias normais, não deve haver nenhuma razão para alterá-la do valor padrão.

queueLength

Descrição

Um valor usado para o ajuste de desempenho. Coleções de dados de estado de padrão são buscadas e transformadas em objetos que são incluídos em uma fila para serem processados e gravados no banco de dados. Essa propriedade controla o tamanho da fila.

completionNotificationScript

Descrição

Especifica o caminho absoluto para que um script seja executado quando o processo ETL for concluído. Se você especificar um script, três argumentos são transmitidos para o script de notificação de conclusão: horário de início, horário de término e número total de registros de padrão de evento processados. O horário de início e o horário de término são valores numéricos que representam o número de milissegundos decorridos desde 1970.

Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS

As propriedades de configuração nesta categoria definem as configurações para o DS de tempo de execução de ETL.

type

Descrição

Uma lista dos tipos de banco de dados suportados para a origem de dados que está sendo definida.

dsname

Descrição

O nome JNDI da origem de dados. Esse nome também deve ser usado na configuração de origem de dados do usuário para assegurar que o usuário tenha acesso às origens de dados de tempo de execução e de destino.

driver

Descrição

O nome do driver JDBC para nós, como qualquer um dos seguintes:

Oracle: `oracle.jdbc.OracleDriver`

Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2: `com.ibm.db2.jcc.DB2Driver`

serverURL

Descrição

A URL da origem de dados, como qualquer uma das seguintes:

Oracle: `jdbc:oracle:thin:@ <your_db_host>:<your_db_port>:<your_db_service_name>`

Microsoft SQL Server: `jdbc:sqlserver:// <your_db_host>:<your_db_port> ;databaseName= <your_db_name>`

IBM DB2: `jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>`

connectionpoolSize

Descrição

Um valor indicando o tamanho do conjunto de conexões, fornecido para o ajuste de desempenho. Os dados de estado padrão são lidos e transformados simultaneamente, dependendo das conexões com o banco de dados disponíveis. Aumentar o tamanho do conjunto de conexões permite mais conexões simultâneas com o banco de dados, sujeitas a limitações de memória e de recursos de leitura/gravação do banco de dados. Por exemplo, se esse valor for configurado como 4, quatro tarefas serão executadas simultaneamente. Se houver uma grande quantia de dados, poderá ser necessário aumentar esse valor para um número como 10 ou 20, desde que haja memória e desempenho do banco de dados suficientes disponíveis.

esquema

Descrição

O nome do esquema do banco de dados ao qual essa configuração está conectada.

connectionRetryPeriod

Descrição

A propriedade `ConnectionRetryPeriod` especifica o tempo em segundos em que o Interact tenta novamente automaticamente a solicitação de conexão com o banco de dados no caso de falha. O Interact tenta reconectar-se automaticamente ao banco de dados durante esse período de tempo antes de relatar um erro ou uma falha do banco de dados. Se o valor estiver configurado como 0, o Interact tentará novamente indefinidamente. Se o valor estiver configurado como -1, não ocorrerá nenhuma nova tentativa.

connectionRetryDelay

Descrição

A propriedade `ConnectionRetryDelay` especifica o tempo em segundos que o Interact espera antes de tentar se reconectar ao banco de dados após uma falha. Se o valor estiver configurado como -1, não ocorrerá nenhuma nova tentativa.

Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS

As propriedades de configuração nesta categoria definem as configurações para o DS de destino de ETL.

type

Descrição

Uma lista dos tipos de banco de dados suportados para a origem de dados que está sendo definida.

dsname

Descrição

O nome JNDI da origem de dados. Esse nome também deve ser usado na configuração de origem de dados do usuário para assegurar que o usuário tenha acesso às origens de dados de tempo de execução e de destino.

driver

Descrição

O nome do driver JDBC para nós, como qualquer um dos seguintes:

Oracle: `oracle.jdbc.OracleDriver`

Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2: `com.ibm.db2.jcc.DB2Driver`

serverURL

Descrição

A URL da origem de dados, como qualquer uma das seguintes:

Oracle: `jdbc:oracle:thin:@ <your_db_host>:<your_db_port>:<your_db_service_name>`

Microsoft SQL Server: `jdbc:sqlserver:// <your_db_host>:<your_db_port> ;databaseName= <your_db_name>`

IBM DB2: `jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>`

connectionpoolSize

Descrição

Um valor indicando o tamanho do conjunto de conexões, fornecido para o ajuste de desempenho. Os dados de estado padrão são lidos e transformados simultaneamente, dependendo das conexões com o banco de dados disponíveis. Aumentar o tamanho do conjunto de conexões permite mais conexões simultâneas com o banco de dados, sujeitas a limitações de memória e de recursos de leitura/gravação do banco de dados. Por exemplo, se esse valor for configurado como 4, quatro tarefas serão executadas simultaneamente. Se houver uma grande quantia de dados, poderá ser necessário aumentar esse valor para um número como 10 ou 20, desde que haja memória e desempenho do banco de dados suficientes disponíveis.

esquema

Descrição

O nome do esquema do banco de dados ao qual essa configuração está conectada.

connectionRetryPeriod

Descrição

A propriedade ConnectionRetryPeriod especifica o tempo em segundos em que o Interact tenta novamente automaticamente a solicitação de conexão com o banco de dados no caso de falha. O Interact tenta reconectar-se automaticamente ao banco de dados durante esse período de tempo antes de relatar um erro ou uma falha do banco de dados. Se o valor estiver configurado como 0, o Interact tentará novamente indefinidamente. Se o valor estiver configurado como -1, não ocorrerá nenhuma nova tentativa.

connectionRetryDelay

Descrição

A propriedade ConnectionRetryDelay especifica o tempo em segundos que o Interact espera antes de tentar se reconectar ao banco de dados após uma falha. Se o valor estiver configurado como -1, não ocorrerá nenhuma nova tentativa.

Interact | ETL | patternStateETL | <patternStateETLName> | relatório

As propriedades de configuração nesta categoria definem as configurações para o processo de agregação de relatório ETL.

ativar

Descrição

Ativar ou desativar a integração de relatório com o ETL. Esta propriedade é configurada como desativar, por padrão.

Se for configurada como desativar, essa propriedade desativará as atualizações na Tabela UARI_DELTA_PATTERNS. Ele não desativar o relatório completamente.

Nota: Para desativar a integração de relatório com ETL, você também deverá alterar o acionador TR_AGGREGATE_DELTA_PATTERNS para desativar na tabela de migração UACI_ETLPATTERNSTATERUN.

retryAttemptsIfAggregationRunning

Descrição

O número de vezes que o ETL tentará verificar se a agregação de relatório está concluída se o sinalizador de bloqueio estiver configurado. Esta propriedade é configurada como 3, por padrão.

sleepBeforeRetryDurationInMinutes

Descrição

Tempo de suspensão, em minutos, entre as tentativas consecutivas. Esta propriedade é configurada como 5 minutos, por padrão.

aggregationRunningCheckSql

Descrição

Essa propriedade permite definir um SQL customizado, que pode ser executado para descobrir se o sinalizador de bloqueio de agregação de relatório está configurado. Por padrão, essa propriedade fica vazia.

Quando essa propriedade não estiver configurada, o ETL executará o seguinte SQL para obter o sinalizador de bloqueio.

```
select count(1) AS ACTIVERUNS from uari_pattern_lock where islock='Y'
=> If ACTIVERUNS is > 0, lock is set
```

aggregationRunningCheck

Descrição

Ative ou desative a verificação se a agregação de relatório estiver em execução antes que a execução do ETL seja realizada. Essa propriedade é configurada como ativar, por padrão.

Apêndice C. Propriedades de configuração do ambiente de design do Interact

Esta seção descreve todas as propriedades de configuração para o ambiente de design do Interact.

Campanha | partições | partição[n] | relatórios

A propriedade **Campanha | partições | partição[n] | relatórios** define os diferentes tipos de pastas para relatórios.

offerAnalysisTabCachedFolder

Descrição

A propriedade `offerAnalysisTabCachedFolder` especifica a localização da pasta que contém a especificação para os relatórios burst (expandidos) de ofertas listados na guia Análise quando você os acessa clicando no link Análise na área de janela de navegação. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

Descrição

A propriedade `segmentAnalysisTabOnDemandFolder` especifica a localização da pasta que contém os relatórios de segmentos listados na guia Análise de um segmento. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

Descrição

A propriedade `offerAnalysisTabOnDemandFolder` especifica a localização da pasta que contém os relatórios de ofertas listados na guia de Análise de uma oferta. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

segmentAnalysisTabCachedFolder

Descrição

A propriedade `segmentAnalysisTabCachedFolder` especifica a localização da pasta que contém a especificação para relatórios burst (expandidos) de

segmentos listados na guia Análise quando você os acessa clicando no link Análise na área de janela de navegação. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']
```

analysisSectionFolder

Descrição

A propriedade `analysisSectionFolder` especifica a localização da pasta raiz na qual as especificações de relatório são armazenadas. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign']
```

campaignAnalysisTabOnDemandFolder

Descrição

A propriedade `campaignAnalysisTabOnDemandFolder` especifica a localização da pasta que contém os relatórios de campanha listados na guia Análise de uma campanha. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']
```

campaignAnalysisTabCachedFolder

Descrição

A propriedade `campaignAnalysisTabCachedFolder` especifica a localização da pasta que contém a especificação para os relatórios burst (expandidos) de campanha listados na guia Análise quando você os acessa clicando no link Análise na área de janela de navegação. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']/folder[@name='cached']
```

campaignAnalysisTabEmessageOnDemandFolder

Descrição

A propriedade `campaignAnalysisTabEmessageOnDemandFolder` especifica a localização da pasta que contém os relatórios do eMessage listados na guia Análise de uma campanha. O caminho é especificado usando a notação XPath.

Valor padrão

```
/content/folder[@name='Affinium Campaign']/folder[@name='eMessage
Reports']
```

campaignAnalysisTabInteractOnDemandFolder

Descrição

Sequência da pasta do servidor de relatório para os relatórios do Interact.

Valor padrão

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

Disponibilidade

Esta propriedade somente será aplicável se você instalar o Interact.

interactiveChannelAnalysisTabOnDemandFolder

Descrição

Sequência da pasta do servidor de relatório para os relatórios da guia Análise do canal interativo.

Valor padrão

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

Disponibilidade

Esta propriedade somente será aplicável se você instalar o Interact.

Campanha | partições | partição[n] | Interact | contactAndResponseHistTracking

Estas propriedades de configuração definem configurações para o módulo de histórico de contatos e respostas do Interact.

isEnabled

Descrição

Se for configurado como `sim`, ativa o módulo de histórico de contatos e respostas do Interact que copia o histórico de contatos e respostas do Interact das tabelas de migração no tempo de execução do Interact para as tabelas de histórico de contatos e respostas do Campaign. A propriedade `interactInstalled` também deve ser configurada como `sim`.

Valor padrão

no

Valores válidos

sim | não

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

runOnceADay

Descrição

Especifica se o ETL do histórico de contatos e respostas deve ser executado uma vez por dia. Se esta propriedade for configurada como `Sim`, o ETL

será executado durante o intervalo planejado especificado por `preferredStartTime` e `preferredEndTime`.

Se o ETL levar mais de 24 horas para executar e, portanto, perder o horário de início para o próximo dia, ele ignorará esse dia e será executado no horário planejado no dia seguinte. Por exemplo, se o ETL for configurado para executar entre 1h e 3h e o processo iniciar à 1h na segunda-feira e concluir às 2h na terça-feira, a próxima execução, originalmente planejada para 1h na terça-feira, será ignorada e o próximo ETL iniciará à 1h na quarta-feira.

O planejamento de ETL não considera mudanças de horário de verão. Por exemplo, se o ETL estiver planejado para executar entre 1h e 3h, ele poderá ser executado às 0h ou às 2h quando ocorrer a mudança de horário de verão.

Valor padrão

Não

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

processSleepIntervalInMinutes

Descrição

O número de minutos que o módulo de histórico de contatos e respostas do Interact aguarda entre a cópia de dados das tabelas de migração do tempo de execução do Interact para tabelas de histórico de contatos e respostas do Campaign.

Valor padrão

60

Valores válidos

Qualquer número inteiro maior que zero.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

preferredStartTime

Descrição

O horário preferencial para iniciar o processo ETL diário. Essa propriedade, quando usada em conjunto com a propriedade preferredEndTime, configura o intervalo de tempo preferencial durante o qual o ETL deve ser executado. O ETL iniciará durante o intervalo de tempo especificado e processará no máximo o número de registros especificados usando `maxJDBCFetchBatchSize`. O formato é HH:mm:ss AM ou PM, usando um relógio de 12 horas.

Valor padrão

12:00:00 AM

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

preferredEndTime

Descrição

O horário preferencial para concluir o processo ETL diário. Essa propriedade, quando usada em conjunto com a propriedade preferredStartTime, configura o intervalo de tempo preferencial durante o qual o ETL deve executar. O ETL iniciará durante o intervalo de tempo especificado e processará no máximo o número de registros especificados usando `maxJDBCFetchBatchSize`. O formato é HH:mm:ss AM ou PM, usando um relógio de 12 horas.

Valor padrão

2:00:00 AM

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

purgeOrphanResponseThresholdInMinutes

Descrição

O número de minutos que o módulo de histórico de contatos e respostas do Interact aguarda antes de limpar as respostas sem contato correspondente. Isso evita a criação de log de respostas sem a criação de log de contatos.

Valor padrão

180

Valores válidos

Qualquer número inteiro maior que zero.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

maxJDBCInsertBatchSize

Descrição

O número máximo de registros de um lote de JDBC antes da confirmação da consulta. Esse não é o número máximo de registros que o módulo de histórico de contatos e respostas do Interact processa em uma iteração. Durante cada iteração, o módulo de histórico de contatos e respostas do Interact processa todos os registros disponíveis nas tabelas de migração. No entanto, todos esses registros são divididos em chunks do `maxJDBCInsertSize`.

Valor padrão

1000

Valores válidos

Qualquer número inteiro maior que zero.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

maxJDBCFetchBatchSize

Descrição

O número máximo de registros de um lote de JDBC a serem buscados no banco de dados temporário. Poderá ser necessário aumentar esse valor para ajustar o desempenho do módulo de histórico de contatos e respostas.

Por exemplo, para processar 2,5 milhões de registros de histórico de contato por dia, você deverá configurar `maxJDBCFetchBatchSize` com um número maior que 2,5 milhões para que todos os registros de um dia sejam processados.

Em seguida, será possível configurar `maxJDBCFetchChunkSize` e `maxJDBCInsertBatchSize` com valores menores (neste exemplo, talvez 50.000 e 10.000, respectivamente). Alguns registros do dia seguinte também poderão ser processados, mas ficarão retidos até o dia seguinte.

Valor padrão

1000

Valores válidos

Qualquer número inteiro maior que zero

maxJDBCFetchChunkSize

Descrição

O número máximo de tamanho do chunk JDBC de leitura de dados durante o ETL (extrair, transformar e carregar). Em alguns casos, um tamanho do chunk maior que o tamanho de inserção pode melhorar a velocidade do processo ETL.

Valor padrão

1000

Valores válidos

Qualquer número inteiro maior que zero

deleteProcessedRecords

Descrição

Especifica se os registros do histórico de contatos e respostas deverão ser retidos após serem processados.

Valor padrão

Sim

completionNotificationScript

Descrição

Especifica o caminho absoluto para um a script a ser executado quando o ETL for concluído. Se você especificar um script, cinco argumentos serão passados para o script de notificação de conclusão: horário de início, horário de encerramento, número total de registros CH processados, número total de registros RH processados e status. O horário de início e o horário de término são valores numéricos que representam o número de milissegundos decorridos desde 1970. O argumento de status indica se a

tarefa ETL foi um sucesso ou uma falha. 0 indica uma tarefa ETL bem-sucedida. 1 indica uma falha e que há alguns erros na tarefa ETL.

Valor padrão

None

fetchSize

Descrição

Permite configurar o fetchSize JDBC ao recuperar registros das tabelas de migração.

Especialmente em bancos de dados Oracle, ajuste a configuração para o número de registros que o JDBC deve recuperar com cada roundtrip de rede. Para lotes grandes de 100 mil ou mais, tente 10000. Tome cuidado para não usar um valor muito grande aqui, porque isso causará impacto no uso da memória e os ganhos serão insignificantes ou prejudiciais.

Valor padrão

None

daysBackInHistoryToLookupContact

Descrição

Limita os registros que são procurados durante as consultas do histórico de respostas àqueles dentro do número de dias especificado anteriormente. Para bancos de dados com um número grande de registros de histórico de respostas, isso pode reduzir o tempo de processamento nas consultas limitando o período de busca ao número de dias especificado.

O valor padrão 0 indica que todos os registros são procurados.

Valor padrão

0 (zero)

Campanha | partições | partição[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

Estas propriedades de configuração definem a origem de dados do módulo de histórico de contatos e respostas do Interact.

jndiName

Descrição

Use a propriedade `systemTablesDataSource` para identificar a origem de dados Java Naming and Directory Interface (JNDI) definida no servidor de aplicativos (Websphere ou WebLogic) para as tabelas de tempo de execução do Interact.

O banco de dados de tempo de execução do Interact é o banco de dados preenchido com os scripts DLL `aci_runtime` e `aci_populate_runtime` e, por exemplo, contém as tabelas a seguir (entre outras): `UACI_CHOfferAttrib` e `UACI_DefaultedStat`.

Valor padrão

Nenhum valor padrão definido.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

databaseType

Descrição

Tipo de banco de dados da origem de dados de tempo de execução do Interact.

Valor padrão

SQLServer

Valores válidos

SQLServer | Oracle | DB2

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

schemaName

Descrição

O nome do esquema contendo as tabelas de migração do módulo de histórico de contatos e respostas. Esse nome deve ser o mesmo que o nome das tabelas do ambiente de tempo de execução.

Não é necessário definir um esquema.

Valor padrão

Nenhum valor padrão definido.

Campanha | partições | partição[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

Estas propriedades de configuração definem o tipo de contato da campanha que é mapeado para um "contato" para propósitos de relatório ou aprendizado.

contacted

Descrição

O valor designado à coluna `ContactStatusID` da tabela `UA_DtlContactHist` nas tabelas de sistema do Campaign para um contato da oferta. O valor deve ser uma entrada válida na tabela `UA_ContactStatus`. Consulte o *Campaign Administrator's Guide* para obter detalhes sobre como incluir tipos de contatos.

Valor padrão

2

Valores válidos

Um número inteiro maior que zero

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

Estas propriedades de configuração definem as respostas de aceitação ou rejeição de relatórios e aprendizado.

accept

Descrição

O valor designado à coluna ResponseTypeID da tabela UA_ResponseHistory nas tabelas de sistema do Campaign para uma oferta aceita. O valor deve ser uma entrada válida na tabela UA_UsrResponseType. Você deve designar à coluna CountsAsResponse o valor 1, uma resposta.

Consulte o *Campaign Administrator's Guide* para obter detalhes de como incluir os tipos de resposta.

Valor padrão

3

Valores válidos

Um número inteiro maior que zero

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

reject

Descrição

O valor designado à coluna ResponseTypeID da tabela UA_ResponseHistory nas tabelas de sistema do Campaign para uma oferta rejeitada. O valor deve ser uma entrada válida na tabela UA_UsrResponseType. Você deve designar o valor 2, rejeição, à coluna CountsAsResponse. Consulte o *Campaign Administrator's Guide* para obter detalhes de como incluir os tipos de resposta.

Valor padrão

8

Valores válidos

Qualquer número inteiro maior que zero.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | relatório

Estas propriedades de configuração definem os nomes de relatório na integração com o Cognos.

interactiveCellPerformanceByOfferReportName

Descrição

Nome do relatório Desempenho da célula interativa por oferta. Este nome deve corresponder ao nome deste relatório no servidor Cognos.

Valor padrão

Desempenho de Célula Interativo por Oferta

treatmentRuleInventoryReportName

Descrição

Nome do relatório Inventário de regra de tratamento. Este nome deve corresponder ao nome deste relatório no servidor Cognos.

Valor padrão

Inventário de regra de tratamento do canal

deploymentHistoryReportName

Descrição

Nome do Relatório de histórico de implementação. Este nome deve corresponder ao nome deste relatório no servidor Cognos

Valor padrão

Histórico de Implementação de Canal

Campaign | partições | partição[n] | Interact | aprendizado

Estas propriedades de configuração permitem ajustar o módulo de aprendizado integrado.

confidenceLevel

Descrição

Uma porcentagem indicando quão confiante você deseja que o utilitário de aprendizado esteja antes de alternar de exploração para exploração. Um valor igual a 0 encerra efetivamente a exploração.

Essa propriedade será aplicável se a propriedade `Interact > offerserving > optimizationType` do tempo de execução do Interact estiver configurada somente como `BuiltInLearning`.

Valor padrão

95

Valores válidos

Um número inteiro entre 0 e 95 divisível por 5 ou 99.

validateonDeployment

Descrição

Se configurado como `No`, Interact não valida o módulo de aprendizado ao implementar. Se configurado como `yes`, Interact valida o módulo de aprendizado ao implementar.

Valor padrão

`Não`

Valores válidos

Sim | Não

maxAttributeNames

Descrição

O número máximo de atributos de aprendizado que o utilitário de aprendizado do Interact monitora.

Essa propriedade será aplicável se a propriedade `Interact > offerserving > optimizationType` do tempo de execução do Interact estiver configurada somente como `BuiltInLearning`.

Valor padrão

`10`

Valores válidos

Qualquer número inteiro.

maxAttributeValues

Descrição

O número máximo de valores que o módulo de aprendizado do Interact controla para cada atributo de aprendizado.

Essa propriedade será aplicável se a propriedade `Interact > offerserving > optimizationType` do tempo de execução do Interact estiver configurada somente como `BuiltInLearning`.

Valor padrão

5

otherAttributeValue

Descrição

O nome padrão do valor de atributo usado para representar todos os valores de atributo além de `maxAttributeValues`.

Essa propriedade será aplicável se a propriedade `Interact > offerserving > optimizationType` do tempo de execução do Interact estiver configurada somente como `BuiltInLearning`.

Valor padrão

`Outro`

Valores válidos

Uma sequência ou um número.

Exemplo

Se maxAttributeValues for configurado como 3 e otherAttributeValue como outro, o módulo de aprendizado controlará os primeiros três valores. Todos os outros valores serão designados à outra categoria. Por exemplo, se estiver controlando o atributo de cor do cabelo do visitante e os primeiros cinco visitantes tiverem cores de cabelo preto, castanho, loiro, ruivo e grisalho, o utilitário de aprendizado controlará as cores de cabelo preto, castanho e loiro. As cores ruivo e grisalho serão agrupadas sob o `otherAttributeValue` outro.

percentRandomSelection

Descrição

O percentual do tempo em que o módulo de aprendizado apresenta uma oferta aleatória. Por exemplo, configurar percentRandomSelection como 5 significa que em 5% do tempo (5 em cada 100 recomendações) o módulo de aprendizado apresentará uma oferta aleatória independente da pontuação. Ativar o `percentRandomSelection` substitui a propriedade de configuração `offerTieBreakMethod`. Quando `percentRandomSelection` está ativado, esta propriedade é configurada, independente do aprendizado estar ativado ou desativado ou se o aprendizado integrado ou externo ser usado.

Valor padrão

5

Valores válidos

Qualquer número inteiro a partir de 0 (que desativa o recurso `percentRandomSelection`) até 100.

recencyWeightingFactor

Descrição

A representação decimal de uma porcentagem do conjunto de dados definida por `recencyWeightingPeriod`. Por exemplo, o valor padrão 0,15 significa que 15% dos dados usados pelo utilitário de aprendizado são provenientes de `recencyWeightingPeriod`.

Essa propriedade será aplicável se a propriedade `Interact > offerserving > optimizationType` do tempo de execução do Interact estiver configurada somente como `BuiltInLearning`.

Valor padrão

0,15

Valores válidos

Um valor decimal menor que 1.

recencyWeightingPeriod

Descrição

O tamanho em horas de dados concedidos à porcentagem de peso `recencyWeightingFactor` pelo módulo de aprendizado. Por exemplo, o valor padrão 120 significa que `recencyWeightingFactor` dos dados usados pelo módulo de aprendizado são provenientes das últimas 120 horas.

Essa propriedade somente será aplicável se `optimizationType` estiver configurado como `builtInLearning`.

Valor padrão

120

minPresentCountThreshold

Descrição

O número mínimo de vezes que uma oferta deve ser apresentada antes que seus dados sejam usados nos cálculos e o módulo de aprendizado entre no modo de exploração.

Valor padrão

0

Valores válidos

Um número inteiro maior que ou igual a zero.

enablePruning

Descrição

Se for configurado como `Sim`, o módulo de aprendizado do Interact determinará por algoritmos quando um atributo de aprendizado (padrão ou dinâmico) não é previsível. Se o atributo de aprendizado não for previsível, o módulo de aprendizado não considerará esse atributo na determinação da ponderação de uma oferta. Isso continua até que o módulo de aprendizado agregue dados de aprendizado.

Se for configurado como `Não`, o módulo de aprendizado sempre usará todos os atributos de aprendizado padrão. Por não remover atributos não previsíveis, o módulo de aprendizado poderá não ser tão preciso quanto poderia.

Valor padrão

Sim

Valores válidos

Sim | Não

Campanha | partições | partição[n] | Interact | aprendizado | learningAttributes | [learningAttribute]

Estas propriedades de configuração definem os atributos de aprendizado padrão.

attributeName

Descrição

Cada `attributeName` é o nome de um atributo de visitante que você deseja que o módulo de aprendizado monitore. Ele deve corresponder ao nome de um par nome-valor nos dados da sessão.

Essa propriedade será aplicável se a propriedade `Interact > offerserving > optimizationType` do tempo de execução do Interact estiver configurada somente como `BuiltInLearning`.

Valor padrão

Nenhum valor padrão definido.

Campanha | partições | partição[n] | Interact | implementação

Estas propriedades de configuração definem as configurações de implementação.

chunkSize

Descrição

O tamanho máximo de fragmentação em KB para cada pacote de implementação do Interact.

Valor padrão

500

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | serverGroups | [serverGroup]

Estas propriedades de configuração definem as configurações do grupo de servidores.

serverGroupName

Descrição

O nome do grupo de servidor de runtime do Interact. Esse é o nome que aparece na guia de resumo do canal interativo.

Valor padrão

Nenhum valor padrão definido.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

Estas propriedades de configuração definem os servidores de runtime do Interact.

instanceURL

Descrição

A URL do servidor de runtime do Interact. Um grupo de servidores pode conter diversos servidores de runtime do Interact, no entanto, cada servidor deve ser criado sob uma nova categoria.

Valor padrão

Nenhum valor padrão definido.

Exemplo

`http://server:port/interact`

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | fluxograma

Estas propriedades de configuração definem o ambiente de tempo de execução do Interact usado para execuções de teste de fluxogramas interativos.

serverGroup

Descrição

O nome do grupo de servidores do Interact que o Campaign usa para realizar uma execução de teste. Esse nome deve corresponder ao nome da categoria criado em `serverGroups`.

Valor padrão

Nenhum valor padrão definido.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

dataSource

Descrição

Use a propriedade `dataSource` para identificar a origem de dados física para o Campaign usar ao realizar execuções de teste de fluxogramas interativos. Essa propriedade deve corresponder à origem de dados definida pela propriedade `Campanha > partições > partiçãoN > dataSources` para a origem de dados de execução de teste definida para o tempo de design do Interact.

Valor padrão

Nenhum valor padrão definido.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

eventPatternPrefix

Descrição

A propriedade `eventPatternPrefix` é um um valor de sequência prefixado aos nomes de padrão de evento para permitir que eles sejam usados em expressões em processos de Seleção ou Decisão em fluxogramas interativos.

Observe que se você alterar esse valor, você deverá implementar as mudanças globais no canal interativo para que essa configuração atualizada entre em vigor.

Valor padrão

`EventPattern`

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

Estas propriedades de configuração definem o código da célula padrão para a tabela de ofertas padrão. Somente será necessário configurar essas propriedades se você estiver definindo designações de ofertas globais.

DefaultCellCode

Descrição

O código de célula padrão que o Interact usa se você não definir um código de célula na tabela de ofertas padrão.

Valor padrão

Nenhum valor padrão definido.

Valores válidos

Uma sequência que corresponde ao formato de código de célula definido no Campaign

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

Estas propriedades de configuração definem o código de célula padrão para a tabela offersBySQL. Somente será necessário configurar essas propriedades se forem usadas consultas SQL para obter um conjunto desejado de ofertas candidatas.

DefaultCellCode

Descrição

O código de célula padrão que o Interact usa para qualquer oferta na(s) tabela(s) OffersBySQL que possuem um valor nulo na coluna de código de célula (ou se a coluna de código de célula estiver completamente ausente). Esse valor deverá ser um código de célula válido.

Valor padrão

Nenhum valor padrão definido.

Valores válidos

Uma sequência que corresponde ao formato de código de célula definido no Campaign

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

Essas propriedades de configuração definem o código da célula padrão da tabela de substituição de pontuação. Somente será necessário configurar essas propriedades se você estiver definindo designações de ofertas individuais.

DefaultCellCode

Descrição

O código de célula padrão que o Interact usará se você não definir um código de célula na tabela de substituição de pontuação.

Valor padrão

Nenhum valor padrão definido.

Valores válidos

Uma sequência que corresponde ao formato de código de célula definido no Campaign

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campanha | partições | partição[n] | servidor | interno

As propriedades nesta categoria especificam as configurações de integração e os limites de internalID para a partição do Campaign selecionada. Se a instalação do Campaign tiver diversas partições, configure essas propriedades para cada partição que deseja afetar.

internalIdLowerLimit

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

As propriedades internalIdUpperLimit e internalIdLowerLimit restringem os IDs internos do Campaign para estarem dentro do intervalo especificado. Observe que os valores são inclusivos: isto é, o Campaign pode usar ambos os limites, inferior e superior.

Valor padrão

0 (zero)

internalIdUpperLimit

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

As propriedades internalIdUpperLimit e internalIdLowerLimit restringem os IDs internos do Campaign para estarem dentro do intervalo especificado. Os valores são inclusivos: isto é, o Campaign pode usar o limite inferior e superior. Se o Distributed Marketing estiver instalado, configure o valor como 2147483647.

Valor padrão

4294967295

eMessageInstalled

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

Indica que o eMessage está instalado. Ao selecionar Sim, os recursos do eMessage ficarão disponíveis na interface do Campaign.

O instalador do IBM configura esta propriedade como Sim para a partição padrão na instalação do eMessage. Para as partições adicionais em que o eMessage foi instalado, você deverá configurar esta propriedade manualmente.

Valor padrão

Não

Valores válidos

Sim | Não

interactInstalled

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

Após instalar o ambiente de design do Interact, esta propriedade de configuração deverá ser configurada como Sim para ativar o ambiente de design do Interact no Campaign.

Se o Interact não estiver instalado, configure como Não. Configurar esta propriedade como Não não remove os menus e as opções do Interact da interface com o usuário. Para remover os menus e as opções, você deve cancelar manualmente o registro do Interact, usando o utilitário configTool.

Valor padrão

No

Valores válidos

Sim | Não

Disponibilidade

Esta propriedade somente se aplicará se o Interact estiver instalado.

MO_UC_integration

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

Ativa a integração com o Marketing Operations para esta partição se a integração for ativada nas definições de configuração da **Plataforma**. Para obter mais informações, consulte o *IBM Marketing Operations and Campaign Integration Guide*.

Valor padrão

No

Valores válidos

Sim | Não

MO_UC_BottomUpTargetCells

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

Para esta partição, permite células ascendentes para Planilhas de célula de destino, se **MO_UC_integration** estiver ativado. Quando for configurado como Sim, as células ascendentes e descendentes de destino ficarão visíveis, mas as células de destino ascendentes serão somente leitura. Para obter mais informações, consulte o *IBM Marketing Operations and Campaign Integration Guide*.

Valor padrão

No

Valores válidos

Sim | Não

Legacy_campaigns

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

Para esta partição, ativa o acesso às campanhas criadas antes de o Marketing Operations e o Campaign serem integrados. Aplica-se somente se **MO_UC_integration** for configurado como Sim. As campanhas de legado também incluem campanhas criadas no Campaign 7.x e vinculadas a projetos Plan 7.x. Para obter mais informações, consulte o *IBM Marketing Operations and Campaign Integration Guide*.

Valor padrão

No

Valores válidos

Sim | Não

IBM Marketing Operations - Integração de oferta

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

Ativará a capacidade de usar o Marketing Operations para executar tarefas de gerenciamento de ciclo de vida de oferta nesta partição, se **MO_UC_integration** for ativado para esta partição. A integração da oferta deve ser ativada nas definições de configuração da **Plataforma**. Para obter mais informações, consulte o *IBM Marketing Operations and Campaign Integration Guide*.

Valor padrão

No

Valores válidos

Sim | Não

UC_CM_integration

Categoria da configuração

Campaign|partitions|partition[n]|server|internal

Descrição

Ativa a integração do segmento on-line do Digital Analytics para uma partição do Campaign. Se este valor é configurado para Sim, a caixa de processo de Seleção em um fluxograma fornece a opção de selecionar **Segmentos do Digital Analytics** como entrada. Para configurar a integração do Digital Analytics para cada partição, escolha **Configurações > Configuração > Campaign | partições | partição[n] | Coremetrics**.

Valor padrão

No

Valores válidos

Sim | Não

numRowsReadToParseDelimitedFile

Categoria da configuração

`Campaign|partitions|partition[n]|server|internal`

Descrição

Esta propriedade é usada ao mapear um arquivo delimitado como uma tabela de usuário. Ela também é usada pela caixa Processo de pontuação ao importar um arquivo de saída de pontuação a partir do IBM SPSS Modeler Advantage Marketing Edition. Para importar ou mapear um arquivo delimitado, o Campaign precisa analisar o arquivo para identificar as colunas, os tipos de dados (tipos de campos) e as larguras das colunas (comprimentos dos campos).

O valor padrão de 100 significa que o Campaign examina as 50 primeiras e as 50 últimas entradas de linha no arquivo delimitado. O Campaign aloca, então, o comprimento do campo com base no maior valor que ele localiza nessas entradas. Na maioria dos casos, o valor padrão é suficiente para determinar os comprimentos dos campos. Entretanto, em arquivos delimitados muito grandes, um filtro posterior poderá exceder o comprimento estimado computado pelo Campaign, o que poderá causar um erro durante o tempo de execução do fluxograma. Portanto, se você estiver mapeando um arquivo muito grande, é possível aumentar este valor para que o Campaign examine mais entradas de linha. Por exemplo, um valor de 200 faz o Campaign examinar as 100 primeiras entradas de linha e as 100 últimas entradas de linha do arquivo.

Um valor de 0 examina o arquivo inteiro. Geralmente, isso é necessário somente se você estiver importando ou mapeando arquivos que têm larguras de dados variáveis de campos que não podem ser identificados lendo as primeiras e últimas linhas. A leitura de arquivo inteiro para arquivos extremamente grandes pode aumentar o tempo de processamento necessário para o mapeamento de tabela e execuções da caixa do processo Pontuar.

Valor padrão

100

Valores válidos

0 (todas as linhas) ou qualquer número inteiro positivo

Campanha | monitoramento

As propriedades nesta categoria especificam se o recurso Monitoramento operacional está ativado, a URL do servidor de Monitoramento operacional e comportamento do armazenamento em cache. O Monitoramento operacional é exibido e permite controlar fluxogramas ativos.

cacheCleanupInterval

Descrição

A propriedade `cacheCleanupInterval` especifica o intervalo, em segundos, entre limpezas automáticas do cache de status do fluxograma.

Essa propriedade não está disponível em versões do Campaign anteriores à 7.0.

Valor padrão

600 (10 minutos)

cacheRunCompleteTime

Descrição

A propriedade cacheRunCompleteTime especifica o período de tempo, em minutos, durante o qual as execuções concluídas são armazenadas em cache e exibidas na página Monitoramento.

Essa propriedade não está disponível em versões do Campaign anteriores à 7.0.

Valor padrão

4320

monitorEnabled

Descrição

A propriedade monitorEnabled especifica se o monitor está ativado.

Essa propriedade não está disponível em versões do Campaign anteriores à 7.0.

Valor padrão

FALSE

Valores válidos

TRUE | FALSE

serverURL

Descrição

A propriedade Campanha > monitoramento > serverURL especifica a URL do servidor de Monitoramento operacional. Essa é uma configuração obrigatória. Modifique o valor se a URL do servidor de Monitoramento operacional não for a padrão.

Se o Campaign estiver configurado para usar comunicações Secure Sockets Layer (SSL), configure o valor dessa propriedade para usar HTTPS. Por exemplo: serverURL=https://*host:SSL_port*/Campaign/OperationMonitor em que:

- *host* é o nome ou endereço IP da máquina na qual o aplicativo da web está instalado
- *SSL_Port* é a porta SSL do aplicativo da web.

Observe o https na URL.

Valor padrão

http://localhost:7001/Campaign/OperationMonitor

monitorEnabledForInteract

Descrição

Se for configurado como TRUE, ativará o servidor do conector JMX do Campaign para o Interact. O Campaign não tem segurança JMX.

Se for configurado como FALSE, não será possível se conectar ao servidor do conector JMX do Campaign.

Esse monitoramento JMX é somente para o módulo de histórico de contatos e respostas do Interact.

Valor padrão

FALSE

Valores válidos

TRUE | FALSE

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

protocol

Descrição

O protocolo de recebimento para o servidor do conector JMX do Campaign, se o monitorEnabledForInteract estiver configurado como sim.

Esse monitoramento JMX é somente para o módulo de histórico de contatos e respostas do Interact.

Valor padrão

JMXMP

Valores válidos

JMXMP | RMI

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

port

Descrição

A porta de recebimento para o servidor do conector JMX do Campaign, se o monitorEnabledForInteract estiver configurado como sim.

Esse monitoramento JMX é somente para o módulo de histórico de contatos e respostas do Interact.

Valor padrão

2004

Valores válidos

Um número inteiro entre 1025 e 65535.

Disponibilidade

Essa propriedade será aplicável apenas se tiver instalado o Interact.

Campaign | partições | partição[n] | Interact | outboundChannels

Estas propriedades de configuração permitem ajustar os canais de saída para mensagens acionadas.

category name

Descrição

Esta propriedade define o nome deste canal de saída. O nome deve ser exclusivo entre todos os canais de saída.

name

Descrição

O nome do seu canal de saída.

Nota: Você deve reiniciar seu servidor de aplicativos para que a mudanças entrem em vigor.

Campaign | partições | partição[n] | Interact | outboundChannels | Dados do Parâmetro

Estas propriedades de configuração permitem ajustar os canais de saída para mensagens acionadas.

category name

Descrição

Esta propriedade define o nome deste parâmetro. O nome deve ser exclusivo entre todos os parâmetros para esse canal de saída.

value

Descrição

Esta propriedade define os parâmetros, no formato de pares de nome e valor, necessários para este gateway de saída.

Apêndice D. Personalização de oferta em tempo real no lado do cliente

Pode haver situações em que deseje fornecer personalização de oferta em tempo real sem implementar o código Java de nível baixo ou chamadas SOAP para o servidor do Interact. Por exemplo, quando um visitante inicialmente carrega uma página da web em que o conteúdo do Javascript é sua única programação estendida disponível, ou quando um visitante abre um email em que apenas o conteúdo HTML é possível. O IBM Interact fornece vários conectores que fornecem gerenciamento de oferta em tempo real em situações em que você tem controle apenas sobre o conteúdo da web que é carregado no lado do cliente ou em que você deseja simplificar sua interface com o Interact.

Sua instalação do Interact inclui dois conectores para personalização de oferta iniciada no lado do cliente:

- "Sobre o Conector da Mensagem do Interact". Usando o Conector da Mensagem, o conteúdo da web em emails (por exemplo) ou outra mídia eletrônica pode conter imagem e identificações de link para fazer chamad ao servidor Interact para apresentação de oferta de carregamento de página e páginas de entrada de click-through.
- "Sobre o Interact Web Connector" na página 304. Usando o Conector da Web (também chamado de Conector JS), as páginas da web podem usar oJavaScript do lado do cliente para gerenciar arbitragem de oferta, apresentação e histórico de contato/resposta por meio da apresentação de oferta de carregamento de página e páginas de entrada de click-through.

Sobre o Conector da Mensagem do Interact

O Conector da Mensagem do Interact permite que emails e outras mídias eletrônicas façam chamadas ao IBM Interact para permitir que ofertas personalizadas sejam apresentadas no tempo de abertura e quando o cliente executa click-through na mensagem para o site especificado. Isto é realizado por meio do uso de duas identificações principais: a identificação de imagem (`IMG`), que carrega as ofertas personalizadas no tempo de abertura e a identificação de link (`A`), que captura informações sobre o click-through e redireciona o cliente para uma página de entrada específica.

Exemplo

O exemplo a seguir mostra algum código HTML que pode ser incluído em um ponto de marketing (por exemplo, em uma mensagem de email) que contém uma URL de identificação `IMG` (que passa informações quando o documento abre no servidor Interact e recupera a imagem de oferta apropriada em resposta) e uma URL de identificação `A` (que determina que informações são passadas para o servidor Interact no click-through):

```
<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>
```

No exemplo a seguir, uma identificação IMG é colocada em uma identificação A, causando o comportamento a seguir:

1. Quando a mensagem de email é aberta, o Conector da Mensagem recebe uma solicitação contendo as informações codificadas na identificação IMG: msgID e linkID para esta mensagem e parâmetros do cliente que incluem ID do usuário, nível de renda e tipo de renda.
2. Estas informações são passadas por uma chamada API para o servidor de runtime do Interact.
3. O servidor de runtime retorna uma oferta para o Conector da Mensagem, que recupera a URL da imagem de oferta e fornece essa URL (com quaisquer parâmetros incluídos) e redireciona a solicitação de imagem para essa URL de oferta.
4. O cliente vê a oferta como uma imagem.

Nesse ponto, o cliente pode clicar nessa imagem para responder à oferta de alguma maneira. Esse click-through, usando a identificação A e seu atributo HREF especificado (que especifica a URL de destino) envia outra solicitação para o Conector da Mensagem para uma página de entrada vinculada a essa URL da oferta. O navegador do cliente é redirecionado à página de entrada conforme configurado na oferta.

Observe que uma identificação de click-through A não é estritamente necessária; a oferta pode consistir em uma imagem apenas, como um cupom para o cliente imprimir.

Instalando o Conector da Mensagem

Os arquivos que você precisa instalar, implementar e executar o Conector da Mensagem foram automaticamente incluídos com sua instalação do servidor de runtime do IBM Interact. Esta seção resume as etapas necessárias para que o Conector da Mensagem fique pronto para uso.

Instalar e implementar o Conector da Mensagem envolve as tarefas a seguir:

- Opcionalmente, definir as configurações padrão para o Conector da Mensagem conforme descrito em "Configurando o Conector da Mensagem".
- Criar as tabelas de banco de dados necessárias para armazenar os dados da transação do Conector da Mensagem conforme descrito em "Criando as tabelas do Conector da Mensagem" na página 299.
- Instalar o aplicativo da web do Conector da Mensagem conforme descrito em "Implementando e executando o Conector da Mensagem" na página 300.
- Criar as identificações IMG e A em seus pontos de marketing (email ou páginas da web, por exemplo) necessárias para chamar ofertas do Conector da Mensagem na abertura e no click-through, conforme descrito em "Criando links do Conector da Mensagem" na página 301.

Configurando o Conector da Mensagem

Antes de implementar o Conector da Mensagem, você deve customizar o arquivo de configuração incluído com sua instalação para corresponder a seu ambiente específico. É possível modificar o arquivo XML chamado MessageConnectorConfig.xml localizado em seu diretório do Conector da Mensagem no servidor de runtime do Interact, semelhante a <Interact_home>/msgconnector/config/MessageConnectorConfig.xml.

Sobre Esta Tarefa

O arquivo `MessageConnectorConfig.xml` contém algumas definições de configuração que são necessárias e algumas que são opcionais. As configurações que você usa devem ser customizadas para sua instalação específica. Siga as etapas aqui para modificar a configuração.

Procedimento

1. Se o Conector da Mensagem for implementado e estiver em execução em seu servidor de aplicativos da web, remova a implementação do Conector da Mensagem antes de continuar.
2. No servidor de runtime do Interact, abra o arquivo `MessageConnectorConfig.xml` em qualquer texto ou editor de XML.
3. Modifique as definições de configuração conforme necessário, assegurando que as configurações *necessárias* a seguir estejam corretas para sua instalação.
   - `<interactUrl>`, a URL do servidor de runtime do Interact à qual as identificações de página do Conector da Mensagem devem se conectar e na qual o Conector da Mensagem está em execução.
   - `<imageErrorLink>`, a URL à qual o Conector da Mensagem será redirecionado se ocorrer um erro ao processar uma solicitação para uma imagem de oferta.
   - `<landingPageErrorLink>`, a URL à qual o Conector da Mensagem será redirecionado se ocorrer um erro ao processar uma solicitação para uma página de entrada de oferta.
   - `<audienceLevels>`, uma seção do arquivo de configuração que contém uma ou mais conjuntos d configurações de nível de público e que especifica o nível de público padrão se não for especificado nenhuma pelo link do Conector da Mensagem. Deve haver pelo menos um nível de público configurado.

   Todas as definições de configuração são descritas em maior detalhe no "Definições de configuração do Conector da Mensagem".
4. Quando você tiver concluído as mudanças na configuração, salve e feche o arquivo `MessageConnectorConfig.xml`.
5. Continue configurando e implementando o Conector da Mensagem.

Definições de configuração do Conector da Mensagem:

Para configurar o Conector da Mensagem, é possível modificar o arquivo XML chamado `MessageConnectorConfig.xml` localizado em seu diretório do Conector da Mensagem no servidor de runtime do Interact, geralmente `<Interact_home>/msgconnector/config/MessageConnectorConfig.xml`. Cada uma das configurações nesse arquivo XML é descrita aqui. Esteja ciente de que se você modificar esse arquivo após o Conector da Mensagem ser implementado e estiver em execução, assegure-se de remover a implementação e reimplementar o Conector da Mensagem ou reiniciar o servidor de aplicativos para recarregar essas configurações após terminar de modificar o arquivo.

Configurações gerais

A tabela a seguir contém uma lista das configurações opcionais e necessárias contidas na seção `generalSettings` do arquivo `MessageConnectorConfig.xml`.

Tabela 24. Configurações gerais do Conector da Mensagem

Elemento	Descrição	Valor padrão
`<interactURL>`	A URL do servidor de runtime do Interact para manipular as chamadas das tags de página do Conector da Mensagem, como o servidor de runtime no qual o Conector da Mensagem está em execução. Este elemento é necessário.	`http://localhost:7001/interact`
`<defaultDateTimeFormat>`	O formato de data padrão.	`MM/dd/aaaa`
`<log4jConfigFileLocation>`	O local do arquivo de propriedade Log4j. É relativo à variável de ambiente *$MESSAGE_CONNECTOR_HOME* se estiver configurada; caso contrário, este valor é relativo ao caminho raiz do aplicativo da web Conector da Mensagem.	`config/ MessageConnectorLog4j.properties`

Valores de parâmetro padrão

A tabela a seguir contém uma lista das configurações opcionais e necessárias contidas na seção `defaultParameterValues` do arquivo `MessageConnectorConfig.xml`.

Tabela 25. Configurações de parâmetro padrão do Conector da Mensagem

Elemento	Descrição	Valor padrão
`<interactiveChannel>`	O nome do canal interativo padrão.	
`<interactionPoint>`	O nome do ponto de interação padrão.	
`<debugFlag>`	Determina se a depuração está ativada. Os valores permitidos são `true` e `false`.	`false`
`<contactEventName>`	O nome padrão do evento de contato que é postado.	
`<acceptEventName>`	O nome padrão do evento de aceitação que é postado.	
`<imageUrlAttribute>`	O nome do atributo de oferta padrão que contém a URL para a imagem de oferta, se nenhum for especificado no link do Conector da Mensagem.	
`<landingPageUrlAttribute>`	A URL padrão para a página de entrada de click-through se nenhuma for especificada no link do Conector da Mensagem.	

Configurações de comportamento

A tabela a seguir contém uma lista das configurações opcionais e necessárias contidas na seção `behaviorSettings` do arquivo `MessageConnectorConfig.xml`.

Tabela 26. Configurações de comportamento do Conector da Mensagem

Elemento	Descrição	Valor padrão
<imageErrorLink>	A URL para a qual o conector redireciona se ocorrer um erro ao processar uma solicitação para uma imagem de oferta. Esta configuração é necessária.	/images/default.jpg
<landingPageErrorLink>	A URL para a qual o conector redireciona se ocorrer um erro ao processar uma solicitação para uma página de entrada de click-through. Esta configuração é necessária.	/jsp/default.jsp
<alwaysUseExistingOffer>	Determina se a oferta armazenada em cache deve ser retornada, mesmo se já tiver expirado. Os valores permitidos são true e false.	false
<offerExpireAction>	A ação a ser tomada se a oferta original for localizada, mas já estiver expirada. Os valores permitidos são: • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

Configurações de armazenamento

A tabela a seguir contém uma lista de configurações opcionais e necessárias contidas na seção storageSettings do arquivo MessageConnectorConfig.xml.

Tabela 27. Configurações de armazenamento do MessageConnector

Elemento	Descrição	Valor padrão
<persistenceMode>	Quando o cache persiste novas entradas no banco de dados. Os valores permitidos são WRITE-BEHIND (em que os dados são gravados no cache inicialmente e atualizados para o banco de dados posteriormente) e WRITE-THROUGH (em que os dados são gravados no cache e no banco de dados ao mesmo tempo).	WRITE-THROUGH
<maxCacheSize>	O número máximo de entradas no cache de memória.	5000
<maxPersistenceBatchSize>	O tamanho do lote máximo ao persistir entradas no banco de dados.	200
<macCachePersistInterval>	O tempo máximo em segundos que uma entrada permanece no cache antes de ser persistida no banco de dados.	3
<maxElementOnDisk>	O número máximo de entradas no cache de disco.	5000
<cacheEntryTimeToExpireInSeconds>	A quantia máxima de tempo para as entradas no cache de disco persistirem antes de expirarem.	60000

Tabela 27. Configurações de armazenamento do MessageConnector (continuação)

Elemento	Descrição	Valor padrão
<jdbcSettings>	Uma seção do arquivo XML que contém informações específicas se uma conexão JDBC for usada. É mutuamente exclusiva com a seção <dataSourceSettings>.	Configurado por padrão para se conectar a um banco de dados SQLServer configurado no servidor local, mas se você ativar esta seção, deverá fornecer as configurações reais de JDBC e credenciais para efetuar login.
<dataSourceSettings>	Uma seção do arquivo XML que contém informações específicas se uma conexão de origem de dados for usada. É mutuamente exclusiva com a seção <jdbcSettings>.	Configurado por padrão para se conectar à origem de dados InteractDS definida no servidor de aplicativos da web local.

Níveis de público

A tabela a seguir contém uma lista das configurações opcionais e necessárias contidas na seção audienceLevels do arquivo MessageConnectorConfig.xml.

Observe que o elemento audienceLevels é opcionalmente usado para especificar o nível de público padrão para usar se nenhum estiver especificado no link do Conector da Mensagem, como no exemplo a seguir:

```
<audienceLevels default="Customer">
```

Neste exemplo, o valor para o atributo padrão corresponde ao nome de um audienceLevel definido nesta seção. Deve haver pelo menos um nível de público definido neste arquivo de configuração.

Tabela 28. Configurações do nível de público do MessageConnector

Elemento	Elemento	Descrição	Valor padrão
<audienceLevel>		O elemento que contém a configuração do nível de público. Forneça um atributo de nome, como em <audienceLevel name="Customer">	
	<messageLogTable>	O nome da tabela de logs. Este valor é necessário.	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<field>	A definição de um ou mais campos de ID de público para este audienceLevel.	
	<name>	O nome do campo de ID de público, conforme especificado no tempo de execução do Interact.	
	<httpParameterName>	O nome do parâmetro correspondente para este campo de ID de público.	
	<dbColumnName>	O nome da coluna correspondente no banco de dados para este campo de ID de público.	

Tabela 28. Configurações do nível de público do MessageConnector (continuação)

Elemento	Elemento	Descrição	Valor padrão
	<type>	O tipo de campo de ID de público, conforme especificado no tempo de execução do Interact. Os valores podem ser string ou numeric.	

Criando as tabelas do Conector da Mensagem

Antes de poder implementar o Conector da Mensagem do IBM Interact, você deve primeiro criar as tabelas no banco de dados em que os dados de tempo de execução do Interact estão armazenados. Você criará uma tabela para cada nível de público que você definiu. Para cada nível de público, o Interact usará as tabelas que você cria para registrar informações sobre transações do Conector da Mensagem.

Sobre Esta Tarefa

Use seu cliente de banco de dados para executar o script SQL do Conector da Mensagem com relação ao banco de dados ou esquema apropriado para criar as tabelas necessárias. Os scripts SQL para seu banco de dados suportado são instalados automaticamente ao instalar o servidor de runtime do Interact. Consulte as planilhas que concluiu no *IBM Interact Installation Guide* para obter detalhes sobre conectar ao banco de dados que contém tabelas de tempo de execução do Interact.

Procedimento

1. Ative seu cliente de banco de dados e conecte-se ao banco de dados no qual suas tabelas de tempo de execução do Interact estão atualmente armazenadas.
2. Execute o script apropriado no diretório <*Interact_home*>/msgconnector/scripts/ddl. A tabela a seguir lista os scripts SQL de amostra que podem ser usados para criar manualmente as tabelas do Conector da Mensagem:

Tabela 29. Scripts para criar as tabelas do Conector da Mensagem

Tipo de origem de dados	Nome do script
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

Observe que estes scripts são fornecidos como amostras. É possível usar uma convenção de nomenclatura ou estrutura diferente para valores de ID de público, portanto, pode ser necessário modificar o script antes de executá-lo. Em geral, é uma melhor prática ter uma tabela dedicada a cada nível de público.

As tabelas são criadas para conter as informações a seguir:

Tabela 30. Informações criadas pelos scripts SQL de amostra

Nome da coluna	Descrição
LogId	A chave primária desta entrada.
MessageId	O identificador exclusivo de cada instância do sistema de mensagens.
LinkId	O identificador exclusivo de cada link na mídia eletrônica (como um email).
OfferImageUrl	A URL para a imagem relacionada da oferta retornada.
OfferLandingPageUrl	A URL para a página de entrada relacionada da oferta retornada.
TreatmentCode	O código de tratamento da oferta retornada.
OfferExpirationDate	A data e hora de expiração da oferta retornada.
OfferContactDate	A data e hora em que a oferta foi retornada ao cliente.
AudienceId	O ID de público da mídia eletrônica.

Observe o seguinte sobre esta tabela:

- Dependendo do nível de público, haverá uma coluna AudienceId para cada componente da chave de público.
- A combinação de MessageId, LinkId e AudienceId(s) forma uma chave exclusiva desta tabela.

Quando o script tiver concluído a execução, você terá criado as tabelas necessárias para o Conector da Mensagem.

Resultados

Agora você está pronto para implementar o aplicativo da web do Conector da Mensagem.

Implementando e executando o Conector da Mensagem

O Conector da Mensagem do IBM Interact é implementado como um aplicativo da web independente em um servidor de aplicativos da web suportado.

Antes de Iniciar

Antes de poder implementar o Conector da Mensagem, assegure-se de que as tarefas a seguir tenham sido concluídas:

- Você deve ter instalado o servidor de runtime do IBM Interact. O aplicativo implementável Conector da Mensagem é instalado automaticamente com o servidor de runtime e está pronto para implementar a partir de seu diretório inicial do Interact.
- Você também deve ter executado os scripts SQL fornecidos com sua instalação para criar as tabelas necessárias no banco de dados de tempo de execução do Interact para uso pelo Conector da Mensagem conforme descrito no "Criando as tabelas do Conector da Mensagem" na página 299

Sobre Esta Tarefa

Assim como você implementa outros aplicativos IBM em um servidor de aplicativos da web antes de poder executá-los, você deve implementar o aplicativo Conector da Mensagem para torná-lo disponível para ofertas de serviço.

Procedimento

1. Conecte-se à sua interface de gerenciamento do servidor de aplicativos da web com os privilégios necessários para implementar um aplicativo.
2. Siga as instruções para seu servidor de aplicativos da web para implementar e executar o arquivo chamado `<Interact_home>/msgconnector/MessageConnector.war` Substitua *<Interact_home>* pelo diretório real no qual o servidor de runtime do Interact está instalado.

Resultados

O Conector da Mensagem agora está disponível para uso. Depois de ter configurado sua instalação do Interact para criar os dados subjacentes que o Conector da Mensagem usará para fornecer ofertas, como canais interativos e estratégias, fluxogramas, ofertas e assim por diante, será possível criar os links em sua mídia eletrônica que o Conector da Mensagem aceitará.

Criando links do Conector da Mensagem

Para usar o Conector da Mensagem para fornecer imagens de ofertas customizadas quando um usuário final interagir com sua mídia eletrônica (como ao abrir uma mensagem de email) e páginas de entrada customizadas quando o usuário final clicar na oferta, será necessário criar os links para integrar em sua mensagem. Esta seção fornece um resumo da marcação de HTML desses links.

Sobre Esta Tarefa

Independentemente do sistema que você usa para gerar suas mensagns não enviadas a usuários finais, é necessário gerar a marcação HTML para conter os campos apropriados (fornecidos nas marcas HTML como atributos) que contêm informações que deseja passar ao servidor de runtime do Interact. Siga as etapas abaixo para configurar o mínimo de informações necessárias para uma mensagem do Conector da Mensagem.

Observe que, embora as instruções aqui se refigram especificamente a mensagens que contêm links do Conector da Mensagem, é possível seguir as mesmas etapas e configuração para incluir links em páginas da web ou qualquer outra mídia eletrônica.

Procedimento

1. Crie o link `IMG` que aparecerá na sua mensagem com, no mínimo, os parâmetros a seguir:
   - `msgID`, indicando o identificador exclusivo para esta mensagem.
   - `linkID`, indicando o identificador exclusivo para o link na mensagem.
   - `audienceID`, o identificador do público ao qual o destinatário da mensagem pertence.

     Observe que se o ID de público for um ID composto, todos esses componentes deverão ser incluídos no link.

   Também é possível incluir parâmetros opcionais que incluem o nível de público, nome do canal interativo, nome do ponto de interação, URL do local da imagem e seus próprios parâmetros customizados não especificamente usados pelo Conector da Mensagem.
2. Opcionalmente, crie um link `A` que inclua o link `IMG` para que, quando o usuário clicar na imagem, o navegador carregue uma página que contém a oferta para o usuário. O link `A` também deve conter os três parâmetros listados acima (`msgID`, `linkID` e `audienceID`), mais quaisquer parâmetros opcionais (nível de público, nome do canal interativo e nome do ponto interativo) e parâmetros customizados não especificamente usados pelo Conector da Mensagem. Observe que o link `A` provavelmente conterá um link `IMG` do Conector da Mensagem, mas também é possível estar isolado na página conforme necessário. Se o link não contiver um link `IMG`, o link `IMG` deverá conter o mesmo conjunto de parâmetros que o link `A` incluído (incluindo quaisquer parâmetros opcionais ou customizados).
3. Quando os links estiverem corretamente definidos, gere e envie as mensagens de email.

Resultados

Para obter informações detalhadas sobre os parâmetros disponíveis e links de amostra, consulte “Parâmetros de solicitação de HTTP de identificação "IMG" e "A"” na página 302

Parâmetros de solicitação de HTTP de identificação "IMG" e "A"

Quando o Conector da Mensagem recebe uma solicitação, seja porque um usuário final abriu um email que contém uma identificação `IMG` codificada pelo Conector da Mensagem ou porque o usuário final usou clicked-through em uma identificação `A`, ele analisa os parâmetros incluídos com a solicitação para retornar os dados da oferta apropriados. Esta seção fornece uma lista dos parâmetros que podem ser incluídos na URL de solicitação (a identificação `IMG` carregada automaticamente quando uma imagem identificada é exibida quando o email é aberto) ou a identificação `A` (carregada quando a pessoa que está visualizando o email clica por meio da mensagem para o site especificado).

Parâmetros

Quando o Conector da Mensagem recebe uma solicitação, ele analisa os parâmetros incluídos com a solicitação. Os parâmetros incluem alguns ou todos os seguintes:

Nome de parâmetro	Descrição	Necessário?	Valor padrão
`msgId`	O identificador exclusivo da instância de email ou página da web.	Sim	None. Isto é fornecido pelo sistema criando a instância exclusiva do email ou página da web que contém a identificação.
`linkId`	O identificador exclusivo do link neste email ou página da web.	Sim	None. Isto é fornecido pelo sistema criando a instância exclusiva do email ou página da web que contém a identificação.
`audienceLevel`	O nível de público ao qual o destinatário desta comunicação pertence.	Não	O audienceLevel especificado como o padrão no elemento `audienceLevels` localizado no arquivo MessageConnectorConfig.xml.
`ic`	O nome do canal interativo (IC) de destino	Não	O valor do elemento `interactiveChannel` localizado na seção `defaultParameterValues` do arquivo MessageConnectorConfig.xml, que é "interactiveChannel" por padrão.
`ip`	O nome do ponto de interação (IP) que se aplica	Não	O valor do elemento `interactionPoint` localizado na seção `defaultParameterValues` do arquivo MessageConnectorConfig.xml, que é "headBanner" por padrão.
`offerImageUrl`	A URL da imagem de oferta de destino para a URL IMG na mensagem.	Não	None.
`offerImageUrlAttr`	O nome do atributo de oferta que possui a URL da imagem de oferta de destino	Não	O valor do elemento `imageUrlAttribute` localizado na seção `defaultParameterValues` do arquivo MessageConnectorConfig.xml.
`offerLandingPageUrl`	A URL da página de entrada correspondente à oferta de destino.	Não	None.
`offerLandingPageUrlAttr`	O nome do atributo de oferta que possui a URL da página de entrada correspondente à oferta de destino.	Não	O valor do elemento `landingPageUrlAttribute` localizado na seção `defaultParameterValues` do arquivo MessageConnectorConfig.xml.
`contactEvent`	O nome do evento de contato.	Não	O valor do elemento `contactEventName` localizado na seção `defaultParameterValues` do arquivo MessageConnectorConfig.xml, que é "contact" por padrão.

Nome de parâmetro	Descrição	Necessário?	Valor padrão
`responseEvent`	O nome do evento de aceitação.	Não	O valor do elemento `acceptEventName` localizado na seção `defaultParameterValues` do arquivo MessageConnectorConfig.xml, que é "accept" por padrão.
`debug`	O sinalizador de depuração. Configure este parâmetro para "true" somente para resolução de problemas e com instrução do suporte técnico do IBM .	Não	O valor do elemento `debugFlag` localizado na seção `defaultParameterValues` do arquivo MessageConnectorConfig.xml, que é "false" por padrão.
<audience id>	O ID de público deste usuário. O nome deste parâmetro é definido no arquivo de configuração.	Sim	None.

Quando o Conector da Mensagem recebe um parâmetro que não é reconhecido (isto é, não aparece na lista acima), ele é manipulado de uma das duas formas possíveis:

- Se um parâmetro não reconhecido for fornecido (por exemplo, "attribute", como em `attribute="attrValue"`) e houver um parâmetro correspondente do mesmo nome mais a palavra "Type" (por exemplo, "attributeType", como em `attributeType="string"`), isto fará com que o Conector da Mensagem crie um parâmetro Interact correspondente e passe-o para o tempo de execução do Interact.

  Os valores para o parâmetro Type podem ser quaisquer um dos seguintes:

  – string
  – numeric
  – datetime

    Para um parâmetro do tipo "datetime," o Conector da Mensagem também procura um parâmetro do mesmo nome mais a palavra "Pattern" (por exemplo, "attributePattern") cujo valor é um formato de data/hora válido. Por exemplo, você pode fornecer o parâmetro `attributePattern="MM/dd/yyyy"`.

    Observe que se você especificar um tipo de parâmetro de "datetime", mas não fornecer um padrão de data correspondente, o valor especificado no arquivo de configuração do Conector da Mensagem (localizado em `<installation_directory>/msgconnector/config/MessageConnectorConfig.xml`) no servidor Interact será usado.
- Se um parâmetro não reconhecido for fornecido e não houver valor de Tipo correspondente, o Conector da Mensagem passará esse parâmetro para a URL de redirecionamento de destino.

Para todos os parâmetros não reconhecidos, o Conector da Mensagem os passará para o servidor de runtime do Interact sem processar ou salvá-los.

Código do conector de mensagem de exemplo

A identificação `A` a seguir contém um exemplo de um conjunto de links do Conector da Mensagem que podem aparecer em um email:

```
<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>
```

Neste exemplo, a identificação `IMG` é carregado automaticamente quando o email é aberto. Ao recuperar a imagem da página especificada, a mensagem passa os parâmetros para o identificador de mensagem exclusivo (msgID), identificador de link exclusivo (linkID) e o identificador de usuário exclusivo (userid), junto com dois parâmetros adicionais (incomeCode and incomeType) a serem passados ao tempo de execução do Interact.

A identificação `A` fornece o atributo HREF (referência hipertexto) que transforma a imagem de oferta em um link que pode ser clicado no email. Se o visualizador da mensagem, ao ver a imagem, fizer click through na página de entrada, o identificador de mensagem exclusivo (`msgId`), o identificador de link (`linkId`) e o identificador de usuário (`userid`) serão transmitidos ao servidor, bem como um parâmetro adicional (`referral`) que é passado à URL de redirecionamento de destino.

Sobre o Interact Web Connector

O Interact WebConnector (também mencionado como JavaScript Connector, ou JSConnector) fornece um serviço no servidor de runtime do Interact que permite que o código JavaScript chame a API Interact Java. Isso permite que as páginas da web façam chamadas ao Interact para personalização de oferta em tempo real usando apenas código JavaScript integrado, sem ter de confiar em linguagens de desenvovimento da web (como Java, PHP, JSP e assim por diante). Por exemplo, você pode integrar um pequeno fragmento do código JavaScript em cada página de seu website que serve ofertas recomendadas pelo Interact, portanto, toda vez que a página for carregada, as chamadas serão feitas à API do Interact para assegurar que as melhores ofertas sejam exibidas na página de carregamento para o visitante do site.

Use o Conector da Web do Interact em situações em que deseja exibir ofertas a visitantes em uma página em que é possível que não tenha controle programático do lado do servidor sobre a exibição de página (como teria com, por exemplo, PHP ou outro script baseado em servidor), mas ainda é possível integrar o código JavaScript em seu conteúdo de página que será executado pelo navegador da web do visitante.

Dica: Os arquivos de Conector da Web do Interact são instalados automaticamente em seu servidor de runtime do Interact, no diretório do <*Interact_home*>/`jsconnector`. No diretório <*Interact_home*>/`jsconnector`, você encontrará um ReadMe.txt que contém notas e detalhes importantes sobre os recursos do Conector da Web, bem como arquivos de amostra e o código fonte do Conector da Web para usar como base para desenvolver suas próprias soluções. Se você não localizar informações para responder suas questões aqui, consulte o diretório `jsconnector` para obter mais informações.

Instalando o Web Connector no servidor de runtime

Uma instância do Web Connector é instalada automaticamente com seu servidor de runtime do IBM Interact e é ativada por padrão. Entretanto, há algumas configurações que você deve modificar antes de poder configurar e usar o Web Connector.

Sobre Esta Tarefa

As configurações que você deve modificar antes de poder usar o Web Connector que está instalado no servidor de runtime são incluídas na configuração do servidor de aplicativos da web. Por este motivo, será necessário reiniciar o servidor de aplicativos da web após concluir estas etapas.

Procedimento

1. Para o servidor de aplicativos da web no qual o servidor de runtime do Interact está instalado, configure as propriedades Java a seguir:

   `-DUI_JSCONNECTOR_ENABLE_INPROCESS=true`

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Substitua `<jsconnectorHome>` pelo caminho para o diretório `jsconnector` no servidor de runtime, que é `<Interact_Home>/jsconnector`.

   A forma na qual você configura as propriedades Java depende de seu servidor de aplicativos da web. Por exemplo, no WebLogic, você editaria o arquivo `startWebLogic.sh` ou `startWebLogic.cmd` para atualizar a configuração `JAVA_OPTIONS`, como neste exemplo:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
   jsconnector"
   ```

   No WebSphere Application Server, você configuraria esta propriedade no painel da Java virtual machine do console de administração.

   Consulte sua documentação do servidor de aplicativos da web para obter instruções específicas sobre configuração de propriedades Java.
2. Reinicie seu servidor de aplicativos da web se ele já estava em execução ou inicie seu servidor de aplicativos da web agora, para assegurar que as novas propriedades Java sejam usadas.

Resultados

Quando o servidor de aplicativos da web tiver concluído seu processo de inicialização, você terá concluído a instalação do Web Connector no servidor de runtime. A próxima etapa é conectar-se a sua página da web Configuração em `http://<host>:<port>/interact/jsp/WebConnector.jsp`, em que `<host>` é o nome do servidor de runtime do Interact e `<port>` é a porta na qual o Web Connector está atendendo, conforme especificado pelo servidor de aplicativos da web.

Instalando o Conector da Web como um aplicativo da web separado

Uma instância do Web Connector é instalada automaticamente com seu servidor de runtime do IBM Interact e é ativada por padrão. Entretanto, também é possível implementar o Conector da Web como seu próprio aplicativo da web (por exemplo, em um servidor de aplicativos da web em um sistema separado) e configurá-lo para se comunicar com o servidor de runtime do Interact remoto.

Sobre Esta Tarefa

Estas instruções descrevem o processo de implementar o Conector da Web como um aplicativo da web separado com acesso a um servidor de runtime do Interact remoto.

Antes de poder implementar o Conector da Web, você deve ter instalado o servidor de runtime do IBM Interact e você deve ter um servidor de aplicativos da

web em outro sistema com acesso à rede (não bloqueado por firewall) ao servidor de runtime do Interact.

Procedimento

1. Copie o diretório `jsconnector` que contém os arquivos do Conector da Web de seu servidor de runtime do Interact no sistema em que o servidor de aplicativos da web (como o WebSphere Application Server) já está configurado e em execução. É possível localizar o diretório `jsconnector` em seu diretório de instalação do Interact.
2. No sistema em que você estará implementando a instância do Conector da Web, configure o arquivo `jsconnector/jsconnector.xml` usando qualquer texto ou editor de XML para modificar o atributo `interactURL`.

   Isto é configurado por padrão para `http://localhost:7001/interact`, mas você deve modificá-lo para que corresponda à URL do servidor de runtime remoto do Interact, como `http://runtime.example.com:7011/interact`.

   Após implementar o Conector da Web, é possível usar uma interface da web para customizar as configurações restantes no arquivo `jsconnector.xml`. Consulte "Configurando o Web Connector" na página 307 para obter mais informações.
3. Para o servidor de aplicativos da web no qual você estará implementando o Conector da Web, configure a propriedade Java a seguir:

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Substitua `<jsconnectorHome>` pelo caminho real para onde copiou o diretório `jsconnector` no servidor de aplicativos da web.

   A forma na qual você configura as propriedades Java depende de seu servidor de aplicativos da web. Por exemplo, no WebLogic, você editaria o arquivo `startWebLogic.sh` ou `startWebLogic.cmd` para atualizar a configuração `JAVA_OPTIONS`, como neste exemplo:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/InteractFiles/
   jsconnector"
   ```

   No WebSphere Application Server, você configuraria esta propriedade no painel da Java virtual machine do console de administração.

   Consulte sua documentação do servidor de aplicativos da web para obter instruções específicas sobre configuração de propriedades Java.
4. Reinicie seu servidor de aplicativos da web se ele já estava em execução ou inicie seu servidor de aplicativos da web nesta etapa para assegurar que a nova propriedade Java seja usada.

   Aguarde até que o servidor de aplicativos da web conclua seu processo de inicialização antes de continuar.
5. Conecte-se à sua interface de gerenciamento do servidor de aplicativos da web com os privilégios necessários para implementar um aplicativo.
6. Siga as instruções para seu servidor de aplicativos da web a fim de implementar e executar o arquivo a seguir: `jsConnector/jsConnector.war`

Resultados

O Conector da Web agora é implementado no aplicativo da web. Após ter configurado totalmente o servidor do Interact como ativo e em execução, a próxima etapa é conectar-se à página da web Configuração do Conector da Web em `http://` *`<host>`*`:` *`<port>`*`/interact/jsp/WebConnector.jsp`, em que *`<host>`* é o sistema que está executando o servidor de aplicativos da web no qual você acabou de implementar o Conector da Web e *`<port>`* é a porta na qual o Conector da Web

está atendendo, conforme especificado pelo servidor de aplicativos da web.

Configurando o Web Connector

As definições de configuração para o Conector da Web do Interact são armazenados em um arquivo chamado `jsconnector.xml` que é armazenado no sistema em que o Conector da Web está implementado (como o próprio servidor de runtime do Interact ou um sistema separado executando um servidor de aplicativos da web). É possível editar o arquivo `jsconnector.xml` diretamente usando qualquer editor de texto ou editor de XML; no entanto, uma maneira mais fácil de configurar quase todas as definições de configuração disponíveis é usar a página Configuração do Conector da Web a partir de seu navegador da web.

Antes de Iniciar

Antes de poder usar a interface da web para configurar o Conector da Web, você deve instalar e implementar o aplicativo da web que fornece o Conector da Web. No servidor de runtime do Interact, uma instância do Conector da Web é instalado automaticamente ao instalar e implementar o Interact. Em qualquer outro servidor de aplicativos da web, você deve instalar e implementar o aplicativo da web do Conector da Web conforme descrito no "Instalando o Conector da Web como um aplicativo da web separado" na página 305.

Procedimento

1. Abra seu navegador da web suportado e abra uma URL semelhante ao seguinte:

   `http://<host>:<port>/interact/jsp/WebConnector.jsp`

   - Substitua *<host>* pelo servidor no qual o Conector da Web está em execução, como o nome do host do servidor de runtime ou o nome do servidor no qual você implementou uma instância separada do Conector da Web.
   - Substitua *<port>* pelo número da porta no qual o aplicativo da web do Conector da Web está atendendo conexões, que geralmente corresponde à porta padrão do servidor de aplicativos da web.
2. Na página Configurações que aparece, conclua as seções a seguir:

Tabela 31. Resumo de configurações do Conector da Web.

Seção	Configurações
Configurações básicas	Use a página Configurações básicas para configurar o comportamento geral do Conector da Web para o site no qual você estará rolando as páginas identificadas. Essas configurações incluem a URL base para o site, informações que o Interact precisa usar sobre os visitantes do site e configurações semelhantes que se aplicam a todas as páginas que planeja identificar com o código do Conector da Web. Consulte "Opções básicas de configuração do WebConnector" na página 309 para obter detalhes.
Tipos de exibição HTML	Use a página Tipos de exibição HTML para determinar o determinar que será fornecido para cada ponto de interação na página. É possível escolher na lista de modelos padrão (arquivos .flt) que contêm alguma combinação de código de folha de estilo em cascata (CSS), código HTML e código Javascript para usar para cada ponto de interação. É possível usar os modelos conforme fornecido, customizar conforme necessário ou criar seus próprios. As definições de configuração nesta página correspondem à seção `interactionPoints` do arquivo de configuração `jsconnector.xml`. Consulte "Tipos de exibição HTML de configuração do WebConnector" na página 310 para obter detalhes.
Páginas aprimoradas	Use a página Páginas aprimoradas para mapear as configurações específicas de página para um padrão de URL. Por exemplo, você pode configurar um mapeamento de página de modo que qualquer URL que contenha o texto "index.htm" exiba sua página de boas-vindas geral, com eventos de carregamento de página específicos e pontos de interação definidos para esse mapeamento. As definições de configuração nesta página correspondem à seção `pageMapping` do arquivo de configuração `jsconnector.xml`. Consulte "Páginas aprimoradas de configuração do WebConnector" na página 313 para obter detalhes.

3. Na página Configurações básicas, verifique se as configurações de todo o site são válidas para sua instalação e especifique opcionalmente o modo de depuração (não recomendado a menos que esteja resolvendo um problema), a integração de Identificação de página do Digital Analytics for On Premises e as configurações padrão para a maioria dos Pontos de interação, em seguida, clique no link Tipos de exibição HTML em Configurações.
4. Na página Tipos de exibição HTML, siga estas etapas para incluir ou modificar modelos de exibição que definem os pontos de interação na página da web do cliente.

   Por padrão, modelos de exibição (arquivos .flt) são armazenados em *<jsconnector_home>*/conf/html.

   a. Selecione o arquivo .flt na lista que deseja examinar ou usar como seu ponto de interação ou clique em Incluir um tipo para criar um novo modelo de Ponto de interação em branco para uso.

      Informações sobre o conteúdo do modelo, se houver, aparecerão próximas à lista modelo.

   b. Opcionalmente, modifique o nome do modelo no campo **Nome de arquivo para este tipo de exibição**. Para um novo modelo, atualize CHANGE_ME.flt para algo mais significativo.

      Se você renomear o modelo aqui, o Conector da Web criará um novo arquivo com o nome na próxima vez que o modelo for salvo. Os modelos serão salvos ao modificar o corpo do texto e, em seguida, navegue a qualquer outro campo.

   c. Modifique ou conclua as informações do fragmento de HTML conforme necessário, incluindo qualquer folha de estilo (CSS), JavaScript e código HTML que deseja incluir. Observe que também é possível incluir variáveis que serão substituídas pelos parâmetros do Interact no tempo de execução. Por exemplo, ${offer.HighlightTitle} é automaticamente substituído pelo título da oferta no local especificado do Ponto de Interação.

      Use os exemplos que aparecem abaixo do campo Fragmento de HTML para indicações de como formatar seus blocos de código CSS, JavaScript ou HTML.
5. Use a página Páginas aprimoradas conforme necessário para configurar os mapeamentos de página que determinam o grau de especificação de padrões URL manipuladas nas páginas.
6. Quando você tiver concluído a configuração das propriedades de configuração, clique em **Apresentar mudanças**. Clicar em **Apresentar mudanças** executará as ações a seguir:
   - Exibe a tag de página do Conector da Web do IBM Interact, que contém o código JavaScript que você pode copiar da página do Conector da Web e inserir em suas páginas da web.
   - Faz backup do arquivo de configuração do Conector da Web existente no servidor Interact (o arquivo jsconnector.xml no servidor em que o Conector da Web está instalado) e cria um novo arquivo de configuração do conector com as configurações que você definiu.

     Os arquivos de configuração de backup são armazenados no *<jsconnector_home>*/conf/archive/jsconnector.xml.*<date>*.*<time>*, como em jsconnector.xml.20111113.214933.750-0500 (em que a sequência de datas é 20111113 e a sequência de tempo, incluindo o indicador de fuso horário, é 214933.750-0500)

Resultados

Agora você concluiu a configuração do Conector da Web.

Para modificar sua configuração, é possível retornar ao início destas etapas e executá-las novamente com novos valores ou é possível abrir o arquivo de configuração (`<Interact_home>/jsconnector/conf/jsconnector.xml`) em qualquer texto ou editor XML e modificá-lo conforme necessário.

Opções básicas de configuração do WebConnector

Use a página Configurações básicas da página Configurações do Conector da Web para configurar o comportamento geral do Conector da Web para o site no qual você rolará as páginas identificadas. Essas configurações incluem a URL base para o site, informações que o Interact precisa usar sobre os visitantes do site e configurações semelhantes que se aplicam a todas as páginas que planeja identificar com o código do Conector da Web.

Configurações em todo o site

As opções de configuração Configurações em todo o site são globais e afetam o comportamento geral da instalação do Conector da Web que você está configurando. É possível especificar os valores a seguir:

Tabela 32. Configurações em todo o site para a instalação do Conector da Web

Configuração	Descrição	Configuração equivalente em jsconnector.xml
URL de API do Interact	A URL base do servidor de runtime do Interact. **Nota:** Esta configuração é usada somente se o Conector da Web não estiver executando dentro do servidor de runtime do Interact (isto é, você implementou-o separadamente).	*<interactURL>*
URL de Conector da Web	A URL base usada para gerar a URL de click-through.	*<jsConnectorURL>*
Nome do canal interativo para o website de destino	O nome do canal interativo que você definiu no servidor Interact que representa este mapeamento de página.	*<interactiveChannel>*
Nível de público de visitantes	O nível de público do Campaign para o visitante de entrada; usado na chamada API para o tempo de execução do Interact.	*<audienceLevel>*
Nome do campo de ID de público na tabela de perfis	Nome do campo de ID de público que será usado na chamada API para o Interact. Observe que não há suporte atualmente para identificadores de público de vários campos.	*<audienceIdField>*
Tipo de dados do campo de ID de público	O tipo de dados do campo ID de público ("numérico" ou "sequência") a ser usado na chamada API para o Interact.	*<audienceIdFieldType>*

Tabela 32. Configurações em todo o site para a instalação do Conector da Web (continuação)

Configuração	Descrição	Configuração equivalente em jsconnector.xml
Nome do cookie que representa o ID de sessão	O nome do cookie que conterá o ID de sessão.	*<sessionIdCookie>*
Nome do cookie que representa o ID do visitante	O nome do cookie que conterá o ID do visitante.	*<visitorIdCookie>*

Recursos opcionais

As opções de configuração Recursos opcionais são configurações globais opcionais para a instalação do Conector da Web que você está configurando. É possível especificar os valores a seguir:

Tabela 33. Configurações em todo o site opcionais para a instalação do Conector da Web

Configuração	Descrição	Configuração equivalente em jsconnector.xml
Ativar modo de depuração	Especifica (com uma resposta `yes` ou `no`) se deseja usar um modo de depuração especial. Se você ativar este recurso, o conteúdo retornado do Conector da Web incluirá uma chamada Javascript para 'alertar', informando o cliente do mapeamento de página particular que acabou de ocorrer. O cliente deve ter uma entrada no arquivo especificado pela configuração *<authorizedDebugClients>* a fim de obter o alerta.	*<enableDebugMode>*
Arquivo de host do cliente de depuração autorizado	O caminho para um arquivo que contém a lista de hosts ou endereços IP (Internet Protocol) que se qualificam para o modo de depuração. Um nome do host do cliente ou endereço IP deve aparecer no arquivo especificado para que informações sobre depuração sejam coletadas.	*<authorizedDebugClients>*
Ativar integração de tag de página do Digital Analytics for On Premises	Especifica (com uma resposta `yes` ou `no`) se o Conector da Web deve anexar a identificação do IBM Digital Analytics for On Premises no fim do conteúdo da página.	`<enableNetInsightTagging>`
Arquivo de modelo HTML de identificação do Digital Analytics for On Premises	O modelo HTML/Javascript usado para integrar uma chamada à identificação do Digital Analytics for On Premises. Em geral, você deve aceitar a configuração padrão a menos que seja instruído a fornecer um modelo diferente.	`<netInsightTag>`

Tipos de exibição HTML de configuração do WebConnector

Use a página Tipos de exibição HTML para determinar o determinar que será fornecido para cada ponto de interação na página. É possível escolher na lista de modelos padrão (arquivos .flt) que contêm alguma combinação de código de folha

de estilo cascata (CSS), código HTML e código JavaScript para usar cada ponto de interação. É possível usar os modelos conforme fornecido, customizar conforme necessário ou criar seus próprios.

Nota: As definições de configuração nesta página correspondem à seção `interactionPoints` do arquivo de configuração `jsconnector.xml`.

O ponto de interação também pode conter marcadores (zonas) em que os atributos de oferta podem ser eliminados automaticamente. Por exemplo, é possível incluir `${offer.TREATMENT_CODE}` que seria substituído pelo código de tratamento designado a essa oferta durante a interação.

Os modelos que aparecem nesta página são carregados automaticamente a partir dos arquivos armazenados no diretório `<Interact_home>/jsconnector/conf/html` do servidor do Web Connector. Todos os novos modelos que criar aqui também serão armazenados nesse diretório.

Para usar a página Tipos de exibição HTML para visualizar ou modificar qualquer um dos modelos existentes, selecione o arquivo `.flt` na lista.

Para criar um novo modelo na página Tipos de exibição HTML, clique em **Incluir um tipo**.

Independentemente do método que você escolher para criar ou modificar um modelo, as informações a seguir aparecerão próximas à lista modelo:

Configuração	Descrição	Configuração equivalente em jsconnector.xml
Nome do arquivo para este tipo de exibição	O nome designado ao modelo que está editando. Esse nome deve ser válido para o sistema operacional no qual o Web Connector está em execução; por exemplo, não é possível usar uma barra (/) no nome se o sistema operacional for Microsoft Windows. Se você estiver criando um novo modelo, este campo será pré-configurado para `CHANGE_ME.flt`. Você deve alterar isto para um valor significativo antes de continuar.	*<htmlSnippet>*

Configuração	Descrição	Configuração equivalente em jsconnector.xml
Fragmento de HTML	O conteúdo específico que o Web Connector deve inserir no Ponto de interação na página da web. Esse fragmento pode conter o código HTML, informações de formatação CSS ou JavaScript a serem executados na página. Cada um desses três tipos de conteúdo deve ser colocado em códigos BEGIN e END, como nos exemplos a seguir: • ${BEGIN_HTML} <*seu conteúdo HTML*> ${END_HTML} • ${BEGIN_CSS} <*suas informações de folha de estilo específicas do Ponto de interação*> ${END_CSS} • ${BEGIN_JAVASCRIPT} <*seu código Javascript específico do Ponto de interação*> ${END_JAVASCRIPT} Também é possível inserir um número de códigos especiais predefinidos que são substituídos automaticamente quando a página é carregada, incluindo o seguinte: • `${logAsAccept}`: Uma macro que usa dois parâmetros (uma URL de destino e o TreatmentCode usado para identificar a aceitação da oferta) e a substitui pela URL de click-through. • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}` • `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` Cada um dos códigos de oferta listados aqui representam atributos de oferta definidos no modelo de oferta no IBM Campaign que foi usado pelo comerciante para criar as ofertas que o Interact está retornando. Observe que o Web Connector usa um mecanismo de modelo chamado FreeMarker que fornece muitas opções adicionais que você pode considerar úteis na configuração de códigos em seus modelos de página. Consulte http://freemarker.org/docs/index.html para obter mais informações.	Sem equivalente, porque o fragmento de HTML reside em seu próprio arquivo separado do jsconnector.xml.

Configuração	Descrição	Configuração equivalente em jsconnector.xml
Exemplo de códigos especiais	Contém amostras do tipo de códigos especiais, incluindo códigos que identificam blocos como HTML, CSS ou JAVASCRIPT e zonas para soltar elementos arrastados que é possível inserir para referir-se a metadados de oferta específicos.	Sem equivalente.

Suas mudanças nesta página são salvas automaticamente ao navegar para outra página de configuração do Web Connector.

Páginas aprimoradas de configuração do WebConnector

Use a página Páginas aprimoradas para mapear as configurações específicas de página para um padrão de URL. Por exemplo, é possível configurar um mapeamento de página de forma que qualquer URL recebida que contenha o texto "index.htm" exiba sua página de boas-vindas geral, com eventos de carregamento de página específicos e pontos de interação definidos para esse mapeamento.

Nota: As definições de configuração nesta página correspondem à seção `pageMapping` do arquivo de configuração `jsconnector.xml`.

Para usar a página Páginas aprimoradas para criar um novo mapeamento de página, clique no link **Incluir uma página** e conclua as informações necessárias para o mapeamento.

Informações da página

As opções de configuração das Informações da página para o mapeamento de página definem o padrão de URL que atua como o acionador para esse mapeamento, mais algumas configurações adicionais para a forma como esse mapeamento de página é manipulado pelo Interact.

Configuração	Descrição	Configuração equivalente em jsconnector.xml
URL contém	Este é o padrão de URL que o Web Connector deve observar na solicitação de página de entrada. Por exemplo, se a URL de solicitação contiver "mortgage.htm", você pode corresponder isso a sua página de informações de hipoteca.	*<urlPattern>*
Nome fácil para esta página ou conjunto de páginas	Um nome significativo para sua própria referência que descreve a que este mapeamento de página é destinado, como "Página de informações de hipoteca".	*<friendlyName>*
Retornar também ofertas como dados JSON para uso de JavaScript	Uma lista suspensa para indicar se você deseja que o Web Connector inclua os dados de oferta brutos no formato JavaScript Object Notation (http://www.json.org/) no fim do conteúdo da página.	*<enableRawDataReturn>*

Eventos para disparar (onload) quando uma visita é feita para esta página ou conjunto de páginas

Esse conjunto de opções de configuração para o mapeamento de página define o padrão de URL que atua como o acionador para esse mapeamento, mais algumas configurações adicionais para a forma como esse mapeamento de página é manipulado pelo Interact.

Nota: As definições de configuração nesta seção correspondem à seção `<pageLoadEvents>` do `jsconnector.xml`.

Configuração	Descrição	Configuração equivalente em jsconnector.xml
Eventos individuais	Uma lista de eventos que estão disponíveis para esta página ou conjunto de páginas. Os eventos nessa lista são aqueles que você definiu no Interact, selecione um ou mais eventos que deseja ocorrer quando a página for carregada. A sequência de chamadas API do Interact é como a seguir: 1. `startSession` 2. `postEvent` para cada evento de carregamento de página individual (contanto que você tenha definido os eventos individuais no Interact) 3. Para cada ponto de interação: • `getOffers` • `postEvent(ContactEvent)`	*<event>*

Pontos de interação (locais de exibição de oferta) nesta página ou conjunto de páginas

Esse conjunto de opções de configuração para o mapeamento de página permitem selecionar quais Pontos de interação aparecem na página do Interact.

Nota: As definições de configuração nesta seção correspondem à seção `<pageMapping>` | `<page>` | `<interactionPoints>` de `jsconnector.xml`.

Configuração	Descrição	Configuração equivalente em jsconnector.xml
Caixa de opção do nome do Ponto de interação	Cada Ponto de interação que foi definido no arquivo de configuração aparece nesta seção da página. Selecionar a caixa de opção próxima ao nome do Ponto de interação exibe um número de opções disponíveis para esse Ponto de interação.	*<interactionPoint>*

Configuração	Descrição	Configuração equivalente em jsconnector.xml
ID do elemento HTML (Interact configurará o innerHTML)	O nome do elemento HTML que deve receber o conteúdo para este Ponto de interação. Por exemplo, se você especificou `<div id="welcomebanner">` na página, você inseriria `welcomebanner` (o valor de ID) neste campo.	*<htmlElementId>*
Tipo de exibição HTML	Uma lista suspensa que permite selecionar o Tipo de exibição HTML (os fragmentos de HTML ou arquivos. flt, definidos em uma página anterior de configuração do Web Connector) para usar para este Ponto de interação.	*<htmlSnippet>*
Número máximo de ofertas para apresentar (se for um carrossel ou flipbook)	O número máximo de ofertas que o Web Connector deve recuperar do servidor do Interact para este Ponto de interação. Este campo é opcional e se aplica somente a um Ponto de interação que atualiza regularmente as ofertas apresentadas sem recarregar a página, como no cenário de carrossel emque diveras ofertas são recuperadas para que possam ser disponibilizadas uma por vez.	*<maxNumberOfOffers>*
Evento para disparar quando a oferta é apresentada	O nome do evento de contato a ser postado para este Ponto de interação.	*<contactEvent>*
Evento para disparar quando a oferta é aceita	O nome do evento de aceitação a ser postado para este Ponto de interação no momeno em que o link de oferta é clicado.	*<acceptEvent>*
Evento para disparar quando a oferta é rejeitada	O nome do evento de rejeição a ser postado para este Ponto de interação. **Nota:** Neste momento, este recurso ainda não foi usado	*<rejectEvent>*

Opções de configuração do Web Connector

Em geral, é possível usar uma interface gráfica do Web Connector para configurar seu Web Connector. Todas as configurações que especificar também serão armazenadas em um arquivo chamado jsconnector.xml, localizado em seu diretório `jsconnector/conf`. Cada um dos parâmetros que são salvos no arquivo de configuração jsconnector.xml está descrito aqui.

Parâmetros e suas descrições

Os parâmetros a seguir são armazenados no arquivo `jsconnector.xml` e são usados para interações do Web Connector. Há duas formas de modificar essas configurações:

- Usando a página da web Configuração do Web Connector que fica disponível automaticamente após ter implementado e iniciado o aplicativo Web Connector. Para usar a página da web Configuração, use seu navegador da web para abrir uma URL semelhante ao seguinte: `http://<host>:<port>/interact/jsp/WebConnector.jsp`.

As mudanças que fizer na página da web Administração serão armazenadas no arquivo `jsconnector.xml` no servidor em que o Web Connector está implementado.

- Edite o arquivo `jsconnector.xml` diretamente usando qualquer editor de texto ou editor de XML. Assegure-se de estar familiarizado com a edição de identificações XML e valores antes de usar este método.

  Nota: Toda vez que editar o arquivo `jsconnector.xml` manualmente, poderá recarregar essas configurações abrindo a página Administração do Web Connector (localizada em `http://<host>:<port>/interact/jsp/jsconnector.jsp`) e clicando em **Recarregar configuração**.

A tabela a seguir descreve as opções de configuração que podem ser configuradas conforme aparecerem no arquivo `jsconnector.xml`.

Tabela 34. Opções de configuração do Web Connector

Grupo de parâmetros	Parâmetro	Descrição
`defaultPageBehavior`		
	`friendlyName`	Um identificador legível por humanos para o Padrão de URL para exibição na página da web Configuração do Web Connector.
	`interactURL`	A URL base do servidor de runtime do Interact. Nota: é necessário configurar este parâmetro apenas se o serviço do Web Connector (jsconnector) estiver em execução como um aplicativo da web implementado. Não é necessário configurar esse parâmetro se o WebConnector estiver em execução automaticamente como parte do servidor de runtime do Interact.
	`jsConnectorURL`	A URL base usada para gerar a URL de click-through, como `http://host:port/jsconnector/clickThru`
	`interactiveChannel`	Nome do canal interativo que representa este mapeamento de página.
	`sessionIdCookie`	Nome do cookie que contém o ID de sessão que é usado nas chamadas API para o Interact.
	`visitorIdCookie`	Nome do cookie para conter o ID de público.
	`audienceLevel`	O nível de público da campanha para o visitante de entrada, usado na chamada API para o tempo de execução do Interact.
	`audienceIdField`	Nome do campo `audienceId` usado na chamada API para o tempo de execução do Interact. **Nota:** Nota: não há suporte atualmente para identificadores de público de vários campos.
	`audienceIdFieldType`	O tipo de dados do campo do ID de público [`numeric` | `string`] usado na chamada API para o tempo de execução do Interact

Tabela 34. Opções de configuração do Web Connector (continuação)

Grupo de parâmetros	Parâmetro	Descrição
	audienceLevelCookie	Nome do cookie que contém o nível de público. Isto é opcional. Se não configurar este parâmetro, o sistema usará o que está definido para audienceLevel.
	relyOnExistingSession	Usado na chamada API para o tempo de execução do Interact. Em geral, esse parâmetro é configurado como "true".
	enableInteractAPIDebug	Usado na chamada API para o tempo de execução do Interact para ativar a saída de depuração para os arquivos de log.
	pageLoadEvents	O evento que será postado uma vez que esta determinada página for carregada. Especifique um ou mais eventos nesta identificação, no formato semelhante a <event>event1</event>.
	interactionPointValues	Todos os itens nesta categoria atuam como valores padrão para valores ausentes nas categorias específicas de IP.
	interactionPointValuescontactEvent	Nome padrão do evento de contato a ser postado para este determinado ponto de interação.
	interactionPointValuesacceptEvent	Nome padrão do evento de aceitação a ser postado para este determinado ponto de interação.
	interactionPointValuesrejectEvent	Nome padrão do evento de rejeição a ser postado para este determinado ponto de interação. (Observe: neste momento, este recurso não é usado.)
	interactionPointValueshtmlSnippet	Nome padrão do modelo HTML a ser atendido para este ponto de interação.
	interactionPointValuesmaxNumberOfOffers	Número máximo padrão de ofertas a serem recuperadas do Interact para este ponto de interação.
	interactionPointValueshtmlElementId	Nome padrão do elemento HTML para receber o conteúdo para este ponto de interação.
	interactionPoints	Esta categoria contém a configuração para cada ponto de interação. Para quaisquer propriedades ausentes, o sistema confiará no que está configurado sob a categoria interactionPointValues.
	interactionPointname	Nome do Ponto de interação (IP).
	interactionPointcontactEvent	Nome do evento de contato a ser postado para este determinado IP.
	interactionPointacceptEvent	Nome do evento de aceitação a ser postado para este determinado IP.
	interactionPointrejectEvent	Nome do evento de rejeição a ser postado para este determinado IP. (Observe que este recurso não está em uso ainda.)

Tabela 34. Opções de configuração do Web Connector (continuação)

Grupo de parâmetros	Parâmetro	Descrição
	interactionPointhtmlSnippet	Nome do modelo HTML a ser atendido para este IP.
	interactionPointmaxNumberOfOffers	Número máximo de ofertas a serem recuperadas de Interact para este IP
	interactionPointhtmlElementId	Nome do elemento HTML a receber o conteúdo para este ponto de interação.
	enableDebugMode	Sinalização booleana (valores aceitáveis: true ou false) para ativar o modo de depuração especial. Se você configurar para true, o conteúdo retornado do Conector da Web incluirá uma chamada JavaScript para 'alertar'informando o cliente do determinado mapeamento de página que acabou de ocorrer. O cliente deve ter uma entrada no arquivo authorizedDebugClients para gerar o alerta.
	authorizedDebugClients	Um arquivo usado pelo modo de depuração especial que contém a lista de nomes de host ou endereços de Internet Protocol (IP) que se qualificam para o modo de depuração.
	enableRawDataReturn	Uma sinalização booleana (valores aceitáveis: true ou false) para determinar se o Conector da Web anexa os dados de oferta brutos em formato JSON no fim do rodapé do conteúdo.
	enableNetInsightTagging	Uma sinalização booleana (valores aceitáveis: true ou false) para determinar se o Conector da Web anexa uma identificação do Digital Analytics for On Premises no fim do conteúdo.
	apiSequence	Representa uma implementação da interface APISequence, que dita a sequência de chamadas API feitas pelo Conector da Web quando um pageTag é chamado. Por padrão, a implementação usa uma sequência de StartSession, pageLoadEvents, getOffers e logContact, em que os dois últimos são específicos a cada Ponto de Interação.
	clickThruApiSequence	Representa uma implementação da interface APISequence, que dita a sequência das chamadas API feitas pelo Conector da Web quando um clickThru é chamado. Por padrão, a implementação usa uma sequência de StartSession e logAccept.
	netInsightTag	Representa o modelo HTML e JavaScript usado para integrar uma chamada à identificação do Digital Analytics for On Premises. Em geral, talvez não precise alterar esta opção.

Usando a página de administrador do Web Connector

O Web Connector inclui uma página de administração que fornece algumas ferramentas para ajudar a gerenciar e testar a configuração, pois pode ser usado com padrões de URL específicos. Você também pode usar a Página do Administrador para recarregar uma configuração que você modificou.

Sobre a Página do administrador

Usando qualquer navegador da web suportado, é possível abrir `http://host:port/interact/jsp/jsconnector.jsp`, em que *host:port* é o nome do host no qual o Web Connector está em execução e a porta na qual está atendendo a conexões, como `runtime.example.com:7001`

É possível usar a Página do administrador de uma das formas a seguir:

Tabela 35. Opções da página do administrador do Web Connector

Opção	Propósito
Recarregar configuração	Clique no link **Recarregar configuração** para recarregar quaisquer mudanças de configuração que foram salvas no disco na memória. Isto é necessário quando você fez mudanças diretamente no arquivo de configuração jsconnector.xml do Conector da Web em vez de usar as páginas da web de configuração.
Visualizar configuração	Visualize a configuração de WebConnector com base no padrão de URL que inserir no campo **Visualizar configuração**. Ao inserir a URL de uma página e clicar em **Visualizar configuração**, o Conector da Web retorna a configuração que o sistema usará com base neste mapeamento padrão. Se nenhuma correspondência for localizada, a configuração padrão será retornada. Isto é útil para teste se a configuração correta estiver sendo usada para uma determinada página.
Executar tag de página	Concluir os campos nesta página e clicar em **Executar tag de página** faz com que o Conector da Web retorne ao resultado `pageTag` baseado no padrão de URL. Isto simula a chamada de uma tag de página. A diferença entre chamar pageTag a partir desta ferramenta e usar um website real é que usar esta Página de Administração causará a exibição de erros ou exceções. Para um website real, as exceções não são retornadas e são visíveis apenas no arquivo de log do Conector da Web.

Página do Web Connector de amostra

Como um exemplo, um arquivo chamado `WebConnectorTestPageSA.html` foi incluído com o Interact Web Connector (no diretório `<Interact_Home/jsconnector/webapp/html`) que demonstra quantos dos recursos do Web Connector seriam identificados em uma página. Para conveniência, essa página de amostra também é mostrada aqui.

Página HTML do Web Connector de amostra

```
<?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[
/* ###############################################################
Esta é uma página de teste que contém WebConnector pageTag.  Como o
nome
deste arquivo possui TestPage integrado, o WebConnector detectará uma
correspondência de padrão URL
com o padrão URL "testpage" na versão
padrão de
jsconnector.xml - a definição de configuração mapeada para
o padrão URL "testpage"
se aplicará aqui. Isso significa que nesta
página deve haver os
ids de elemento html correspondentes
aos IPs para este padrão URL
```

```
(ex. 'welcomebanner', 'crosssellcarousel' e 'textservicemessage')
######################################################################## */

/* ########################################################################
Esta seção configura os cookies para sessionId e visitorId.
Observe que em um website de produção real, isto é feito mais
provavelmente pelo componente de
login. Para fins de teste, é feito aqui... o nome do cookie
tem de corresponder ao que está configurado no jsconnector xml.
######################################################################## */
     function setCookie(c_name,value,expiredays)
     {
       var exdate=new Date();
       exdate.setDate(exdate.getDate()+expiredays);
       document.cookie=c_name+ "=" +escape(value)+
       ((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
     }
     setCookie("SessionID","123");
     setCookie("CustomerID","1");

   /* ####################################################
   Agora configure os IDs de elemento HTML que correspondem aos IPs
   #################################################### */
     document.writeln("<div id='welcomebanner'> This should change, "
  + "otherwise something is wrong <\/div>");
     document.writeln("<div id='crosssellcarousel'> This should change, "
  + "otherwise something is wrong <\/div>");
     document.writeln("<div id='textservicemessage'> This should change, "
  + "otherwise something is wrong <\/div>");
   //]]&gt;
</script><!--
#############################################################################
isto é o que é colado do arquivo pageTag.txt no diretório conf da
instalação do WebConnector... a var unicaWebConnectorBaseURL precisa
ser
ajustada para estar em conformidade com seu ambiente local WebConnector
#############################################################################
-->
<!-- INÍCIO: Identificação de página do IBM Interact Web Connector -->
<!--
# ****************************************************************************
#  Materiais Licenciados - Propriedade da IBM
#  IBM Interact
#  (c) Copyright IBM Corporation 2001, 2012.
#  Direitos restritos para usuários do governo dos Estados Unidos - Uso, duplicação e divulgação
#  restritos pelo documento GSA ADP Schedule Contract com a 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>Página do Interact Web Connector de
amostra</title>
   </head>
   <body>
   <!-- FIM: Tag de página do IBM Interact Web Connector -->
   <!--
################################################################################
fim da colagem pageTag
```

```
################################################################################
-->
   </body>
</html>
```

Apêndice E. Integração do Interact e do Digital Recommendations

O IBM Interact pode ser integrado ao IBM Digital Recommendations para fornecer recomendações de produto geradas pelo Interact. Ambos os produtos podem oferecer recomendações de produto para ofertas, mas usando métodos diferentes. O Digital Recommendations usa o comportamento da web de um visitante (filtragem colaborativa) para construir correlações entre os visitantes e as ofertas recomendadas. O Interact é baseado no comportamento anterior do cliente, nos atributos, no histórico e menos em ofertas no nível de visualização, aprendizado que oferece a melhor correspondência de perfil de comportamento de um cliente (com base em informações demográficas e outras sobre o cliente). As taxas de aceitação de oferta ajudam a construir um modelo preditivo por meio de autoaprendizado. Usando o melhor de ambos os produtos, o Interact pode usar um perfil pessoal para definir as ofertas que transmitirão um ID de categoria ao Digital Recommendations e recuperar os produtos recomendados com base na popularidade (o "conhecimento popular") a serem exibidos ao visitante como parte das ofertas selecionadas. Assim, é possível fornecer melhores recomendações aos clientes que resultarão em mais click-throughs e melhores resultados do que se qualquer dos produtos agisse sozinho.

As seções a seguir descrevem como essa integração funciona e como usar o aplicativo de amostra fornecido para criar sua própria integração de oferta customizada.

Visão geral da integração do Interact ao Digital Recommendations

Esta seção descreve como o IBM Interact pode integrar-se ao IBM Digital Recommendations para fornecer recomendações do produto geradas pelo Interact, incluindo uma descrição do processo e os mecanismos pelos quais a integração ocorre.

O IBM Interact integra-se ao IBM Digital Recommendations por meio de uma interface de programação de aplicativos (API) Representational State Transfer (REST), disponibilizada a partir da instalação do Digital Recommendations. Ao fazer as chamadas da API REST com o ID de categoria apropriado, o Interact pode recuperar os produtos recomendados e incluí-los nas informações de oferta exibidas na página customizada que o visitante está visualizando.

Quando um visitante visualiza a URL da página da web (como a página JSP de amostra incluída com a instalação do Interact), a página chama o Interact para buscar uma oferta. Assumindo que a oferta tenha sido configurada dentro do Interact com os parâmetros corretos, ocorrerão as etapas a seguir, no caso mais simples:

1. A lógica de página identifica o ID do cliente do visitante.
2. Uma chamada de API para o Interact é feita, transmitindo a informações necessárias para gerar uma oferta para esse cliente.
3. A oferta retornada fornece a página da web com pelo menos três atributos: a URL da imagem da oferta, a URL da página de entrada quando o cliente clica e o ID de categoria a ser usado para determinar quais produtos são recomendados.

4. Em seguida, o ID de categoria é usado para chamada o Digital Recommendations para recuperar os produtos recomendados. Este conjunto de produtos está no formato JSON (JavaScript Object Notation) em ordem de produtos mais vendidos nessa categoria.
5. Em seguida, a oferta e os produtos são exibidos no navegador do visitante.

Essa integração é útil para combinar a recomendação da oferta e recomendações de produto. Por exemplo, em uma página da web, pode haver dois pontos de interação: um para um oferta e um para as recomendações que correspondem a essa oferta. Para isso, a página da web faz uma chamada para o Interact para fazer uma segmentação em tempo real para determinar a melhor oferta (por exemplo, para 10% de desconto em todos os dispositivos pequenos). Quando a página recebe a oferta do Interact, essa oferta contém ID de categoria (neste exemplo, para dispositivos pequenos). Em seguida, a página transmite o ID de categoria para dispositivos pequenos para o Digital Recommendations usando uma chamada de API e recebe como resposta as melhores recomendações de produto para essa categoria com base na popularidade.

Um exemplo mais simples é quando uma página da web faz uma chamada para o Interact somente para descobrir uma categoria (por exemplo, faqueiro de alta qualidade) que corresponda ao perfil do cliente. Em seguida, ela transmite o ID de categoria recebido para o Digital Recommendations, e obtém as recomendações do produto faqueiro.

Pré-requisitos de integração

Antes de poder usar a integração Digital Recommendations - Interact, você deverá assegurar-se de atender aos pré-requisitos descritos nesta seção.

Assegure-se de que os seguintes pré-requisitos sejam verdadeiros:

- Você está familiarizado com o uso da API do Interact, conforme documentado em outro local no *Guia do Administrador* e na ajuda online.
- Você está familiarizado com a API REST do Digital Recommendations, conforme descrito na documentação do desenvolvedor do Digital Recommendations.
- Você possui um entendimento básico de HTML, JavaScript, CSS e JSON (JavaScript Object Notation).

  JSON é importante porque a API REST do Digital Recommendations retorna as informações do produto solicitadas, como dados formatados como JSON.
- Você está familiarizado com a codificação do lado do servidor de páginas da web, porque o aplicativo de demonstração fornecido com o Interact usa JSP (embora o JSP não seja obrigatório).
- Você possui uma conta válida do Digital Recommendations e a lista de IDs de categoria que planeja ter para que o Interact recupere recomendações de produto (os produtos mais vendidos ou mais populares na categoria especificada).
- Você possui o link da API REST do Digital Recommendations (uma URL para o ambiente do Digital Recommendations).

  Consulte o aplicativo de amostra incluído com a instalação do Interact para obter um exemplo ou consulte o código de amostra em "Usando o Projeto de amostra de integração" na página 326 para obter mais informações.

Configurando uma oferta para a integração do Digital Recommendations

Antes que a página da web possa chamar o Digital Analytics Digital Recommendations para recuperar um produto recomendado, você deverá configurar a oferta do IBM Interact com as informações necessárias para transmitir ao Digital Recommendations.

Sobre Esta Tarefa

Para configurar uma oferta para ser vinculada ao Digital Recommendations, primeiro assegure-se de que as seguintes condições estejam em vigor:

- Assegure-se de que o servidor de runtime do Interact esteja corretamente configurado e em execução.
- Assegure-se de que o servidor de runtime possa estabelecer uma conexão com o servidor do Digital Recommendations, incluindo assegurar-se de que o firewall não impeça o estabelecimento de saída de uma conexão da web padrão (porta 80).

Para configurar uma oferta para integração com o Digital Recommendations, execute as seguintes etapas.

Procedimento

1. Crie ou edite uma oferta para o Interact.

   Para obter informações sobre a criação e a modificação de ofertas, consulte o *IBM Interact User's Guide* e a documentação do IBM Campaign.
2. Além das outras configurações na oferta, assegure-se de que a oferta inclua os atributos de oferta a seguir:
   - A URL (Localizador Uniforme de Recursos) vinculada à imagem da oferta.
   - A URL vinculada à página de entrada da oferta.
   - Um ID de categoria do Digital Recommendations associado a essa oferta.

     É possível recuperar o ID de categoria manualmente a partir da configuração do Digital Recommendations. O Interact não pode recuperar os valores de ID de categoria diretamente.

   No aplicativo da web de demonstração incluído com a instalação do Interact, esses atributos de oferta são chamados `ImageURL`, `ClickThruURL` e `CategoryID`. É possível usar quaisquer nomes que sejam significativos para você, desde que o aplicativo da web corresponda aos valores que a oferta está esperando.

   Por exemplo, é possível definir uma oferta chamada "10PercentOff" que contenha estes atributos, em que o ID de categoria (recuperado a partir da configuração do Digital Recommendations) seja `PROD1161127`, a URL do click-through da oferta seja `http://www.example.com/success` e a URL da imagem a ser exibida para a oferta seja `http://localhost:7001/sampleIO/img/10PercentOffer.jpg` (uma URL que, neste caso, seja local para o servidor de runtime do Interact).
3. Defina as regras de tratamento para um canal interativo no qual essa oferta será incluída e implemente o canal interativo como de costume.

Resultados

Agora, a oferta está definida com as informações necessárias para a integração do Digital Recommendations. O trabalho restante para permitir que o Digital

Recommendations forneça recomendações do produto ao Interact é realizado configurando as páginas da web para fazer as chamadas de API apropriadas.

Ao configurar o aplicativo da web para entregar a página integrada aos visitantes, assegure-se de que os seguintes arquivos sejam incluídos no diretório `WEB-INF/lib`:

- *Interact_Home*`/lib/interact_client.jar`, necessário para manipular chamadas da página da web para a API do Interact.
- *Interact_Home*`/lib/JSON4J_Apache.jar`, necessário para manipular os dados retornados da chamada para API REST do Digital Recommendations, que retorna os dados formatados como JSON.

Consulte "Usando o Projeto de amostra de integração" para obter informações adicionais sobre como entregar as ofertas aos clientes.

Usando o Projeto de amostra de integração

Cada instalação de tempo de execução do Interact inclui um projeto de amostra que demonstra o processo de integração Digital Recommendations - Interact. O projeto de amostra fornece uma demonstração completa, de ponta a ponta da criação de uma página da web que chame uma oferta que contenha um ID de categoria que, em seguida, será transmitido ao Digital Recommendations para recuperar uma lista de produtos recomendados para apresentação nos pontos de interação da página.

Visão geral

É possível usar o projeto de amostra incluído da maneira em que ele é fornecido, se desejar testar o processo de integração ou usá-lo como um ponto de início para desenvolver suas próprias páginas customizadas. O projeto de amostra está localizado no seguinte arquivo:

Interact_home`/samples/IntelligentOfferIntegration/MySampleStore.jsp`

Este arquivo, além de conter um exemplo de funcionamento integral do processo de integração, também contém comentários abrangentes que explicam o que deve ser configurado no Interact, o que deve ser customizado no arquivo `.jsp` e como implementar a página adequadamente para ser executada com a sua instalação.

MySampleStore.jsp

Por conveniência, o arquivo MySampleStore.jsp é mostrado aqui. Essa amostra poderá ser atualizada com as liberações subsequentes do Interact, portanto, use o arquivo incluído com a instalação como um ponto de início para os exemplos que precisar.

```
<!--
# ********************************************************************************
#  Licensed Materials - Property of IBM
#  IBM Interact
#  (c) Copyright IBM Corporation 2001, 2011.
#  US Government Users Restricted Rights - Use, duplication or disclosure
#  restricted by GSA ADP Schedule Contract with IBM Corp.
# ********************************************************************************

-->


<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page import="java.net.URL,
```

```
   java.net.URLConnection,
   java.io.InputStreamReader,
   java.io.BufferedReader,
   com.unicacorp.interact.api.*,
   com.unicacorp.interact.api.jsoverhttp.*,
   org.apache.commons.json.JSONObject,
   org.apache.commons.json.JSONArray" %>

<%

 /*****************************************************************************
 * This sample jsp program demonstrates integration of Interact and Digital Recommendations.
 *
 * When the URL for this jsp is accessed via a browser. the logic will call Interact
 * to fetch an Offer.  Based on the categoryID associated to the offer, the logic
 * will call Digital Recommendations to fetch recommended products.  The offer and products
 * will be displayed.
 * To toggle the customerId in order to demonstrate different offers, one can simply
 * append cid=<id> to the URL of this JSP.
 *
 * Prerequisites to understand this demo:
 * 1) familiarity of Interact and its java API
 * 2) familiarity of IntelligentOffer and its RestAPI
 * 3) some basic web background ( html, css, javascript) to mark up a web page
 * 4) Technology used to generate a web page (for this demo, we use JSP executed on the server side)
 *
 *
 * Steps to get this demo to work:
 * 1) set up an Interact runtime environment that can serve up offers with the following
 * offer attributes:
 *  ImageURL : url that links to the image of the offer
 *  ClickThruURL : url that links to the landing page of the offer
 *  CategoryID : Digital Recommendations category id associated to the offer
 * NOTE: alternate names for the attributes may be used as long as the references to those
 * attributes in this jsp are modified to match.
 * 2) Obtain a valid REST API URL to the Intelligent Offer environment
 * 3) Embed this JSP within a Java web application
 * 4) Make sure interact_client.jar is in the WEB-INF/lib directory (communication with Interact)
 * 5) Make sure JSON4J_Apache.jar (from interact install) is in the
  *    WEB-INF/lib directory (communication with IO)
 * 6) set the environment specific properties in the next two sections
 *****************************************************************************/

 /*****************************************************************************
 *   *****************CHANGE THESE SETTINGS TO REFLECT YOUR ENV*******************
 * Set your Interact environment specific properties here...
 *****************************************************************************/

 final String sessionId="123";
 final String interactiveChannel = "SampleIO";
 final String audienceLevel = "Customer";
 final String audienceColumnName="CustomerID";
 final String ip="ip1";
 int customerId=1;
 final String interactURL="http://localhost:7011/interact/servlet/InteractJSService";
 final boolean debug=true;
 final boolean relyOnExistingSession=true;

 /*****************************************************************************
   *****************CHANGE THESE SETTINGS TO REFLECT YOUR ENV*******************
 * Set your Digital Recommendations environment specific properties here...
 *****************************************************************************/

 final String ioURL="http://recs.coremetrics.com/iorequest/restapi";
 final String zoneID="ProdRZ1";
 final String cID="90007517";



 /*****************************************************************************
```

```
   ********************************************************************************/

 StringBuilder interactErrorMsg = new StringBuilder();
 StringBuilder intelligentOfferErrorMsg = new StringBuilder();

 // get the customerID if passed in as a parameter
 String cid = request.getParameter("cid");
 if(cid != null)
 {
  customerId = Integer.parseInt(cid);
 }

 // call Interact to get offer
 Offer offer=getInteractOffer(interactURL,sessionId,interactiveChannel,audienceLevel,
        audienceColumnName,ip,customerId,debug,relyOnExistingSession,interactErrorMsg);

 // get specific attributes from the offer (img url, clickthru url, & category id)
 String offerImgURL=null;
 String offerClickThru=null;
 String categoryId="";

 if(offer != null)
 {
  for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
     {
         if(offerAttribute.getName().equalsIgnoreCase("ImageURL"))
         {
          offerImgURL=offerAttribute.getValueAsString();
         }
         else if(offerAttribute.getName().equalsIgnoreCase("ClickThruURL"))
         {
          offerClickThru=offerAttribute.getValueAsString();
         }
         else if(offerAttribute.getName().equalsIgnoreCase("CategoryID"))
         {
          categoryId=offerAttribute.getValueAsString();
         }
     }
 }

 // call Digital Recommendations to get products
 JSONObject products=getProductsFromIntelligentOffer(ioURL, cID, zoneID, categoryId,
        intelligentOfferErrorMsg);

%>

<html>
 <head>
  <title>My Favorite Store</title>

  <script language="javascript" type="text/javascript">
   var unicacarousel=(function(){var g=false;var h;var j=0;var k=0;var l=0;var m=40;
          var n=new Array(0,2,6,20,40,60,80,88,94,97,99,100);var o=function(a){var b=a.parentNode;
          h=b.getElementsByTagName("UL")[0];var c=h.getElementsByTagName("LI");j=c[0].offsetWidth;
          k=c.length;l=Math.round((b.offsetWidth/j));unicacarousel.recenter()};var p=function(a)
          {var b=parseFloat(h.style.left);if(isNaN(b))b=0;for(var i=0;i<n.length;i++)
          {setTimeout("unicacarousel.updateposition("+(b+(a*(n[i]/100)))+");",((i*m)+50))}
          setTimeout("unicacarousel.recenter();",((i*m)+50))};return{gotonext:function(a,b)
          {if(!g){o(a);g=true;p((-1*b*j))}},gotoprev:function(a,b){if(!g){o(a);g=true;p((b*j))}},
          updateposition:function(a){h.style.left=a+"px"},recenter:function(){var a=parseFloat(h.style.left);
          if(isNaN(a))a=0;var b=j*Math.round(((l-k)/2));var c=Math.abs(Math.round((b-a)/j));
          if(a<b){var d=h.getElementsByTagName("LI");var e=new Array();
          for(var i=0;i<c;i++){e[e.length]=d[i]}for(var i=0;i<e.length;i++)
          {h.insertBefore(e[i],null)}unicacarousel.updateposition(b)}else
          if(a>b){var d=h.getElementsByTagName("LI");var e=new Array();
          for(var i=0;i<c;i++){e[e.length]=d[d.length-c+i]}var f=d[0];
          for(var i=0;i<e.length;i++){h.insertBefore(e[i],f)}unicacarousel.updateposition(b)}g=false}})();
  </script>

  <style type="text/css">
```

```
 .unicaofferblock_container {width:250px; position:relative; display:block;
                             text-decoration:none; color:#000000; cursor: pointer;}
 .unicaofferblock_container .unicateaserimage {margin:0px 0.5em 0.25em 0px; float:left;}
 .unicaofferblock_container .unicabackgroundimage {position:absolute; top:0px; left:0px;}
 .unicaofferblock_container .unicabackgroundimagecontent {width:360px; height:108px;
                   padding:58px 4px 4px 20px; position:relative; top:0px;}
 .unicaofferblock_container h4 {margin:0px; padding:0px; font-size:14px;}

 .unicacarousel {width:588px; position:relative; top:0px;}
 .unicacarousel_sizer {width:522px; height:349px; margin:0px 33px; padding:0;
                       overflow:hidden; position:relative;}
 .unicacarousel_rotater {height:348px; width:1000px; margin:0 !important;
                         padding:0; list-style:none; position:absolute; top:0px;
                         left:0px;}
 .unicacarousel li {width:167px; height:349px; float:left; padding:0 4px;
                    margin:0px !important; list-style:none !important;
                    text-indent:0px !important;}
 .unicacarousel_gotoprev, .unicacarousel_gotonext {width:18px; height:61px;
                     top:43px; background:url(../img/carouselarrows.png) no-repeat;
                    position:absolute; z-index:2; text-align:center; cursor:pointer;
                    display:block; overflow:hidden; text-indent:-9999px;
                    font-size:0px; margin:0px !important;}
 .unicacarousel_gotoprev {background-position:0px 0; left:0;}
 .unicacarousel_gotonext {background-position:-18px 0; right:0;}

  </style>


 </head>

 <body>

  <b>Welcome To My Store</b> Mr/Mrs. <%=customerId %>
  <br><br>
<% if(offer != null) { %>
 <!-- Interact Offer HTML -->

 <div onclick="location.href='<%=offerClickThru %>'" class="unicaofferblock_container">
  <div class="unicabackgroundimage">
   <a href="<%=offerClickThru %>"><img src="<%=offerImgURL %>" height="170"
         width="695" border="0"></a>
  </div>
 </div>


<% } else { %>
 No offer available.. <br> <br>
 <%=interactErrorMsg.toString() %>
<% } %>

<% if(products != null) { %>
 <!-- IntelligentOffer Products HTML -->
 <br><br><br> <br><br><br> <br><br><br> <br><br><br> <br>
 <div class="unicacarousel">
 <div class="unicacarousel_sizer">
  <ul class="unicacarousel_rotater">


 <% JSONArray recs = products.getJSONObject("io").getJSONArray("recs");
  if(recs != null)
  {
   for(int x=0;x< recs.length();x++)
   {
    JSONObject rec = recs.getJSONObject(x);
    if(rec.getString("Product Page") != null &&
              rec.getString("Product Page").trim().length()>0) {
    %>

    <li>
     <a href="<%=rec.getString("Product Page") %>" title="<%=rec.getString("Product Name") %>">
```

```
            <img src="<%=rec.getString("Image") %>"  width="166" height="148" border="0" />
     <%=rec.getString("Product Name") %>
     </a>
    </li>

    <% }
   }
  }
 %>
 </ul>
 </div>
 <p class="unicacarousel_gotoprev" onclick="unicacarousel.gotoprev(this,1);"></p>
 <p class="unicacarousel_gotonext" onclick="unicacarousel.gotonext(this,1);"></p>
</div>
<% } else { %>
 <div>
 <br><br> <br><br><br> <br><br><br> <br><br><br> <br>
 No products available...<br> <br>
 <%=intelligentOfferErrorMsg.toString() %>
 </div>
<% } %>

 </body>
</html>



<%!
 /*****************************************************************************
 * The following are convenience functions that will fetch from Interact and
 *   Digital Recommendations
 *****************************************************************************/

 /*****************************************************************************
 * Call Digital Recommendations to retrieve recommended products
 *****************************************************************************/
 private JSONObject getProductsFromIntelligentOffer(String ioURL, String cID,
     String zoneID, String categoryID, StringBuilder intelligentOfferErrorMsg)
 {

  try {

   ioURL += "?cm_cid="+cID+"&cm_zoneid="+zoneID+"&cm_targetid="+categoryID;
   System.out.println("CoreMetrics URL:"+ioURL);
   URL url = new java.net.URL(ioURL);

    URLConnection conn = url.openConnection();

    InputStreamReader inReader = new InputStreamReader(conn.getInputStream());
    BufferedReader in = new BufferedReader(inReader);

    StringBuilder response = new StringBuilder();

    while(in.ready())
    {
     response.append(in.readLine());
    }

    in.close();

    intelligentOfferErrorMsg.append(response.toString());

    System.out.println("CoreMetrics:"+response.toString());

    if(response.length()==0)
     return null;

    return new JSONObject(response.toString());
  }
  catch(Exception e)
```

```
 {
  intelligentOfferErrorMsg.append(e.getMessage());
  e.printStackTrace();
 }

 return null;

}

/*****************************************************************************
* Call Interact to retrieve offer
*****************************************************************************/
private Offer getInteractOffer(String interactURL,String sessionId,String interactiveChannel,
    String audienceLevel,
       String audienceColumnName,String ip, int customerId,boolean debug,
                      boolean relyOnExistingSession, StringBuilder interactErrorMsg)
{
 try {
  InteractAPI api = InteractAPI.getInstance(interactURL);
  NameValuePairImpl custId = new NameValuePairImpl();
     custId.setName(audienceColumnName);
      custId.setValueAsNumeric(Double.valueOf(customerId));
    custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
      NameValuePairImpl[] audienceId = { custId };

      // call startSession
     Response response = api.startSession(sessionId, relyOnExistingSession,
         debug, interactiveChannel, audienceId, audienceLevel, null);

     if(response.getStatusCode() == Response.STATUS_ERROR)
     {
      printDetailMessageOfWarningOrError("startSession",response, interactErrorMsg);
  }

     // call getOffers
     response = api.getOffers(sessionId, ip, 1);
     if(response == null  || response.getStatusCode() == Response.STATUS_ERROR)
     {
      printDetailMessageOfWarningOrError("getOffers",response, interactErrorMsg);
  }

           OfferList offerList=response.getOfferList();

           if(offerList != null && offerList.getRecommendedOffers() != null)
           {
            return offerList.getRecommendedOffers()[0];
           }
 }
 catch(Exception e)
 {
  interactErrorMsg.append(e.getMessage());
  e.printStackTrace();
 }
 return null;
}

   private void printDetailMessageOfWarningOrError(String command, Response response,
       StringBuilder interactErrorMsg)
   {
    StringBuilder sb = new StringBuilder();
       sb.append("Calling "+command).append("<br>");
       AdvisoryMessage[] messages = response.getAdvisoryMessages();

       for(AdvisoryMessage msg : messages)
       {
        sb.append(msg.getMessage()).append(":");
        sb.append(msg.getDetailMessage());
        sb.append("<br>");
```

```
        }
        interactErrorMsg.append(sb.toString());
    }
%>
```

Apêndice F. Integração do Interact e do Digital Data Exchange

Com o Digital Data Exchange, seu website pode vincular com Interact para fornecer um mecanismo de execução omni-channel poderoso que fornece as melhores ofertas para os canais ideais e evoluir (aprender) a partir do feedback da oferta para continuamente aumentar a efetividade do marketing.

É possível usar esta ferramenta se sua equipe de marketing usa o Interact para gerenciamento de ofertas omni-channel e deseja estender estas ofertas inteligentes personalizadas para seus websites.

O IBM Digital Data Exchange integra o IBM e soluções de marketing de terceiros com insights do cliente digital através de uma API de organização de dados em tempo real e uma solução de gerenciamento de tags corporativa.

Sem o IBM Digital Data Exchange, os comerciantes dependem da TI para vincular o Interact com seu website e chamar a API do Interact a partir de várias páginas da web. Com o IBM Digital Data Exchange, os comerciantes podem ignorar a TI e acessar o IBM Digital Data Exchange diretamente para incluir tags do IBM Digital Data Exchange em várias páginas da web.

Pré-Requisitos

Antes de poder usar a integração Interact e Digital Data Exchange, você deve certificar-se de atender aos pré-requisitos descritos nesta seção.

Certifique-se de que os pré-requisitos a seguir sejam verdadeiros.

- Você está familiarizado com a API JavaScript Interact conforme documentado em outro lugar no Guia do Administrador e na ajuda online.
- Você está familiarizado com a identificação do Digital Data Exchange e grupos de páginas.
- Você tem uma conta do Digital Data Exchange válida.
- Seu arquivo `interactapi.js` é publicamente hospedado, portanto, ele pode ser acessado nas configurações **Vendor**.

Integrando o IBM Interact com seu website através do IBM Digital Data Exchange

Use estas etapas para integrar o Interact com seu website através do Digital Data Exchange.

Procedimento

1. Especifique o local do arquivo `Interactapi.js`.
   a. Navegue até **Fornecedores** > **Configurações do Fornecedor** em Digital Data Exchange.
   b. Selecione IBM Interact no menu suspenso **Fornecedor**.
   c. No **Caminho da Biblioteca**, insira a URL na qual hospedou o `Interactapi.js`. Não inclua o protocolo (http ou https) nesta URL.
   d. No **Caminho para o Servlet Rest Público**, inclua o caminho no Servlet Rest.

2. Navegue até **Gerenciar** > **Configurações Globais** no Digital Data Exchange para especificar o nome do objeto para usar como o identificador da página no **Identificador de Página Exclusivo**. Por exemplo, é possível configurar o nome do objeto para `digitalData.pageInstanceID`.
3. Inclua o arquivo `eluminate.js` e um identificador na página da web na qual deseja Digital Data Exchange inserir as tags. Você deve fornecer a cada página da web um identificador exclusivo de forma que o Digital Data Exchange possa distinguir entre diversas páginas.

   Por exemplo, é possível incluir o script a seguir em sua página da web.

```
<!-- Setting Page Identifier -->
          <script>
                  digitalData={pageInstanceID:"INTERACT_HomePage"};
          </script>

<!-- Including eluminate script -->
         <script type="text/javascript" src="http://libs.
                  coremetrics.com/eluminate.js">
         </script>
         <script type="text/javascript">
                  cmSetClientID("51310000|INTERACTTEST",false,"data.
                  coremetrics.com",document.domain);
         </script>
```

4. Em tags de criação, segmentos de código, funções e outros itens do Digital Data Exchange que você deseja incluir em sua página da web.
5. Crie grupos de páginas para definir o você deseja arquivado em cada página.

   Veja o IBM Digital Data Exchange User Guide para obter mais informações.

Tags do Interact no Digital Data Exchange

Use as tags do Digital Data Exchange padrão para definir variações das tags que são apropriadas para páginas da web nas quais os dados são representados a partir de diferentes locais. Uma vez definidas, estas tags são incluídas na lista de tags do Interact. As tags podem não ter campos para definir ou podem não ter campos de tag requeridos e podem ser usadas diretamente.

As tags do Interact a seguir estão disponíveis no Digital Data Exchange sob **Tags**.

- Terminar Sessão
- Obter Ofertas
- Carregar Biblioteca
- Postar Evento
- Definir Público
- Iniciar Sessão

Para usar as tags Interact, edite as tags para definir o Campo de Tag, o Método, o Nome do Objeto, o Tipo de Dados e o Modificador para cada tag Interact.

As tags Postar Evento, Configurar Público e Iniciar Sessões aceitam campos de tag customizados. Use o ícone Incluir Campo de Tag, em seguida, clique no ícone Editar para definir o parâmetro customizado. O processo é o mesmo que qualquer definição de parâmetro, com a exceção de que o nome do parâmetro pode ser editado e deve incluir o nome do parâmetro, dois pontos e o tipo de dados do parâmetro. A ordem do parâmetro customizado na tag pode ser modificada com as setas para cima e para baixo.

As tags também podem ser ligadas a funções JavaScript ou objetos HTML de forma que eles disparem após a função disparar ou em um evento de objeto HTML.

Para obter mais informações sobre como definir, ligar e trabalhar com tags, veja o IBM Digital Data Exchange User Guide.

Para obter casos de uso detalhados da integração do Interact e Digital Data Exchange, veja https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Terminar Sessão

A tag Terminar Sessão marca o término de uma sessão da web.

Os campos de tag a seguir estão disponíveis para a tag Terminar Sessão.

Tabela 36. Tags Terminar Sessão

Campo de Tag	Descrição
*ID de Sessão	Identifica o ID de Sessão.
Nome da Função de Retorno de Chamada em Sucesso	Define o nome da função a ser chamada quando o método terminar sessão é bem-sucedido.
Nome da Função de Retorno de Chamada em Falha	Define o nome da função a ser chamada quando o método terminar sessão falha.

Qualquer **Campo de Tag** marcado com um * é requerido.

Obter Ofertas

Use a tag Obter Ofertas para solicitar ofertas do servidor de tempo de execução.

Os campos de tag a seguir estão disponíveis para a tag Obter Ofertas.

Tabela 37. Tags Obter Ofertas

Campo de Tag	Descrição
*ID de Sessão	Identifica o ID de Sessão.
*Nome do Ponto de Interação	Identifica o nome do ponto de interação que este método referencia. Esse nome deve corresponder exatamente ao nome do ponto de interação definido no canal interativo.
*Número Solicitado	Identifica o número de ofertas solicitadas.
Nome da Função de Retorno de Chamada em Sucesso	Define o nome da função a ser chamada quando o método obter ofertas é bem-sucedido.
Nome da Função de Retorno de Chamada em Falha	Define o nome da função a ser chamada quando o método obter ofertas falha.

Qualquer **Campo de Tag** marcado com um * é requerido.

A tag Obter Ofertas deve ser designada a um grupo de páginas cujo contêiner é configurado para `Default`.

Biblioteca de Carregamento

A tag Carregar Biblioteca carrega a biblioteca JavaScript Interact na seção do cabeçalho da página.

A tag Carregar Biblioteca não possui parâmetros. Ela leva o local da biblioteca do `Caminho da Biblioteca` em **Configurações do Fornecedor**. Ela deve ser incluída em um grupo de páginas usando o conjunto de contêineres para `Cabeçalho` e deve ser executada em cada página que possui uma identificação Interact.

Importante: Nenhuma das outras tags funcionará se a tag carregar biblioteca não estiver incluída. O interact.js não é carregado se esta tag não estiver incluída.

Postar Evento

Use a tag Postar Evento para executar qualquer evento definido no canal interativo.

Os campos de tag a seguir estão disponíveis para a tag Postar Evento.

Tabela 38. Tags Postar Evento

Campo de Tag	Descrição
*ID de Sessão	Identifica o ID de Sessão.
*Nome do Evento	Identifica o nome do evento. O nome do evento deve corresponder ao nome do evento conforme definido no canal interativo. Este nome não faz distinção entre maiúsculas e minúsculas.
Nome da Função de Retorno de Chamada em Sucesso	Define o nome da função a ser chamada quando o método postar evento é bem-sucedido.
Nome da Função de Retorno de Chamada em Falha	Define o nome da função a ser chamada quando o método postar evento falha.

Qualquer **Campo de Tag** marcado com um * é requerido.

Os parâmetros opcionais podem ser incluídos com o recurso de campo da tag customizada. Nomes de tag de customização consistem no nome do parâmetro, dois pontos e o tipo de dados.

Definir Público

Use a tag Configurar Público para configurar o ID de público e o nível para um visitante.

Os campos de tag a seguir estão disponíveis para a tag Configurar Público.

Tabela 39. Tags Configurar Público

Campo de Tag	Descrição
*ID de Sessão	Identifica o ID de Sessão.
*ID de Público	Identifica o ID de Público. Os nomes devem corresponder aos nomes de coluna física de qualquer tabela contendo o ID de Público. O ID de público não pode conter mais de 17 dígitos significativos. Se um ID de público tiver mais de 17 dígitos significativos, ele deverá ser particionado ou o ID de público deverá mudar para uma sequência.
*Nível de Público	Define o Nível de Público.
Nome da Função de Retorno de Chamada em Sucesso	Define o nome da função a ser chamada quando o método configurar público é bem-sucedido.
Nome da Função de Retorno de Chamada em Falha	Define o nome da função a ser chamada quando o método configurar público falha.

Qualquer **Campo de Tag** marcado com um * é requerido.

Os parâmetros opcionais podem ser incluídos com o recurso de campo da tag customizada. Nomes de tag de customização consistem do nome do parâmetro, dois pontos e o tipo de dados.

Iniciar Sessão

A tag Iniciar Sessão cria e define uma sessão da web.

Os campos de tag a seguir estão disponíveis para a tag Iniciar Sessão.

Tabela 40. Tags Iniciar Sessão

Campo de Tag	Descrição
*ID de Sessão	Identifica o ID de Sessão.

Tabela 40. Tags Iniciar Sessão (continuação)

Campo de Tag	Descrição
*Canal Interativo	Define o nome do canal interativo ao qual esta sessão se refere. Estes nome deve corresponder ao nome do canal interativo definido na Campanha exatamente.
*ID de Público	Identifica o ID de Público. Os nomes devem corresponder aos nomes de coluna física de qualquer tabela contendo o ID de Público.
*Nível de Público	Define o Nível de Público.
*Contar com a Sessão Existente	Define se esta sessão usa uma sessão nova ou uma sessão existente
*Depurar	Ativa ou desativa as informações de depuração.
Nome da Função de Retorno de Chamada em Sucesso	Define o nome da função a ser chamada quando o método iniciar sessão é bem-sucedido.
Nome da Função de Retorno de Chamada em Falha	Define o nome da função a ser chamada quando o método iniciar sessão falha.

Qualquer **Campo de Tag** marcado com um * é requerido.

Os parâmetros opcionais podem ser incluídos com o recurso de campo da tag customizada. Nomes de tag de customização consistem no nome do parâmetro, dois pontos e o tipo de dados.

A tag Iniciar Sessão deve ser designada a um grupo de páginas cujo contêiner está configurado para `Default`.

Configurações de tag de exemplo

Este exemplo mostra uma configuração simples das configurações de tag Iniciar Sessão, Postar Evento, Obter Ofertas e Terminar Sessão.

Para qualquer tag, é possível obter os valores de campo de tag do cookie com o método de cookie ou do objeto JavaScript com o método javascriptobject.

Estas tags suportam parâmetros adicionais que este exemplo simples não é mostrado. É possível localizar mais informações sobre os parâmetros adicionais no IBM Digital Data Exchange User Guide.

Para obter casos de uso detalhados da integração do Interact e Digital Data Exchange, veja https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Configurações de tag Iniciar Sessão de exemplo

Clique em **Tags** > **Tags IBM** > **IBM Interact** > **Tipo: Iniciar Sessão** para criar uma tag Iniciar Sessão. Edite a tag com as configurações a seguir.

Configurações de ID de sessão

- **Método**: `Constant`
- **Constante**: `5555`
- **Tipo de Dados**: `String`
- **Modificador**: `<null>`

Configurações de Canal Interativo

- **Método**: `Constant`
- **Constante**: `WSCDemo`
- **Tipo de Dados**: `String`
- **Modificador**: `<null>`

Configurações de ID de Público

- **Método**: Constant
- **Constante**: USERS_ID,2002,numeric
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações de Nível de Público

- **Método**: Constant
- **Constante**: WSCUserId
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações Contar com Sessões Existentes

- **Método**: Constant
- **Constante**: False
- **Tipo de Dados**: Boolean
- **Modificador**: <null>

Depuração

- **Método**: Constant
- **Constante**: True
- **Tipo de Dados**: Boolean
- **Modificador**: <null>

Configurações do Nome da Função de Retorno de Chamada em Sucesso

- **Método**: Unassigned
- **Valor**: <null>

Configurações do Nome da Função de Retorno de Chamada em Falha

- **Método**: Unassigned
- **Valor**: <null>

Configurações de tag Obter Ofertas de exemplo

Clique em **Tags** > **Tags IBM** > **IBM Interact** > **Tipo: Obter Ofertas** para criar uma tag Obter Ofertas. Edite a tag com as configurações a seguir.

Configurações de ID de sessão

- **Método**: Constant
- **Constante**: 5555
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações de Nome de Ponto de Interação

- **Método**: Constant
- **Constante**: AuroraHomepageHeaderBannerLeft
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações de Número Solicitado

- **Método**: Constant
- **Constante**: 1
- **Tipo de Dados**: integer
- **Modificador**: <null>

Configurações do Nome da Função de Retorno de Chamada em Sucesso

- **Método**: Constant
- **Constante**: onOfferReturnSuccess
- **Tipo de Dados**: string
- **Modificador**: <null>

Configurações do Nome da Função de Retorno de Chamada em Falha

- **Método**: Constant
- **Constante**: onOfferReturnError
- **Tipo de Dados**: string
- **Modificador**: <null>

Configurações de tag Postar Evento de exemplo

Clique em **Tags** > **Tags IBM** > **IBM Interact** > **Tipo: Postar Evento** para criar uma tag Postar Evento. Edite a tag com as configurações a seguir.

Configurações de ID de sessão

- **Método**: Constant
- **Constante**: 5555
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações de Nome de Evento

- **Método**: Constant
- **Constante**: ACCEPTOFFER
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações do Nome da Função de Retorno de Chamada em Sucesso

- **Método**: Constant
- **Constante**: onSuccessTestFunction
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações do Nome da Função de Retorno de Chamada em Falha

- **Método**: Constant
- **Constante**: onErrorTestFunction
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações de campo de parâmetro adicional

- **Campo de Tag**: UACIOfferTrackingCode:string

- **Método**: JavaScriptObject
- **Nome do Objeto**: oa.treatmentCode
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações de tag Terminar Sessão de exemplo

Clique em **Tags** > **Tags IBM** > **IBM Interact** > **Tipo: Terminar Sessão** para criar uma tag Terminar Sessão. Edite a tag com as configurações a seguir.

Configurações de ID de sessão

- **Método**: Constant
- **Constante**: 5555
- **Tipo de Dados**: String
- **Modificador**: <null>

Configurações do Nome da Função de Retorno de Chamada em Sucesso

- **Método**: Unassigned
- **Valor**: <null>

Configurações do Nome da Função de Retorno de Chamada em Falha

- **Método**: Unassigned
- **Valor**: <null>

Funções de exemplo

Para as funções usadas para as configurações Nome da Função de Retorno de Chamada em Sucesso e Nome da Função de Retorno de Chamada em Falha, apenas é necessário especificar o nome da função ao criar uma nova tag se a função já estiver presente em sua página da web.

Também é possível usar Digital Data Exchange Utilitários para criar funções e incluí-las em suas páginas da web.

O exemplo a seguir mostra como exibir uma oferta retornada do Interact em sua página da web. É necessário incluir este script na página ou usar o fragmento de código do Digital Data Exchange para injetá-lo.

```
<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>
```

Verifique sua configuração de integração

Use a ferramenta de testeDigital Data Exchange e o arquivo `Interact.log` para resolver qualquer problema de configuração.

É possível usar a ferramenta de teste Digital Data Exchange para verificar a enciclopédia para ver se sua configuração funciona conforme esperado. Para abrir a ferramenta de teste, clique em **Implementação** > **Ferramenta de Teste** no Digital Data Exchange.

Veja o IBM Digital Data Exchange User Guide para obter mais informações sobre a ferramenta de teste.

É possível visualizar o arquivo `Interact.log` para ver detalhes sobre as várias chamadas de API Interact que são feitas. Inclua a Função de Retorno de Chamada em Sucesso e a Função de Retorno de Chamada Em Falha em cada tag para depurar as várias chamadas.

Entrando em Contato com o Suporte Técnico do IBM

Se encontrar um problema que não puder resolver consultando a documentação, o contato de suporte designado por sua empresa pode registrar uma chamada com o suporte técnico da IBM . Use as informações nesta seção para assegurar que seu problema seja eficientemente resolvido com êxito.

Se você não for um contato de suporte designado em sua empresa, entre em contato com seu administrador da IBM para obter informações.

Informações a Serem Reunidas

Antes de entrar em contato com o suporte técnico da IBM , reúna as informações a seguir:

- Uma breve descrição da natureza de seu problema.
- Mensagens de erro detalhadas que você vê quando o problema ocorre.
- Etapas detalhadas para reproduzir o problema.
- Arquivos de log relacionados, arquivos de sessão, arquivos de configuração e arquivos de dados.
- As informações sobre seu ambiente de produto e sistema, que podem ser obtidas como descrito em "Informações de sistema".

Informações de Sistema

Ao ligar para o suporte técnico da IBM , pode ser que você seja solicitado a fornecer informações sobre seu ambiente.

Se o seu problema não impedi-lo de efetuar login, a maior parte dessas informações está disponível na página Sobre, que fornece as informações sobre seus aplicativos IBM instalados.

É possível acessar a página Sobre selecionando **Ajuda > Sobre**. Se a página Sobre não estiver acessível, é possível obter o número da versão de qualquer aplicativo IBM visualizando o arquivo `version.txt` localizado sob o diretório de instalação para cada aplicativo.

Informações de Contato para o Suporte Técnico da IBM

Para obter as formas de contato com o suporte técnico da IBM , consulte o website do Suporte Técnica do Produto IBM : (http://www.unica.com/about/product-technical-support.htm).

Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos E.U.A.

O IBM pode não oferecer os produtos, serviços ou recursos discutidos neste documento em outros países. Consulte seu representante IBM local para obter informações sobre os produtos e serviços atualmente disponíveis em sua área. Toda referência a um produto, programa ou serviço IBM não tem a intenção de declarar ou implica em que apenas esse produto, programa ou serviço IBM possa ser usado. Qualquer produto, programa ou serviço funcionalmente equivalente que não infrinja nenhum direito de propriedade intelectual da IBM pode ser usado alternativamente. Todavia, é responsabilidade do usuário avaliar e verificar a operação de qualquer produto, programa ou serviço não IBM.

A IBM pode ter patentes ou solicitações de patentes pendentes que cobrem os assuntos descritos neste documento. O fornecimento deste documento não concede ao Cliente nenhuma licença a essas patentes. Pedidos de licença podem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Para perguntas sobre licença relacionadas a informações de byte duplo (DBCS), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie dúvidas, por escrito ao:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

O parágrafo a seguir não se aplica ao Reino Unido ou qualquer outro país em que tais disposições não estejam de acordo com a legislação local: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA", SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS A ELAS NÃO SE LIMITANDO, AS GARANTIAS IMPLÍCITAS DE NÃO-INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Alguns estados não permitem a exclusão de garantias expressas ou implícitas em determinadas transações, portanto, essa declaração pode não se aplicar ao Cliente.

Estas informações podem incluir imprecisões técnicas ou erros tipográficos. Alterações são periodicamente feitas nas informações aqui existentes e essas alterações serão incorporadas em novas edições da publicação. A IBM pode fazer melhorias e/ou alterações no(s) produto(s) e/ou no(s) programa(s) descrito(s) nesta publicação a qualquer momento sem aviso.

Todas as referências nestas informações a websites sites não IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a estes websites sites. Os materiais contidos nesses websites sites não fazem parte dos materiais para este produto IBM e a utilização desses websites sites é de inteira responsabilidade do Cliente.

A IBM pode usar ou distribuir qualquer das informações fornecidas por você da maneira que achar conveniente, sem que isso implique em qualquer obrigação para com o Cliente.

Os licenciados deste programa que desejam obter informações sobre ele para o propósito de ativação: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) o uso mútuo das informações que foram trocadas, devem entrar em contato com:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Essas informações podem estar disponível, sujeitas aos termos e condições adequados, incluindo em alguns casos, o pagamento de uma tarifa.

O programa licenciado descrito nesta publicação e todo o material licenciado disponível são fornecidos pela IBM sob os termos do Contrato com o Cliente IBM, do Contrato de Licença de Programa Internacional IBM ou de qualquer outro contrato equivalente.

Todos os dados de desempenho aqui contidos foram determinados em um ambiente de controle. Assim, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido tomadas em sistemas em nível de desenvolvimento e não há garantia de que estas medidas serão as mesmas em sistemas disponíveis em geral. Além disso, algumas medidas podem ter sido estimadas por meio de extrapolação. Os resultados reais podem variar. Os usuários deste documento devem verificar os dados aplicáveis para seu ambiente específico.

Informações relativas a produtos não IBM foram obtidas dos fornecedores desses produtos, seus anúncios publicados ou outras fontes disponíveis de publicidade. A IBM não testou esses produtos e não pode confirmar a precisão do desempenho, da compatibilidade ou de outras afirmações relacionadas aos produtos não IBM. Perguntas sobre os recursos de produtos não IBM devem ser endereçadas aos fornecedores desses produtos.

Todas as instruções relativas as direções ou intenções futuras da IBM estão sujeitas a mudanças ou retirada sem aviso prévio, e apenas representam metas e objetivos.

Todos os preços IBM mostrados são preços de varejo sugeridos pela IBM, são atuais e estão sujeitos a alterações sem aviso prévio. Os preços do revendedor podem variar.

Estas informações contêm exemplos de dados e de relatórios usados em operações de negócios diárias. Para ilustrá-las como completamente possíveis, os exemplos incluem os nomes dos indivíduos, das empresas, das marcas e dos produtos. Todos

esses nomes são fictícios e qualquer similaridade com nomes e endereços usados por uma empresa real é mera coincidência.

LICENÇA DE COPYRIGHT:

Estas informações contêm programas aplicativos de amostra em idioma de origem, que ilustra técnicas de programação em várias plataformas operacionais. Você pode copiar, modificar e distribuir esses programas de amostra em qualquer formato sem o pagamento à IBM, para os propósitos de desenvolvimento, uso, marketing ou distribuição de programas aplicativos de acordo com a interface de programação de aplicativos para a plataforma operacional para a qual os programas de amostra foram escritos. Esses exemplos não foram completamente testados sob todas as condições. A IBM, dessa forma, não pode garantir ou indicar a confiabilidade, capacidade de manutenção ou função desses programas. Os programas de amostra são fornecidos "no estado em que se encontra", sem garantia de tipo algum. A IBM não será responsável por quaisquer danos decorrentes do uso pelo Cliente dos programas de amostra.

Se estiver vendo estas informações em cópia eletrônica, as fotografias e ilustrações coloridas podem não aparecer.

Marcas Registradas

IBM, o logotipo da IBM e ibm.com são marcas e/ou marcas registradas da International Business Machines Corp., registradas em muitas jurisdições em todo o mundo. Outros produtos e nomes de serviços podem ser marcas registradas da IBM ou de outras empresas. Uma lista atual de marcas registradas da IBM está disponível na Web em “Copyright and trademark information” em www.ibm.com/legal/copytrade.shtml.

Impresso no Brasil