Versión 9 Release 1.2
Mayo de 2016

IBM Interact Guía del administrador

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información de la sección “Avisos” en la página 357.

Esta edición se aplica a la versión 9, release 1, modificación 2 de IBM Interact y a todos los releases y modificaciones posteriores a menos que se indique lo contrario en nuevas ediciones.

Contenido

Capítulo 1. Administración de IBM Interact

Cuando administra Interact, configura y mantiene los usuarios y los roles, los orígenes de datos y las características opcionales del producto. También supervisa y mantiene los entornos de diseño y ejecución. Hay interfaces de programación de aplicaciones (API) específicas del producto disponibles para su uso.

La administración de Interact consta de varias tareas. Estas tareas son, entre otras:

- Mantenimiento de usuarios y roles
- Mantenimiento de orígenes de datos
- Configuración de las características de presentación de ofertas opcionales de Interact
- Supervisión y mantenimiento del rendimiento de entorno de ejecución

Antes de empezar a administrar Interact, puede familiarizarse con algunos conceptos clave sobre cómo funciona Interact que le facilitarán las tareas. En las siguientes secciones se describen las tareas administrativas asociadas con Interact.

La segunda parte de la guía de administración describe las API disponibles con Interact:

- API de Interact
- API ExternalCallout
- API de aprendizaje

Conceptos clave de Interact

IBM® Interact es un motor interactivo que ofrece objetivos de marketing personalizados para diversas audicencias.

En esta sección se describen algunos de los conceptos clave que debe conocer antes de empezar a trabajar con Interact.

Niveles de audiencia

Un nivel de audiencia es una recopilación de identificadores que pueden ser el objetivo de una campaña. Puede definir los niveles de audiencia según el conjunto correcto de audiencias de su campaña.

Por ejemplo, un conjunto de campañas puede utilizar los niveles de audiencia "Unidad familiar", "Posible cliente", "Cliente" y "Cuenta". Cada uno de estos niveles representa una determinada vista de los datos de marketing disponibles para una campaña.

Los niveles de audiencia normalmente se organizan jerárquicamente. Utilizando los ejemplos anteriores:

- La unidad familiar se encuentra en la parte superior de la jerarquía, y cada unidad familiar puede contener varios clientes, así como uno o varios posibles clientes.
- Cliente es el siguiente nivel en la jerarquía, y cada cliente puede tener varias cuentas.
- Cuenta está en la parte inferior de la jerarquía.

En los entornos interempresariales existen otros ejemplos más complejos de jerarquías de audiencia, donde debe haber niveles de audiencia para las empresas, las compañías, las divisiones, los grupos, las personas, las cuentas, etc.

Estos niveles de audiencia puede tener diferentes relaciones entre ellos, por ejemplo de uno a uno, de varios a uno, o de varios a varios. Al definir los niveles de audiencia, permite que estos conceptos se representen en Campaign, para que los usuarios puedan gestionar las relaciones entre las distintas audiencias para poder definir objetivos. Por ejemplo, aunque puede haber varios posibles clientes por unidad familiar, puede limitar el envío de correos a un posible cliente por unidad familiar.

Entorno de diseño

Utilizar el entorno de diseño para configurar varios componentes de Interact y desplegarlos en el entorno de ejecución.

El entorno de diseño es donde se completa la mayor parte de la configuración de Interact. En el entorno de diseño se definen eventos, puntos de interacción, segmentos inteligentes y reglas de tratamiento. Después de configurar estos componentes, los despliega en el entorno de ejecución.

El entorno de diseño se instala con la aplicación web de Campaign.

Eventos

Un evento es una acción que realiza un visitante y que desencadena una acción en el entorno de ejecución. Ejemplos de un suceso pueden ser: colocar un visitante en un segmento, presentar una oferta o registrar datos.

En primer lugar, los eventos se crean en un canal interactivo y a continuación los desencadena una llamada a la API de Interact utilizando el método `postEvent`. Un evento puede producir una o más de las acciones siguientes definidas en el entorno de diseño de Interact:

- **Desencadenar la resegmentación**. El entorno de ejecución ejecuta de nuevo todos los diagramas de flujo interactivos para el nivel de audiencia actual que está asociado con el canal interactivo, utilizando los datos actuales en la sesión del visitante.

  Cuando diseñe su interacción, a menos que especifique un diagrama de flujo específico, una acción de resegmentación ejecuta de nuevo todos los diagramas de flujo interactivos que están asociados a este canal interactivo con el nivel de audiencia actual, y que cualquier solicitud de ofertas espera hasta que han finalizado todos los diagramas de flujo. Una resegmentación excesiva dentro de una sola visita puede afectar al rendimiento del punto de encuentro de forma visible para el cliente.

  Coloque el cliente en nuevos segmentos después de que se añadan considerables datos nuevos al objeto de sesión de ejecución, como nuevos datos de solicitudes de la API de Interact (por ejemplo, cambiar la audiencia) o acciones de cliente (por ejemplo, añadir nuevos elementos a una lista de deseos o un carro de compra).
- **Registrar contacto de oferta**. El entorno de ejecución indica las ofertas recomendadas para que el servicio de base de datos las registre en el historial de contactos.

  El punto de encuentro debe proporcionar los códigos de tratamiento para las ofertas de las que se van a registrar los contactos. Si es necesario limitar el número de solicitudes entre el punto de contacto y el servidor de ejecución, es

posible registrar el contacto de oferta en la misma llamada en que solicitó las ofertas sin proporcionar ningún código de tratamiento.

Si el punto de encuentro no devuelve los códigos de tratamiento de las ofertas que se Interact ha presentado al visitante, el entorno de ejecución registra la última lista de ofertas recomendadas.

- **Registrar aceptación de oferta**. El entorno de ejecución marca la oferta seleccionada para el servicio de base de datos para registrarla en el historial de respuestas.
- **Registrar rechazo de oferta**. El entorno de ejecución marca la oferta seleccionada para el servicio de base de datos para registrarla en el historial de respuestas.
- **Desencadenar expresión de usuario**. Una *acción de expresión* es una acción que se puede definir utilizando macros de Interact, incluyendo funciones, variables y operadores, incluso EXTERNALCALLOUT. Puede asignar el valor de retorno de la expresión a cualquier atributo del perfil.

  Al pulsar el icono de edición situado junto a Desencadenar expresión de usuario aparecerá el diálogo de edición Expresión de usuario estándar, donde puede especificar e l nivel de audiencia, el nombre de campo opcional al que asignar los resultados y la definición de la propia expresión.
- **Desencadenar sucesos**. Puede utilizar la acción Desencadenar sucesos para un nombre de evento que desee que la acción lance. Si introduce un evento que ya se ha definido, el evento se desencadena cuando se ejecuta la acción en cuestión. Si el nombre de evento que introduce no existe, esta acción genera la creación de dicho evento con la acción especificada.

También puede utilizar eventos para desencadenar acciones definidas por el método `postEvent`, incluyendo registrar datos en una tabla, incluir datos para aprendizaje o desencadenar diagramas de flujo individuales.

Para su comodidad, los eventos se pueden organizar en categorías en el entorno de diseño. Las categorías no tienen ninguna finalidad funcional en el entorno de ejecución.

Canales interactivos

Utilice los canales interactivos en Interact para coordinar todos los objetos, datos y recursos de servidor que están implicados en el marketing interactivo.

Un canal interactivo es una representación en Campaign de un punto de encuentro donde el método de la interfaz es un diálogo interactivo. Esta representación de software se utiliza para coordinar todos los objetos, datos y recursos de servidor implicados en el marketing interactivo.

Un canal interactivo es una herramienta que se utiliza para definir eventos y puntos de interacción. También puede acceder a los informes de un canal interactivo desde la pestaña Análisis de ese canal interactivo.

Los canales interactivos también contienen asignaciones de servidor de preparación y tiempo de ejecución de producción. Puede crear varios canales interactivos para organizar los eventos y puntos de interacción si tiene solo un conjunto de servidores de preparación y tiempo de ejecución de producción, o para dividir los eventos y puntos de interacción por sistema orientado al cliente.

Diagramas de flujo interactivos

Utilización de diagramas de flujo interactivos para dividir sus clientes en segmentos y asignar un perfil a un segmento.

Un diagrama de flujo interactivo está relacionado con un diagrama de flujo por lotes de Campaign, pero no es exactamente igual. Los diagramas de flujo interactivos realizan la misma función principal que los diagramas de flujo por lotes: dividir los clientes en grupos conocidos como segmentos. Sin embargo, para los diagramas de flujo interactivos, los grupos son segmentos inteligentes. Interact utiliza estos diagramas de flujo interactivos para asignar un perfil a un segmento con un evento de comportamiento o del sistema que indica que se requiere volver a segmentar un visitante.

Los diagramas de flujo interactivos contienen un subconjunto de los procesos de los diagramas de flujo por lotes y los procesos específicos de los diagramas de flujo interactivos.

Nota: Los diagramas de flujo interactivos se pueden crear solo en una sesión de Campaign.

Puntos de interacción

Un punto de interacción es un lugar del punto de encuentro donde desea presentar una oferta.

Los puntos de interacción contienen contenido de completado predeterminado en ubicaciones en los que el entorno de ejecución no tiene otro contenido apto para presentar.Los puntos de interacción se pueden organizar en zonas.

Ofertas

Una oferta representa un único mensaje de marketing, que se puede entregar de varias maneras.

En Campaign, puede crear ofertas que se utilicen en una o varias campañas.

Las ofertas se pueden reutilizar:

- En distintas campañas
- En distintos puntos en el tiempo
- Para distintos grupos de personas (celdas)
- Como distintas "versiones" variando los campos parametrizados de la oferta

Puede asignar ofertas a los puntos de interacción en los puntos de encuentro que se presentan a los visitantes.

Perfiles

Un perfil es el conjunto de datos de cliente utilizados por el entorno de ejecución. Estos datos pueden ser un subconjunto de los datos de cliente disponibles en la base de datos de cliente, los datos recopilados en tiempo real o una combinación de ambos.

Los datos del cliente se utilizan con los fines siguientes:

- Para asignar un cliente a uno o más segmentos inteligentes en escenarios de interacción en tiempo real.

Necesita un conjunto de datos de perfil para cada nivel de audiencia por el que desee segmentar. Por ejemplo, si está segmentando por ubicación, podría incluir sólo el código postal del cliente de toda la información de dirección que tenga.

- Para personalizar ofertas
- Como atributos para realizar seguimiento para el aprendizaje

  Por ejemplo, puede configurar Interact para supervisar el estado civil de un visitante y cuántos visitantes de cada estado aceptan una oferta específica. A continuación, el entorno de ejecución puede utilizar esa información para acotar la selección de oferta.

Estos datos son de solo lectura para el entorno de ejecución.

Entorno de ejecución

El entorno de ejecución conecta al punto de encuentro y realiza interacciones. El entorno de ejecución puede consistir en uno o varios servidores de ejecución conectados al punto de encuentro.

El entorno de ejecución utiliza la información desplegada desde el entorno de diseño en combinación con la API de Interact para presentar ofertas para su punto de encuentro.

Sesiones de ejecución

Existe una sesión de ejecución en el servidor de ejecución para cada visitante a su punto de contacto. Esta sesión contiene todos los datos del visitante que utiliza el entorno de ejecución para asignar visitantes a segmentos y recomendar ofertas.

Puede crear una sesión de ejecución al utilizar la llamada `startSession`.

Puntos de encuentro

Un punto de encuentro es una aplicación o ubicación donde puede interactuar con un cliente. Un punto de encuentro puede ser un canal donde el cliente inicia el contacto (una interacción "entrante") o donde se contacta con el cliente (una interacción "saliente").

Ejemplos comunes son los sitios web y las aplicaciones de centro de atención al cliente. Mediante la API de Interact, puede integrar Interact con los puntos de encuentro para presentar ofertas a los clientes en función de la acción que realicen en el punto de encuentro. Los puntos de encuentro también se denominan sistemas orientados al cliente.

Reglas de tratamiento

Las reglas de tratamiento asignan una oferta a un segmento inteligente. Estas asignaciones las restringe adicionalmente la zona definida de forma personalizada que se asocia a la oferta en la regla de tratamiento.

Por ejemplo, puede tener un conjunto de ofertas que asigna a un segmento inteligente en la zona "inicio de sesión", pero un conjunto de ofertas distintas para el mismo segmento en la zona "después de compra". Las reglas de tratamiento se definen en una pestaña de estrategia de interacción de una campaña.

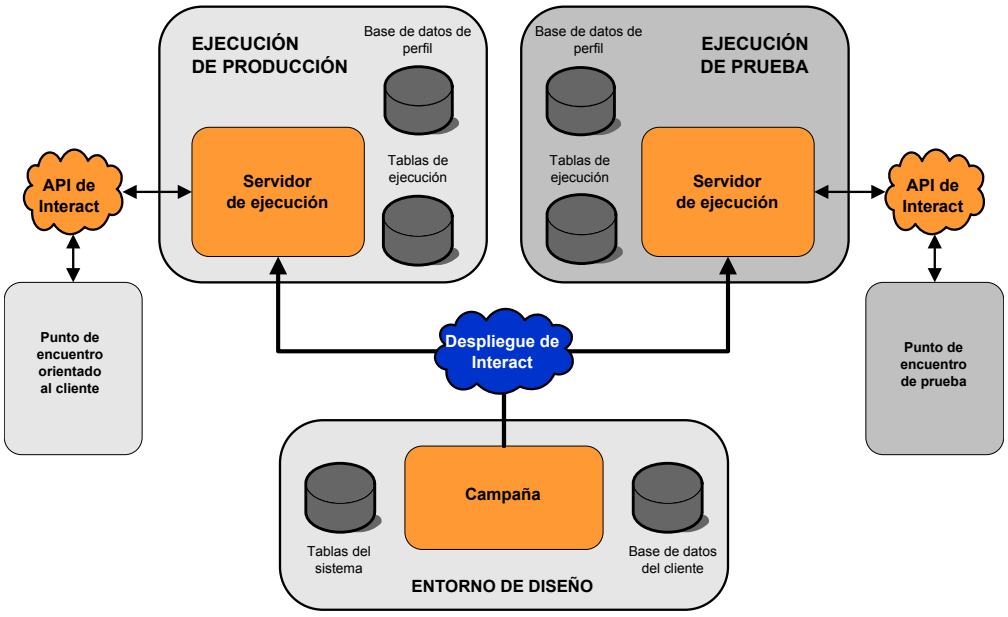
Cada regla de tratamiento también tiene una puntuación de marketing. Si se asigna un cliente a más de un segmento, y por lo tanto más de una oferta es aplicable, las puntuaciones de marketing ayudan a definir qué oferta sugiere Interact. Las ofertas

que sugiere el entorno de ejecución pueden resultar influenciadas por un módulo de aprendizaje, una lista de supresión de ofertas y asignaciones de ofertas globales e individuales.

Arquitectura de Interact

El entorno de Interact está formado por dos componentes principales, como mínimo: el entorno de diseño y el entorno de ejecución. También puede tener servidores de ejecución de pruebas opcionales.

La siguiente figura muestra una descripción general de la arquitectura de alto nivel.



El entorno de diseño es donde se realiza la mayor parte de la configuración de Interact. El entorno de diseño se instala con la aplicación web de Campaign, y hace referencia a las tablas del sistema de Campaign y las bases de datos del cliente. Puede utilizar el entorno de diseño para definir los puntos de interacción y los eventos que utiliza con la API.

Después de diseñar y configurar cómo desea que el entorno de ejecución maneje las interacciones con el cliente, puede desplegar los datos en un grupo de servidores de preparación para realizar pruebas o en un grupo de servidores de ejecución de producción para la interacción con el cliente en tiempo real.

La API de Interact proporciona la conexión entre el punto de encuentro y el servidor de ejecución. Puede hacer referencia a los objetos (los puntos de interacción y los eventos) creados en el entorno de diseño con la API de Interact y utilizarlos para solicitar información al servidor de ejecución.

Consideraciones de red de Interact

Una instalación de producción de Interact abarca al menos dos máquinas. En un entorno de producción de gran volumen, con varios servidores de ejecución de Interact y bases de datos distribuidas, la instalación puede llegar a abarcar docenas de máquinas.

Para garantizar el mejor rendimiento, se deben tener en cuenta varios requisitos de topología de red.

- Si la implementación de la API de Interact inicia y finaliza sesiones en la misma llamada, por ejemplo:

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

  no es necesario que habilite la persistencia de sesiones (sesiones permanentes) entre el equilibrador de carga y los servidores de ejecución de Interact. Puede configurar la gestión de sesiones de los servidores de ejecución de Interact para la memoria caché local.
- Si la implementación de la API de Interact utiliza varias llamadas para iniciar y finalizar sesiones, por ejemplo:

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

  y utiliza un equilibrador de carga para los servidores de ejecución de Interact, debe habilitar algún tipo de persistencia para el equilibrador de carga (también conocidas como sesiones permanentes). Si no es posible, o si no utiliza un equilibrador de carga, configure la gestión de sesiones de los servidores de Interact para un `cacheType` distribuido. Si está utilizando una memoria caché distribuida, todos los servidores de ejecución de Interact deben poder comunicarse mediante multidifusión. Es posible que deba ajustar la red para que la comunicación entre los servidores de Interact que utilizan la misma dirección IP y el mismo puerto no afecte al rendimiento del sistema. Un equilibrador de carga con sesiones permanentes proporciona un mejor rendimiento que el uso de la memoria caché distribuida.
- Si tiene varios grupos de servidores que utilizan un `cacheType` distribuido, cada uno debe utilizar un puerto de multidifusión exclusivo. Se recomienda utilizar un puerto de multidifusión exclusivo y una dirección para cada grupo de servidores.
- Mantenga los servidores de Interact del entorno de ejecución, la Marketing Platform, los equilibradores de carga y el punto de encuentro en la misma ubicación geográfica para garantizar el máximo rendimiento. El tiempo de diseño y el tiempo de ejecución pueden estar en distintas ubicaciones geográficas, pero el despliegue será más lento.
- Utilice una conexión de red rápida (al menos 1 Gb) entre el grupo de servidores de Interact y su punto de encuentro asociado.
- El tiempo de diseño requiere acceso http o https al tiempo de ejecución para completar las tareas de despliegue. Los cortafuegos u otras aplicaciones de red deben estar configurados para permitir el despliegue. Es posible que deba ampliar las longitudes de tiempo de espera HTTP entre el entorno de diseño y los entornos de ejecución si tiene grandes despliegues.
- El módulo de historial de respuestas y contactos requiere acceso a la base de datos de tiempo de diseño (tablas del sistema de Campaign) y acceso a la base de datos de ejecución (las tablas de ejecución de Interact). Debe configurar las bases de datos y la red correctamente para que se produzca esta transferencia de datos.

En una instalación de prueba o preparación, puede instalar el tiempo de diseño y el tiempo de ejecución de Interact en la misma máquina. Este escenario no se recomienda para un entorno de producción.

Inicio de sesión en IBM EMM

Utilice este procedimiento para iniciar la sesión en IBM EMM.

Antes de empezar

Necesita lo siguiente.

- Una conexión de intranet (red) para acceder a su servidor de IBM EMM.
- Un navegador soportado instalado en el sistema.
- Nombre de usuario y contraseña para iniciar la sesión en IBM EMM.
- El URL para acceder a IBM EMM en la red.

El URL es:

`http://`*`host.dominio.com:puerto`*`/unica`

donde

`host` es la máquina donde se ha instalado Marketing Platform.

`dominio.com` es el dominio en el que reside la máquina host

`puerto` es el número de puerto en el que el servidor de aplicaciones de Marketing Platform permanece a la escucha.

Nota: El procedimiento siguiente supone que está iniciando la sesión en una cuenta que tiene acceso Admin a Marketing Platform.

Procedimiento

Acceda al URL de IBM EMM utilizando su navegador.

- Si IBM EMM se ha configurado para integrarse con Windows Active Directory o con la plataforma de control de acceso web e inicia la sesión en dicho sistema, aparece la página predeterminada del panel de control. Su inicio de sesión ha finalizado.
- Si ve la pantalla de inicio de sesión, inicie la sesión utilizando las credenciales del administrador predeterminado. En un entorno de inicio de sesión único, utilice `asm_admin` con `password` como contraseña. En un entorno con varias particiones, utilice `platform_admin` con `password` como contraseña.

  Una solicitud le pedirá que cambie la contraseña. Puede especificar la contraseña existente, pero para garantizar una buena de seguridad, debe elegir una nueva.
- Si IBM EMM se ha configurado para usar SSL, es posible que se le indique aceptar un certificado de seguridad digital, la primera vez que inicie sesión. Pulse **Sí** para aceptar el certificado.

Si el inicio de sesión resulta satisfactorio, IBM EMM muestra la página de panel de control predeterminado.

Resultados

Con los permisos predeterminados asignados a las cuentas de administrador de Marketing Platform, puede administrar las cuentas y seguridad de los usuarios mediante las opciones del menú **Configuración**. Para realizar las tareas de

administración de más alto nivel para los paneles de control de IBM EMM, debe iniciar la sesión como **platform_admin**.

Capítulo 2. Configuración de usuarios de IBM Interact

Interact requiere que configure dos conjuntos de usuarios: los usuarios del entorno de ejecución y los usuarios del entorno de diseño.

- Los **usuarios de tiempo de ejecución** se crean en la Marketing Platform configurada para trabajar con los servidores de ejecución.
- Los **usuarios de tiempo de diseño** son los usuarios de Campaign. Configure la seguridad de los distintos miembros de su equipo de diseño como para Campaign.

Configuración del usuario del entorno de ejecución

Después de instalar Interact, debe configurar al menos un usuario de Interact, el usuario del entorno de ejecución. Los usuarios de tiempo de ejecución se crean en Marketing Platform.

Acerca de esta tarea

El usuario del entorno de ejecución proporciona acceso a las tablas de ejecución. El usuario del entorno de ejecución es el nombre de usuario y la contraseña que utiliza para desplegar los canales interactivos. El servidor de ejecución utiliza la autenticación JDBC del servidor de aplicaciones web para las credenciales de la base de datos. No es necesario añadir orígenes de datos de entorno de ejecución al usuario de entorno de ejecución.

Cuando cree usuarios de ejecución:

- Si tiene instancias de Marketing Platform diferentes para cada servidor de ejecución, debe crear el mismo usuario y la misma contraseña en cada una de ellas. Todos los servidores de ejecución que pertenecen al mismo grupo de servidores deben compartir las credenciales de usuario.
- Si utiliza un programa de utilidad de carga de base de datos, debe definir las tablas de ejecución como un origen de datos con credenciales de inicio de sesión para el entorno de ejecución en las propiedades de configuración bajo `Interact > general > systemTablesDataSource`.
- Si habilita la seguridad para la supervisión JMX con el protocolo JMXMP, necesitará un usuario aparte para la seguridad de supervisión JMX.

Consulte la documentación de Marketing Platform para ver los pasos necesarios para crear el usuario de ejecución.

Configuración de usuarios del entorno de diseño

Los usuarios del entorno de diseño son usuarios de Campaign. Puede configurar los usuarios del entorno de diseño de la forma en que configura los permisos de rol de Campaign.

Acerca de esta tarea

Algunos usuarios del entorno de diseño también requieren algunos permisos de Campaign como Macros personalizadas.

Cuando crea usuarios del entorno de diseño:

- Si tiene usuarios de Campaign con permiso para editar los diagramas de flujo interactivos, otórgueles acceso al origen de datos de tablas de ejecución de prueba.
- Si ha instalado y configurado Interact, las siguientes opciones adicionales están disponibles para la Política global predeterminada y las nuevas políticas.
-

Categoría	**Permisos**
Campañas	• Ver estrategias de interacción de campaña: posibilidad de ver pero no editar pestañas de estrategia de interacción en una campaña. • Editar estrategias de interacción de campaña: capacidad de realizar cambios en las pestañas de estrategia de interacción, incluidas las reglas de tratamiento. • Suprimir estrategias de interacción de campañas: posibilidad de eliminar pestañas de estrategia de interacción de las campañas. La supresión de una pestaña de estrategia de interacción está restringida si la estrategia de interacción se ha incluido en un despliegue de canal interactivo. • Añadir estrategias de interacción de campaña: posibilidad de crear nuevas pestañas de estrategia de interacción en una campaña. • Iniciar despliegues de estrategias de interacción de campaña: posibilidad de marcar una pestaña de estrategia de interacción para su despliegue o para anular el despliegue.
Canales interactivos	• Desplegar canales interactivos: capacidad de desplegar un canal interactivo en los entornos de ejecución de Interact. • Editar canales interactivos: capacidad de realizar cambios en la pestaña de resumen de canales interactivos. • Suprimir canales interactivos: posibilidad de eliminar canales interactivos. La supresión de canales interactivos está restringida si el canal interactivo se ha desplegado. • Ver canales interactivos: posibilidad de ver pero no editar los canales interactivos. • Añadir canales interactivos: posibilidad de crear nuevos canales interactivos. • Ver informes de canal interactivo: posibilidad de ver la pestaña Análisis del canal interactivo. • Añadir objetos hijo a canales interactivos: posibilidad de añadir puntos de interacción, zonas, eventos y categorías.

Categoría	Permisos
Sesiones	• Ver diagramas de flujo interactivos: posibilidad de ver un diagrama de flujo interactivo en una sesión. • Añadir diagramas de flujo interactivos: posibilidad de crear nuevos diagramas de flujo interactivos en una sesión. • Editar diagramas de flujo interactivos: capacidad de realizar cambios en diagramas de flujo interactivos. • Suprimir diagramas de flujo interactivos: posibilidad de eliminar diagramas de flujo interactivos. La supresión de diagramas de flujo interactivos está restringida si el canal interactivo al que está asignado este diagrama de flujo interactivo se ha desplegado. • Copiar diagramas de flujo interactivos: posibilidad de copiar diagramas de flujo interactivos. • Ejecución de pruebas en diagramas de flujo interactivos: posibilidad de iniciar una ejecución de prueba en un diagrama de flujo interactivo. • Revisar diagramas de flujo interactivos: posibilidad de ver un diagrama de flujo interactivo y abrir procesos para ver la configuración, pero sin poder hacer cambios. • Desplegar diagramas de flujo interactivos: posibilidad de marcar un diagrama de flujo interactivo para su despliegue o para anular el despliegue.

Permisos del entorno de diseño de ejemplo

En este ejemplo se muestran los permisos que se otorgan a dos roles diferentes: uno para los usuarios que crean diagramas de flujo interactivos y otro para los usuarios que definen estrategias de interacción.

Rol de diagrama de flujo interactivo

Esta tabla muestra los permisos que se otorgan al rol de diagrama de flujo interactivo:

Categoría	Permiso
Macro personalizada	El rol de usuario tiene estos permisos: • Añadir macros personalizadas • Editar macros personalizadas • Utilizar macros personalizadas
Campo derivado	El rol de usuario tiene estos permisos: • Añadir campos derivados • Editar campos derivados • Utilizar campos derivados
Plantilla de diagrama de flujo	El rol de usuario tiene estos permisos: • Pegar plantillas
Plantilla de segmento	El rol de usuario tiene estos permisos: • Añadir segmentos • Editar segmentos

Categoría	Permiso
Sesión	El rol de usuario tiene estos permisos: • Ver resumen de sesión • Ver diagramas de flujo interactivos • Añadir diagramas de flujo interactivos • Editar diagramas de flujo interactivos • Copiar diagramas de flujo interactivos • Realizar ejecución de prueba de diagramas de flujo interactivos • Desplegar diagramas de flujo interactivos

Rol de estrategia de interacción

Esta tabla muestra los permisos que se otorgan al rol de estrategia de interacción:

Categoría	Permiso
Campaña	El rol de usuario tiene estos permisos: • Ver resumen de campaña • Gestionar celdas objetivo de campaña • Ver estrategias de interacción de campaña • Editar estrategias de interacción de campaña • Añadir estrategias de interacción de campaña • Iniciar despliegues de estrategia de interacción de campaña
Oferta	El rol de usuario tiene estos permisos: • Ver resumen de oferta
Plantilla de segmento	El rol de usuario tiene estos permisos: • Ver resumen de segmento
Sesión	El rol de usuario tiene estos permisos: • Revisar diagramas de flujo interactivos

Capítulo 3. Gestión de orígenes de datos de Interact

Interact requiere varios orígenes de datos para funcionar correctamente. Algunos orígenes de datos contienen la información que Interact necesita para funcionar, mientras que otros orígenes de datos contienen los datos.

En las siguientes secciones se describen los orígenes de datos de Interact, incluida la información que necesita para configurarlos correctamente y algunas sugerencias para su mantenimiento.

Orígenes de datos de Interact

Interact requiere varios conjuntos de datos para funcionar. Los conjuntos de datos se almacenan y se recuperan de orígenes de datos y los orígenes de datos que configure dependerán de las características de Interact que esté habilitando.

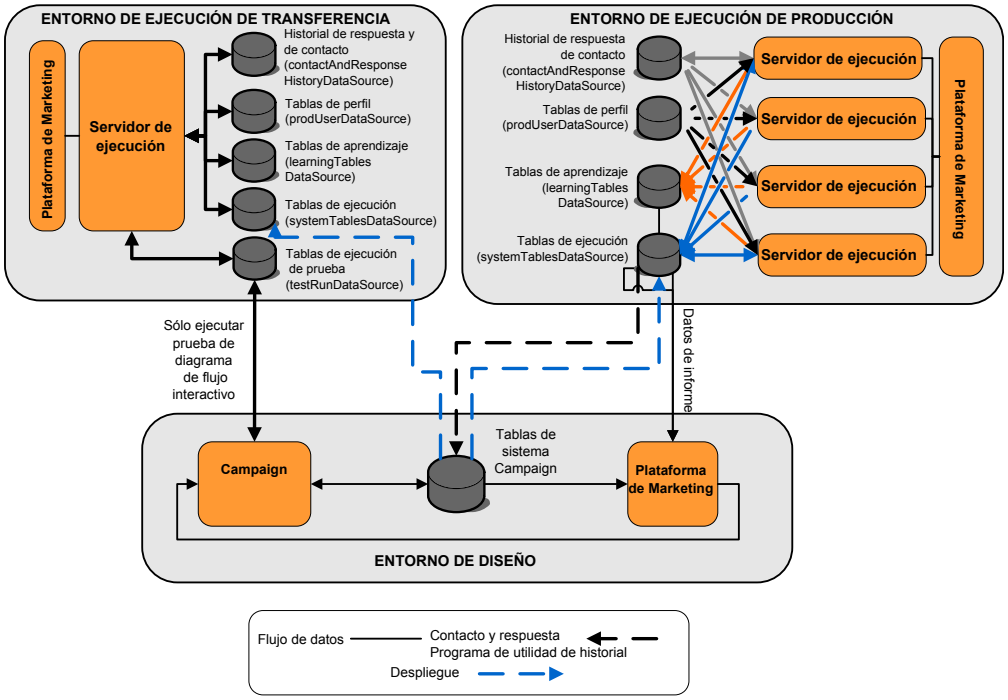
- **Tablas del sistema de Campaign**. Aparte de todos los datos de Campaign, las tablas del sistema de Campaign contienen datos para los componentes de Interact que crea en el entorno de diseño como, por ejemplo, las reglas de tratamiento y los canales interactivos. El entorno de diseño y las tablas del sistema de Campaign comparten el mismo esquema y la misma base de datos física.
- **Tablas de ejecución**(`systemTablesDataSource`). Este origen de datos contiene los datos de despliegue del entorno de diseño, las tablas de preparación del historial de contactos y respuestas, y las estadísticas de tiempo de ejecución.
- **Tablas de perfil** (`prodUserDataSource`). Este origen de datos contiene los datos de cliente, además de la información que se recopila en tiempo real, necesaria para que los diagramas de flujo interactivos coloquen correctamente a los visitantes en segmentos inteligentes. Si se basa solamente en los datos en tiempo real, no necesita las tablas de perfil. Si utiliza tablas de perfil, debe tener al menos una tabla de perfil por cada nivel de audiencia que utilice el canal interactivo.

  Las tablas de perfil también pueden contener las tablas utilizadas para aumentar la presentación de ofertas, incluidas las tablas de supresión de oferta, anulación de puntuaciones y asignación de ofertas globales e individuales.
- **Tablas de ejecución** (`testRunDataSource`). Este origen de datos contiene una muestra de todos los datos que necesitan los diagramas de flujo interactivos para colocar a los visitantes en segmentos inteligentes, incluidos los datos que imitan qué se recopila en tiempo real durante una interacción. Estas tablas sólo son necesarias para el grupo de servidores designado como grupo de servidores de ejecución de prueba para el entorno de diseño.
- **Tablas de aprendizaje** (`learningTablesDataSource`). Este origen de datos contiene todos los datos que recopila la utilidad de aprendizaje incorporado. Estas tablas pueden incluir una tabla que define atributos dinámicos. Si no utiliza el aprendizaje o utiliza una utilidad de aprendizaje externa que ha creado, no necesita las tablas de aprendizaje.
- **Historial de contactos y respuestas para respuesta de sesiones cruzadas** (`contactAndResponseHistoryDataSource`). Este origen de datos contiene las tablas de historial de contactos de Campaign o una copia de ellas. Si no utiliza la característica de respuesta de sesiones cruzadas, no es necesario configurar estas tablas del historial de contactos.

Bases de datos y las aplicaciones

Es posible que los orígenes de datos que cree para que los utilice Interact también se utilicen para intercambiar o compartir datos con otras aplicaciones de IBM EMM.

El siguiente diagrama muestra los posibles orígenes de datos de Interact y cómo se relacionan con las aplicaciones de IBM EMM.



- El grupo de servidores de ejecución de prueba y Campaign acceden a las tablas de ejecución de prueba.
- Las tablas de ejecución de prueba sólo se utilizan para las ejecuciones de prueba de diagrama de flujo interactivo.
- Cuando utiliza un servidor de ejecución para probar un despliegue, incluida la API de Interact, el servidor de ejecución utiliza las tablas de perfiles para los datos.
- Si configura el módulo de historial de contactos y respuestas, el módulo utiliza un proceso ETL (Extracción, Transformación y Carga) de fondo para mover los datos de las tablas de preparación de ejecución a las tablas de historial de contactos y respuestas de Campaign.
- La función de creación de informes consulta los datos de las tablas de aprendizaje, las tablas de ejecución y las tablas del sistema de Campaign para visualizar informes en Campaign.

Debe configurar los entornos de ejecución de prueba para que utilicen un conjunto de tablas diferente al de los entornos de ejecución de producción. Al utilizar tablas diferentes para la preparación y la producción, puede mantener separados los resultados de las pruebas y los resultados reales. Tenga en cuenta que el módulo de historial de contactos y respuestas siempre inserta datos en las tablas de historial de contactos y respuestas de Campaign (Campaign no tiene tablas de historial de contactos y respuestas de prueba). Si utiliza tablas de aprendizaje aparte para el entorno de ejecución de prueba y desea ver los resultados en los

informes, necesitará una instancia independiente de IBM Cognos BI para ejecutar los informes de aprendizaje para el entorno de prueba.

Tablas del sistema de Campaign

Al instalar el entorno de diseño de Interact, también crea nuevas tablas específicas de Interact en las tablas del sistema de Campaign. Las tablas que cree dependen de las características de Interact que esté habilitando.

Si habilita el módulo de historial de contactos y respuestas, el módulo copia el historial de contactos y respuestas de las tablas de preparación de las tablas de ejecución en las tablas del sistema de Campaign. Las tablas predeterminadas son `UA_ContactHistory`, `UA_DtlContactHist` y `UA_ResponseHistory`, pero el módulo del historial de contactos y respuestas utiliza las tablas que estén correlacionadas en Campaign para las tablas de historial de contactos y respuestas.

Si utiliza las tablas de ofertas globales y las tablas de anulación de puntuaciones para asignar ofertas, es posible que necesite completar la tabla `UACI_ICBatchOffers` de las tablas del sistema de Campaign si utiliza ofertas que no están incluidas en las reglas de tratamiento para el canal interactivo.

Tablas de ejecución

Si tiene más de un nivel de audiencia, debe crear tablas de preparación para los datos del historial de contactos y respuestas para cada nivel de audiencia.

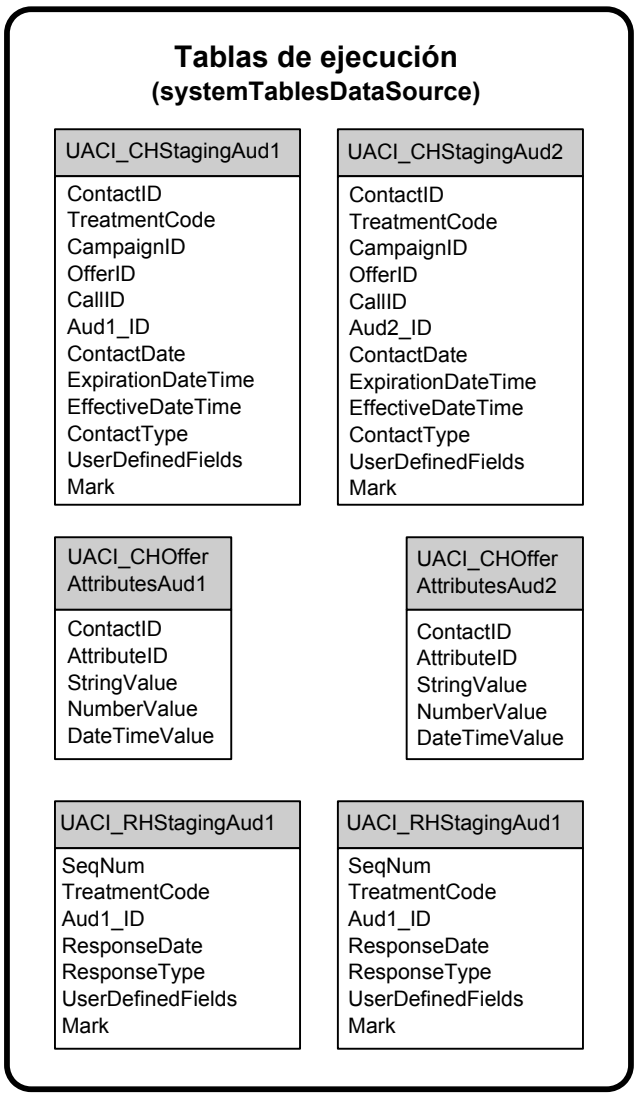
Los scripts SQL crean las siguientes tablas para el nivel de audiencia predeterminado:

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

Debe crear copias de estas tres tablas para cada uno de los niveles de audiencia en las tablas de ejecución.

Si las tablas del historial de contactos y respuestas de Campaign tienen campos definidos por el usuario, debe crear los mismos nombres y tipos de campo en las tablas `UACI_CHStaging` y `UACI_RHStaging`. Puede completar estos campos durante el tiempo de ejecución mediante la creación de pares nombre-valor del mismo nombre en los datos de sesión. Por ejemplo, las tablas del historial de contactos y respuestas contienen el campo `catalogID`. Debe añadir el campo `catalogID` a las tablas `UACI_CHStaging` y `UACI_RHStaging`. Posteriormente, la API de Interact completa este campo mediante la definición de un parámetro de evento como un par nombre-valor denominado `catalogID`. Los datos de sesión pueden proporcionarlos la tabla de perfil, los datos temporales, el aprendizaje o la API de Interact.

En el siguiente diagrama se incluyen las tablas de muestra para las audiencias Aud1 y Aud2. Este diagrama no incluye todas las tablas de la base de datos de tiempo de ejecución.



Todos los campos de las tablas son necesarios. Puede modificar `CustomerID` y `UserDefinedFields` para que coincidan con sus tablas del historial de contactos y respuestas de Campaign.

Tablas de ejecución de pruebas

Las tablas de ejecución de prueba sólo se utilizan para las ejecuciones de prueba de diagramas de flujo interactivos. Las ejecuciones de prueba de diagramas de flujo interactivos deben probar su lógica de segmentación. Sólo necesita configurar una base de datos de ejecución de prueba para la instalación de Interact. Las tablas de ejecución de prueba no necesitan estar en una base de datos autónoma. Por ejemplo, puede utilizar tablas de datos de cliente para Campaign.

El usuario de base de datos asociado con las tablas de ejecución de prueba debe tener privilegios CREATE para añadir las tablas de resultados de ejecución de prueba.

La base de datos de ejecución de prueba debe contener todas las tablas correlacionadas en el canal interactivo.

Estas tablas deben contener datos para ejecutar los escenarios que desea probar en los diagramas de flujo interactivos. Por ejemplo, si los diagramas de flujo interactivos tienen lógica para clasificar a las personas en segmentos basándose en la opción seleccionada en un sistema de correo de voz, debe tener al menos una fila para cada selección posible. Si está creando una interacción que funciona con un formulario en el sitio web, debe incluir filas que representen los datos que faltan o están dañados, por ejemplo, utilice `name@domaincom` para el valor de una dirección de correo electrónico.

Cada tabla de ejecución de prueba debe contener como mínimo una lista de ID para el nivel de audiencia adecuado, y una columna que represente los datos en tiempo real que espera utilizar. Como las ejecuciones de prueba no tienen acceso a los datos en tiempo real, debe proporcionar datos de muestra para cada dato en tiempo real esperado. Por ejemplo, si desea utilizar datos recopilados en tiempo real como el nombre de la última página web visitada, que se almacena en el atributo `lastPageVisited`, o el número de artículos de un carro de la compra, que se almacena en el atributo `shoppingCartItemCount`, debe crear columnas con los mismos nombres y completarlas con datos de muestra. Esto permite ejecutar en prueba las ramas de la lógica del diagrama de flujo que tienen una naturaleza de comportamiento o contextual.

Las ejecuciones de prueba de los diagramas de flujo interactivos no están optimizadas para trabajar con grandes conjuntos de datos. Puede limitar el número de filas que se utilizan para la ejecución de prueba en el proceso Interacción. No obstante, siempre se seleccionará el primero conjunto de filas. Para garantizar que se seleccionen otros conjuntos de filas, utilice distintas vistas de las tablas de ejecución de prueba.

Para probar el rendimiento de los diagramas de flujo interactivos en el tiempo de ejecución, debe crear un entorno de ejecución de prueba, incluida una tabla de perfil para el entorno de prueba.

En la práctica, necesitará tres conjuntos de tablas para la prueba: una tabla de ejecución de prueba para las ejecuciones de prueba de los diagramas de flujo interactivos, tablas de perfiles de prueba para el grupo de servidores de pruebas, y un conjunto de tablas de perfiles de producción.

Sustitución de los tipos de datos predeterminados para tablas creadas dinámicamente

El entorno de ejecución de Interact crea dinámicamente tablas en dos escenarios: durante una ejecución de prueba de un diagrama de flujo y durante la ejecución de un proceso Instantánea que graba en una tabla que no existe ya. Para crear estas tablas, Interact se basa en tipos de datos codificados para cada tipo de base de datos soportada.

Puede sustituir los tipos de datos predeterminados creando una tabla de tipos de datos alternativos, denominada `uaci_column_types`, en `testRunDataSource` o `prodUserDataSource.` Esta tabla adicional permite a Interact contener casos no habituales que no se incluyen en los tipos de datos codificados.

Cuando la tabla `uaci_column_types` está definida, Interact utiliza los metadatos para las columnas como los tipos de datos que se utilizarán para cualquier generación de tabla. Si no se define la tabla `uaci_column_types`, o si se han encontrado excepciones al intentar leer la tabla, se utilizarán los tipos de datos predeterminados.

Durante el inicio, en primer lugar el sistema de ejecución comprueba el testRunDataSource para la tabla uaci_column_types. Si la tabla uaci_column_types no existe en testRunDataSource, o si el prodUserDataSource es de un tipo de base de datos distinto, a continuación Interact comprueba el prodUserDataSource para la tabla.

Sustitución de los tipos de datos predeterminados

Utilice este procedimiento para sustituir los tipos de datos predeterminados para las tablas creadas dinámicamente.

Acerca de esta tarea

Debe reiniciar el servidor de ejecución siempre que cambie la tabla uaci_column_types. Planifique los cambios para que el reinicio del servidor tenga un efecto mínimo en las operaciones.

Procedimiento

1. Cree una tabla en TestRunDataSource o ProdUserDataSource con las propiedades siguientes:

   Nombre de tabla: uaci_column_types

   Nombres de columna:

   - uaci_float
   - uaci_number
   - uaci_datetime
   - uaci_string

   Utilice el tipo de datos adecuado al que da soporte la base de datos para definir cada columna.
2. Reinicie el servidor de ejecución para permitir que Interact reconozca la nueva tabla.

Tipos de datos predeterminados para tablas creadas dinámicamente

Para cada base de datos soportada que utiliza el sistema del tiempo de ejecución de Interact, hay tipos de datos codificados que se utilizan de forma predeterminada para columnas de valores flotantes, numéricos, de fecha/hora y de cadena.

Tabla 1. Tipos de datos predeterminados para tablas creadas dinámicamente

Base de datos	Tipos de datos predeterminados
DB2	• float • bigint • timestamp • varchar
Informix	• float • int8 • DATETIME YEAR TO FRACTION • char2

Tabla 1. Tipos de datos predeterminados para tablas creadas dinámicamente (continuación)

Base de datos	Tipos de datos predeterminados
Oracle	• `float` • `number(19)` • `timestamp` • `varchar2`
SQL Server	• `float` • `bigint` • `datetime` • `nvarchar`

Base de datos de perfil

El contenido de la base de datos de perfil depende totalmente de los datos que necesita para configurar los diagramas de flujo interactivos y la API de Interact. Interact requiere o recomienda que cada base de datos contenga determinadas tablas o datos.

La base de datos de perfil debe contener lo siguiente:

- Todas las tablas correlacionadas en el canal interactivo.

  Estas tablas deben contener todos los datos necesarios para ejecutar los diagramas de flujo interactivos en producción. Estas tablas deben acoplarse, dinamizarse e indexarse correctamente. Como existe un coste de rendimiento al acceder a los datos dimensionales, debe utilizar un esquema desnormalizado siempre que sea posible. Como mínimo, debe indexar la tabla de perfil en los campos ID de nivel de audiencia. Si se recuperan otros campos de las tablas dimensionales, deben indexarse correctamente para reducir el tiempo de captación de base de datos. Los ID de audiencia de las tablas de perfiles deben coincidir con los ID de audiencia definidos en Campaign.
- Si establece la propiedad de configuración `enableScoreOverrideLookup` en true, debe incluir una tabla de anulación de puntuaciones para al menos un nivel de audiencia. Los nombres de tabla de anulación de puntuaciones se definen con la propiedad `scoreOverrideTable`.

  La tabla de anulación de puntuaciones puede contener pares individuales de cliente y oferta. Puede crear una tabla de anulación de puntuaciones de muestra, `UACI_ScoreOverride`, ejecutando el script SQL `aci_usertab` en la base de datos de perfil. También debe indexar esta tabla en la columna ID de audiencia.

  Si establece la propiedad `enableScoreOverrideLookup` en false, no necesita incluir una tabla de anulación de puntuaciones.
- Si establece la propiedad de configuración `enableDefaultOfferLookup` en true, debe incluir la tabla de ofertas globales (`UACI_DefaultOffers`). Puede crear la tabla de ofertas globales ejecutando el script SQL `aci_usertab` en la base de datos de perfil.

  La tabla de ofertas globales puede contener pares de audiencia y oferta.
- Si establece la propiedad `enableOfferSuppressionLookup` en true, debe incluir una tabla de supresión de ofertas para al menos un nivel de audiencia. Los nombres de tabla de supresión de ofertas se definen con la propiedad `offerSuppressionTable`.

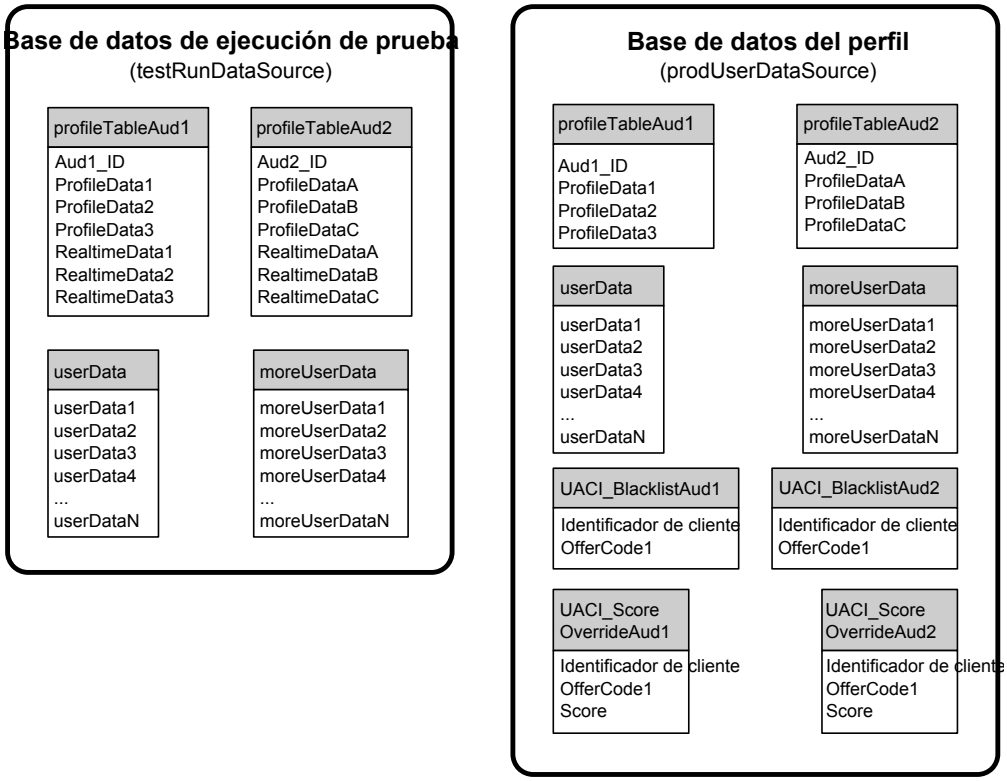
La tabla de supresión de ofertas puede contener una fila para cada oferta suprimida para un miembro de la audiencia, aunque no se necesita una entrada para todos los miembros de audiencia. Puede crear una tabla de supresión de ofertas de muestra, `UACI_BlackList`, ejecutando el script SQL `aci_usertab` en la base de datos de perfil.

Si establece la propiedad `enableOfferSuppressionLookup` en false, no necesita incluir una tabla de supresión de ofertas.

Una gran cantidad de datos en cualquiera de estas tablas puede disminuir el rendimiento. Para obtener los mejores resultados, coloque los índices adecuados en las columnas de nivel de audiencia de las tablas utilizadas en el tiempo de ejecución que tengan grandes cantidades de datos.

Todas las propiedades de configuración referencias anteriormente están en la categoría **Interact > perfil** o **Interact > perfil > Niveles de audiencia > *AudienceLevel***. El script SQL `aci_usertab` se encuentra en el directorio `ddl` del directorio de instalación del entorno de ejecución.

El siguiente diagrama incluye las tablas de muestra de las bases de datos de perfil y ejecución de prueba para los niveles de audiencia Aud1 y Aud2.



Tablas de aprendizaje

Si utiliza el aprendizaje incorporado de Interact, debe configurar las tablas de aprendizaje. Estas tablas contienen todos los datos de los que aprende la característica de aprendizaje incorporado.

Si utiliza atributos de aprendizaje dinámicos, debe completar la tabla `UACI_AttributeList`.

El aprendizaje implica escribir en las tablas de preparación intermedias y agregar información de las tablas de preparación en las tablas de aprendizaje. Las propiedades de configuración `insertRawStatsIntervalInMinutes` y `aggregateStatsIntervalInMinutes` en la categoría `Interact > offerserving > Configuración de aprendizaje incorporado` determinan con qué frecuencia se completan las tablas de aprendizaje.

El atributo `insertRawStatsIntervalInMinutes` determina con qué frecuencia se mueve la información de contacto y aceptación de cada cliente desde la memoria a las tablas de preparación, `UACI_OfferStatsTX` y `UACI_OfferTxAll`. La información almacenada en las tablas de preparación se agrega y traslada a las tablas `UACI_OfferStats` y `UACI_OfferStatsAll` a intervalos regulares determinados por la propiedad de configuración `aggregateStatsIntervalInMinutes`.

El aprendizaje incorporado de Interact utiliza estos datos para calcular las puntuaciones finales de las ofertas.

Historial de contactos para el seguimiento de respuestas de sesiones cruzadas

Si habilita la característica de respuestas de sesiones cruzadas, el entorno de ejecución necesita acceso de sólo lectura a las tablas del historial de contactos de Campaign. Puede configurar el entorno de ejecución para ver las tablas del sistema de Campaign, o puede crear una copia de las tablas del historial de contactos de Campaign. Si crea una copia de las tablas, debe gestionar el proceso de mantener la copia actualizada. El módulo del historial de contactos y respuestas no actualizará la copia de las tablas del historial de contactos.

Debe ejecutar el script SQL `aci_crhtab` en estas tablas de historial de contactos para añadir las tablas necesarias para la característica de seguimiento de respuestas de sesiones cruzadas.

Ejecución de scripts de base de datos para habilitar características de Interact

Para utilizar las características opcionales disponibles en Interact, ejecute los scripts de base de datos en la base de datos para crear tablas o actualizar las tablas existentes.

La instalación de Interact, tanto el entorno de diseño como el entorno de ejecución, incluye los scripts **`ddl`** de las características. Los scripts **`ddl`** añaden las columnas necesarias a las tablas.

Para habilitar cualquiera de las características opcionales, ejecute el script correspondiente en la base de datos o tabla que se indica.

`tipoBD` es el tipo de base de datos, por ejemplo `sqlsvr` para Microsoft SQL Server, `ora` para Oracle, o `db2` para IBM DB2.

Utilice la tabla siguiente para ejecutar los scripts de base de datos en la base de datos para crear tablas o actualizar tablas existentes:

Tabla 2. Scripts de base de datos

Nombre de característica	Script de características	Ejecutar en	Cambio
Ofertas globales, supresión de oferta y anulación de puntuaciones	**aci_usrtab_***tipoBD***.sql** en *Inicio_Interact*\ddl\ acifeatures\ (directorio de instalación del entorno de ejecución)	La base de datos de perfil (userProdDataSource)	Crea las tablas UACI_DefaultOffers, UACI_BlackList y UACI_ScoreOverride.
Puntuación	**aci_scoringfeature_***tipoBD***.sql** en *Inicio_Interact*\ddl\ acifeatures\ (directorio de instalación del entorno de ejecución)	Tablas de anulación de puntuaciones en la base de datos de perfil (userProdDataSource)	Añade las columnas LikelihoodScore y AdjExploreScore.
Aprendizaje	**aci_lrnfeature_***tipoBD***.sql** en *Inicio_Interact*\interactDT\ ddl\acifeatures\ (directorio de instalación del entorno de diseño)	Base de datos de Campaign que contiene sus tablas de historial de contactos	Añade las columnas RTSelectionMethod, RTLearningMode y RTLearningModelID a la tabla UA_DtlContactHist. Añade también las columnas RTLearningMode y RTLearningModelID a la tabla UA_ResponseHistory. Este script también lo requieren las características de creación de informes que proporciona el paquete de informes opcional de Interact.

Acerca del seguimiento del historial de contactos y respuestas

Puede configurar el entorno de ejecución para registrar el historial de contactos y respuestas en las tablas de historial de contactos y respuestas de Campaign. Los servidores de ejecución almacenan el historial de contactos y respuestas en las tablas de preparación. El módulo de historial de respuestas y contactos copia estos datos desde las tablas de preparación en las tablas de historial de contactos y respuestas de Campaign.

El módulo del historial de contactos y respuestas sólo funciona si establece las propiedades `interactInstalled` y `contactAndResponseHistTracking > isEnabled` en la página de configuración del entorno de diseño en `yes`.

Si utiliza el módulo de seguimiento de respuestas de sesiones cruzadas, el módulo del historial de contactos y respuestas es una entidad aparte.

Tipos de contactos y respuestas

Puede registrar un tipo de contacto y dos tipos de respuesta en Interact. Puede registrar más tipos de respuesta personalizados con el método `postEvent`.

Propiedades de la tabla contactAndResponseHistTracking

Esta tabla lista las propiedades que se encuentran en la categoría `contactAndResponseHistTracking`:

Evento	Tipo de contacto/respuesta	Propiedad de configuración
Registrar contacto de oferta	Contacto	contactado
Registrar aceptación de oferta	Respuesta	aceptar
Registrar rechazo de oferta	Respuesta	rechazar

Propiedades de la tabla UA_UsrResponseType

Asegúrese de que la columna CountsAsResponse de la tabla UA_UsrResponseType de las tablas del sistema de Campaign esté configurada correctamente. Todos estos tipos de respuesta deben existir en la tabla UA_UsrResponseType.

Para que sea una entrada válida en UA_UsrResponseType, debe definir un valor para todas las columnas de la tabla, incluida CountsAsResponse. Los valores válidos para CountsAsResponse son:

- 0: ninguna respuesta
- 1: una respuesta
- 2: un rechazo
-

Estas respuestas se utilizan para la creación de informes.

Tipos de respuesta adicionales

En Interact, puede utilizar el método postEvent en la API de Interact para desencadenar un evento que registra una acción "aceptar" o "rechazar" para una oferta. También puede aumentar el sistema para que la llamada postEvent pueda registrar tipos de respuesta adicionales como, por ejemplo, Explorar, Considerar, Confirmar o Cumplir.

Todos estos tipos de respuesta deben existir en la tabla UA_UsrResponseType en las tablas del sistema de Campaign. Utilizando parámetros de evento específicos en el método postEvent, puede registrar tipos de respuesta adicionales y definir si debe incluirse una aceptación en el aprendizaje.

Para registrar tipos de respuesta adicionales, debe añadir los siguientes parámetros de evento:

- **UACIResponseTypeCode**: una cadena que representa un código de tipo de respuesta. El valor debe ser una entrada válida en la tabla UA_UsrResponseType.

  Para que sea una entrada válida en UA_UsrResponseType, debe definir todas las columnas de la tabla, incluida CountsAsResponse. Los valores válidos para CountsAsResponse son 0, 1, o 2. 0 indica que no hay respuesta, 1 indica una respuesta y 2 indica un rechazo. Estas respuestas se utilizan para la creación de informes.
- **UACILogToLearning**: un número con el valor 1 o 0. 1 indica que Interact deberá registrar el evento como aceptado en el sistema de aprendizaje o habilitar la supresión de ofertas dentro de una sesión. 0 indica que Interact no deberá registrar el evento en el sistema de aprendizaje o habilitar la supresión de ofertas dentro de una sesión. Este parámetro le permite crear varios métodos postEvent registrando distintos tipos de respuesta sin influenciar el aprendizaje. Si no define UACILogToLearning, Interact utiliza el valor predeterminado 0.

Si se proporciona un responseTypeCode cuando se publica un evento de aceptación, la oferta no se ha suprimido al aceptarlo. Independientemente del valor de ResponseTypeCode (por ejemplo, 0, 1, 2), si logToLearningAsAccept es 0, la oferta nunca debe suprimirse. Para suprimir la oferta, postEvent no debe especificar el parámetro UACIResponseTypeCode. Si se proporciona el parámetro UACIResponseTypeCode, el valor de UACILogToLearning debe ser 1 si desea que se suprima la oferta.

Si lo desea, puede crear varios eventos con la acción Registrar aceptación de oferta, uno para cada tipo de respuesta que desee registrar, o un solo evento con la acción Registrar aceptación de oferta que utiliza para cada llamada postEvent utilizada para registrar tipos de respuesta diferentes.

Por ejemplo, cree un evento con la acción Registrar aceptación de oferta para cada tipo de respuesta. Defina las siguientes propiedades personalizadas en la tabla UA_UsrResponseType [como Nombre (código)]: Explorar (EXP), Considerar (CON) y Confirmar (CMT). A continuación, cree tres eventos y denomínelos LogAccept_Explore, LogAccept_Consider y LogAccept_Commit. Los tres eventos son exactamente el mismo (tienen la acción Registrar aceptación de oferta), pero los nombres son diferentes, para que la persona que trabaja con la API pueda distinguirlos.

O bien, puede crear un solo evento con la acción Registrar aceptación de oferta que utiliza para todos los tipos de respuesta personalizados. Por ejemplo, denomínelo LogCustomResponse.

Cuando trabaja con la API, no hay ninguna diferencia funcional entre los eventos, pero los convenios de denominación pueden clarificar el código. Asimismo, si asigna a cada respuesta personalizada un nombre diferente, el informe Resumen de actividad de eventos de canal muestra información más precisa.

En primer lugar, configure todos los pares nombre-valor

```
//Definir pares de nombre y valor para UACIResponseTypeCode
// Tipo de respuesta Explore
NameValuePair responseTypeEXP = new NameValuePairImpl();
responseTypeEXP.setName("UACIResponseTypeCode");
responseTypeEXP.setValueAsString("EXP");
responseTypeEXP.setValueDataType(NameValuePair.DATA_TYPE_STRING);

// Tipo de respuesta Consider
NameValuePair responseTypeCON = new NameValuePairImpl();
responseTypeCON.setName("UACIResponseTypeCode");
responseTypeCON.setValueAsString("CON");
responseTypeCON.setValueDataType(NameValuePair.DATA_TYPE_STRING);

// Tipo de respuesta Commit
NameValuePair responseTypeCMT = new NameValuePairImpl();
responseTypeCMT.setName("UACIResponseTypeCode");
responseTypeCMT.setValueAsString("CMT");
responseTypeCMT.setValueDataType(NameValuePair.DATA_TYPE_STRING);

//Definir pares de nombre y valor para UACILOGTOLEARNING
//No se registra en el aprendizaje
NameValuePair noLogToLearning = new NameValuePairImpl();
noLogToLearning.setName("UACILOGTOLEARNING");
noLogToLearning.setValueAsString("0");
noLogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);

//Se registra en el aprendizaje
NameValuePair LogToLearning = new NameValuePairImpl();
```

```
LogToLearning.setName("UACILogToLearning");
LogToLearning.setValueAsString("1");
LogToLearning.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
```

Este primer ejemplo muestra cómo utilizar los eventos individuales.

```
//EJEMPLO 1: Este conjunto de llamadas postEvent utilizan los eventos
denominados individualmente
//PostEvent con una respuesta Explore
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogAccept_Explore, postEventParameters);

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

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

Este segundo ejemplo muestra cómo utilizar un solo evento.

```
//EJEMPLO 2: Este conjunto de llamadas postEvent utilizan un solo evento
//PostEvent con una respuesta Explore
NameValuePair[] postEventParameters = { responseTypeEXP, noLogToLearning };
response = api.postEvent(sessionId, LogCustomResponse, postEventParameters);

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

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

Ambos ejemplos realizan exactamente las mismas acciones, sin embargo, una versión puede ser más fácil de leer que la otra.

Correlación de tablas de preparación del entorno de ejecución con tablas de historial de Campaign

Las tablas de preparación del historial de contactos de Interact se correlacionan con las tablas de historial de Campaign. Debe tener una de las tablas de preparación del entorno de ejecución para cada nivel de audiencia.

Correlación de la tabla de preparación del historial de contactos UACI_CHStaging

Esta tabla muestra cómo se correlaciona la tabla de preparación del entorno de ejecución `UACI_CHStaging` con la tabla del historial de contactos de Campaign. Los nombres de tabla que se muestran son las tablas de muestra creadas para la audiencia predeterminada en las tablas de ejecución y las tablas del sistema de Campaign.

Tabla 3. Historial de contactos

UACI_CHStaging **Nombre de columna de la tabla de preparación de historial de contactos de Interact**	**Tabla de historial de contactos de Campaign**	**Nombre de columna de la tabla**
ContactID	N/D	N/D
TreatmentCode	UA_Treatment	TreatmentCode

Tabla 3. Historial de contactos (continuación)

UACI_CHStaging **Nombre de columna de la tabla de preparación de historial de contactos de Interact**	**Tabla de historial de contactos de Campaign**	**Nombre de columna de la tabla**
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` es una clave para unir la tabla `UACI_CHOfferAtrib` con la tabla `UACI_CHStaging`. La columna `userDefinedFields` puede contener los datos que elija.

Correlación de la tabla de preparación del historial de contactos UACI_CHOfferAttrib

Esta tabla muestra cómo se correlaciona la tabla de preparación del entorno de ejecución `UACI_CHOfferAttrib` con la tabla del historial de contactos de Campaign. Los nombres de tabla que se muestran son las tablas de muestra creadas para la audiencia predeterminada en las tablas de ejecución y las tablas del sistema de Campaign.

Tabla 4. Atributos de ofertas

UACI_CHOfferAttrib **Nombre de columna de la tabla de preparación de historial de contactos de Interact**	**Tabla de historial de contactos de Campaign**	**Nombre de columna de la tabla**
ContactID	N/D	N/D
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

Correlación de la tabla de preparación del historial de respuestas de contactos UACI_RHStaging

Esta tabla muestra cómo se correlaciona la tabla de preparación del entorno de ejecución `UACI_RHStaging` con la tabla del historial de respuestas de Campaign. Los nombres de tabla que se muestran son las tablas de muestra creadas para la audiencia predeterminada en las tablas de ejecución y las tablas del sistema de Campaign.

Tabla 5. Historial de respuestas

UACI_RHStaging **Nombre de columna de la tabla de preparación del historial de respuestas de Interact**	**Tabla del historial de respuestas de Campaign**	**Nombre de columna de la tabla**
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` es una clave que utiliza el módulo de historial de contactos y respuestas para identificar datos, pero no se registra en las tablas de respuestas de Campaign. La columna `userDefinedFields` puede contener los datos que elija.

Columnas adicionales en las tablas de preparación

Si añade columnas a las tablas de preparación, el módulo de historial de contactos y respuestas los graba en las tablas `UA_DtlContactHist` o `UA_ResponseHistory` en columnas de la misma tabla.

Por ejemplo, si añade la columna `linkFrom` a la tabla `UACI_CHStaging`, el módulo de historial de contactos y respuestas copia esos datos en la columna `linkFrom` de la tabla `UA_DtlContactHist`.

Columnas adicionales en las tablas de historial de respuestas y contactos de Campaign

Si tiene columnas adicionales en las tablas de historial de contactos y respuestas de Campaign, añada las columnas coincidentes a las tablas de preparación antes de ejecutar el módulo de historial de contactos y respuestas.

Puede completar columnas adicionales de las tablas de preparación creando columnas con el mismo nombre que los pares nombre-valor de los datos de sesión de ejecución.

Por ejemplo, puede crear pares nombre-valor `NumberItemsInWishList` y `NumberItemsInShoppingCart`, y añadirlos a la tabla `UACI_RHStaging`. Cuando se produzca un evento Registrar aceptación de oferta o Registrar rechazo de oferta, el entorno de ejecución completará estos campos. El entorno de ejecución completa la tabla `UACI_CHStaging` cuando se produce un evento Registrar contacto de oferta.

Uso de tablas para incluir una puntuación para una oferta

Puede utilizar los campos definidos por el usuario para incluir la puntuación utilizada para presentar una oferta. Añada una columna denominada `PuntuaciónFinal` a la tabla `UACI_CHStaging` de las tablas de ejecución y a la tabla `UA_DtlContactHist` de las tablas del sistema de Campaign. Interact completa automáticamente la columna `PuntuaciónFinal` con la puntuación final utilizada para la oferta si está utilizando aprendizaje incorporado.

Si está creando un módulo de aprendizaje personalizado, puede utilizar el método setActualValueUsed de la interfaz de ITreatment y el método logEvent de la interfaz de ILearning.

Si no está utilizando aprendizaje, añada una columna denominada Puntuación a la tabla UACI_CHStaging de las tablas de ejecución y a la tabla UA_DtlContactHist de las tablas del sistema de Campaign. Interact completa automáticamente la columna Puntuación con la puntuación utilizada para la oferta.

Crear nuevas tablas de historial en Campaign y tablas de preparación en Interact

Si está utilizando un nivel de audiencia distinto del de Cliente, tendrá que crear tablas de historial nuevas en Campaign y tablas de preparación nuevas en Interact.

Por ejemplo, el script de ejemplo siguiente se utiliza en la base de datos de tiempo de diseño de DB2 de IBM para crear tablas de historial en Campaign para un nivel de audiencia del tipo Cuenta.

```
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;
```

El script del ejemplo siguiente se utiliza en la base de datos DB2 de IBM de tiempo de ejecución para crear tablas de preparación de historial en Interact para un nivel de audiencia del tipo Cuenta.

```
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);
```

Configuración de la supervisión JMX para el módulo de historial de contactos y respuestas

Utilice este procedimiento para configurar la supervisión JMX para el módulo de historial de contactos y respuestas. Los protocolos JMXMP y RMI están soportados. La configuración de la supervisión JMX no habilita la seguridad para el módulo de historial de contactos y respuestas. Utilice Marketing Platform para el entorno de diseño para configurar la supervisión JMX.

Acerca de esta tarea

Para utilizar la herramienta de supervisión JMX para el módulo de historial de contactos y respuestas, la dirección predeterminada que se utiliza para:

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

Cuando visualiza los datos en la herramienta de supervisión JMX, los atributos de resultados se organizan primero por partición y, a continuación, por nivel de audiencia.

Procedimiento

En la Marketing Platform del entorno de diseño, edite las siguientes propiedades de configuración en la categoría `Campaign > monitoring`.

Propiedad de configuración	Valor
`monitorEnabledForInteract`	**True**
port	El número de puerto del servicio JMX
protocol	El protocolo que se utiliza: • **JMXMP** • **RMI** La seguridad no está habilitada para el módulo de historial de contactos y respuestas, aunque seleccione el protocolo JMXMP.

Acerca del seguimiento de respuestas de sesiones cruzadas

Los visitantes no siempre completan una transacción en una única visita al punto de encuentro. Un cliente puede añadir un elemento a su carro de compra en el sitio web y no completar la compra hasta dos días después. No es factible mantener la sesión de ejecución activa de forma indefinida. Puede habilitar el seguimiento de respuestas de sesiones cruzadas para realizar seguimiento de una presentación de oferta en una sesión y correlacionarla con una respuesta en otra sesión.

De forma predeterminada, el seguimiento de respuestas de sesiones cruzadas de Interact puede realizar la correlación según los códigos de tratamiento o los códigos de oferta. También puede configurarlo para correlacionar cualquier código

personalizado que elija. La respuesta de sesiones cruzadas se correlaciona según los datos disponibles. Por ejemplo, el sitio web incluye una oferta con un código promocional generado en el momento de la visualización de un descuento válido durante una semana. Un usuario puede añadir artículos a su carro de compra, pero no completar la compra hasta tres días después. Cuando se utiliza la llamada a `postEvent` para registrar un evento de aceptación, puede incluir solo el código promocional. Dado que el tiempo de ejecución no puede encontrar un código de tratamiento u oferta que correlacionar en la sesión actual, el tiempo de ejecución coloca el evento de aceptación con la información disponible en una tabla de preparación de respuesta de sesiones cruzadas (XSessResponse). El servicio CrossSessionResponse lee periódicamente la tabla XSessResponse e intenta correlacionar los registros con los datos de historial de contactos disponibles. El servicio CrossSessionResponse correlaciona el código promocional con el historial de contactos y recopila todos los datos necesarios para registrar la respuesta adecuada. A continuación, el servicio CrossSessionResponse graba la respuesta en las tablas de preparación de respuestas y, si el aprendizaje está habilitado, las tablas de aprendizaje. A continuación, el módulo del historial de contactos y respuestas graba la respuesta en las tablas del historial de contactos y respuestas de Campaign. El proceso satisfactorio de respuesta entre distintas sesiones depende de los registros del historial de contactos originales que se hayan migrado a la base de datos de Campaign mediante el proceso ETL del historial de contactos.

Configuración del origen de datos de seguimiento de respuestas de sesiones cruzadas

El seguimiento de respuestas de sesiones cruzadas de Interact correlaciona los datos de sesión del entorno de ejecución con el historial de contactos y respuestas de Campaign. De forma predeterminada, el seguimiento de respuestas de sesiones cruzadas realiza la correlación según el código de tratamiento o el código de oferta. Puede configurar el entorno de ejecución para que realice la correlación según un código alternativo personalizado.

- Si elige correlacionar según un código alternativo, debe definir el código alternativo en la tabla `UACI_TrackingType` de las tablas de ejecución de Interact.
- El entorno de ejecución debe tener acceso a las tablas de historial de contactos de Campaign. Puede ser configurando el entorno de ejecución para tener acceso a las tablas de historial de contactos de Campaign o creando una copia de las tablas del historial de contactos en el entorno de ejecución.

  Este acceso es de solo lectura y es independiente de la utilidad de historial de contactos y respuestas.

  Si crea una copia de las tablas, es su responsabilidad asegurarse de que los datos de la copia del historial de contactos sean precisos. Puede configurar el periodo de tiempo que el servicio CrossSessionResponse mantiene las respuestas no correlacionadas para correlacionar la frecuencia con la que se renuevan los datos en la copia de las tablas del historial de contactos utilizando la propiedad `purgeOrphanResponseThresholdInMinutes`. Si está utilizando el módulo de historial de contactos y respuestas, debe coordinar las actualizaciones de ETL para asegurarse de que tienen los datos más actuales.

Configuración de las tablas de historial de contactos y respuestas para seguimiento de respuestas de sesiones cruzadas

Tanto si crea una copia de las tablas de historial de contactos como si utiliza las propias tablas en las tablas del sistema de Campaign, debe realizar los pasos siguientes para configurar las tablas de historial de contactos y respuestas.

Antes de empezar

Las tablas de historial de contactos y respuestas se deben correlacionar correctamente en Campaign antes de realizar estos pasos.

Procedimiento

1. Ejecute el script SQL `aci_lrnfeature` en el directorio `interactDT/ddl/acifeatures` del directorio de instalación del entorno de diseño de Interact en las tablas `UA_DtlContactHist` y `UA_ResponseHistory` de las tablas del sistema de Campaign.

   Esto añade la columna `RTSelectionMethod` a las tablas `UA_DtlContactHist` y `UA_ResponseHistory`. Ejecute el script `aci_lrnfeature` en estas tablas para cada uno de sus niveles de audiencia. Edite el script según sea necesario con la tabla correcta para cada nivel de audiencia.
2. Si desea copiar las tablas de historial de contactos en el entorno de ejecución, hágalo ahora.

   Si está creando una copia de las tablas de historial de contactos de Campaign accesible por el entorno de ejecución para el soporte de seguimiento de respuestas de sesiones cruzadas, utilice las directrices siguientes:
   - El seguimiento de respuestas de sesiones cruzadas requiere acceso de solo lectura a estas tablas.
   - El seguimiento de respuestas de sesiones cruzadas requiere las tablas siguientes del historial de contactos de Campaign.
     - `UA_DtlContactHist` (para cada nivel de audiencia)
     - `UA_Treatment`

     Debe actualizar regularmente los datos de estas tablas para asegurarse de que se realiza un seguimiento preciso de las respuestas.
3. Ejecute el script SQL `aci_crhtab` en el directorio `ddl` del directorio de instalación del entorno de ejecución de Interact en el origen de datos del historial de contactos y respuestas.

   Este script crea las tablas `UACI_XsessResponse` y `UACI_CRHTAB_Ver`.
4. Cree una versión de la tabla `UACI_XsessResponse` para cada nivel de audiencia.

Resultados

Para mejorar el rendimiento del seguimiento de respuestas de sesiones cruzadas, es posible que desee limitar la cantidad de datos de historial de contactos, ya sea mediante la forma en la que copia los datos de historial de contactos o configurando una vista a las tablas de historial de contactos de Campaign. Por ejemplo, si tiene una práctica de empresa por la que ninguna oferta es válida más de 30 días, debe limitar los datos del historial de contactos a los últimos 30 días. Para modificar el número de días de datos de historial de contactos que se deben mantener, abra la propiedad de configuración **Campaign | particiones | partición*n* | Interact | contactAndResponseHistTracking** y establezca el valor de **daysBackInHistoryToLookupContact**.

No verá resultados del seguimiento de respuestas de sesiones cruzadas hasta que se ejecute el módulo de historial de contactos y respuestas. Por ejemplo, el valor predeterminado `processSleepIntervalInMinutes` es de 60 minutos. Por lo tanto, puede pasar al menos una hora hasta que las respuestas de sesiones cruzadas aparezcan en el historial de respuestas de Campaign.

Tabla UACI_TrackingType

La tabla UACI_TrackingType forma parte de las tablas del entorno de ejecución. Esta tabla define los códigos de seguimiento utilizados con el seguimiento de respuestas de sesiones cruzadas. El código de seguimiento define qué método utiliza el entorno de ejecución para correlacionar la oferta actual de una sesión de ejecución con el historial de contactos y respuestas.

Columna	Tipo	Descripción
TrackingCodeType	int	Número que representa el tipo de código de seguimiento. Los comandos SQL utilizados para correlacionar información de los datos de sesión con las tablas de historial de contactos y respuestas hacen referencia a este número.
Name	varchar(64)	Nombre del tipo de código de seguimiento. Se pasa a los datos de sesión mediante el parámetro reservado UACI_TrackingCodeType con el método postEvent.
Descripción	varchar(512)	Breve descripción del tipo de código de seguimiento. Este campo es opcional.

De forma predeterminada, el entorno de ejecución tiene dos tipos de código de seguimiento definidos, tal como se muestra en la tabla siguiente. Para cualquier código alternativo, debe definir un TrackingCodeType exclusivo.

TrackingCodeType	Nombre	Descripción
1	Código de tratamiento	Código de tratamiento generado por UACI
2	Código de oferta	Código de oferta de campaña UAC

UACI_XSessResponse

La tabla UACI_XSessResponse forma parte de las tablas del entorno de ejecución. Esta tabla se utiliza para el seguimiento de respuestas de sesiones cruzadas.

Debe existir una instancia de esta tabla para cada nivel de audiencia en el origen de datos del historial de contactos y respuestas disponible para el seguimiento de respuestas de sesiones cruzadas de Interact.

Columna	Tipo	Descripción
SeqNumber	bigint	Identificador de la fila de datos. El servicio CrossSessionResponse procesa todos los registros en el orden SeqNumber.
ICID	bigint	ID de canal interactivo
IDAudiencia	bigint	ID de audiencia de este nivel de audiencia. El nombre de esta columna debe coincidir con el ID de audiencia definido en Campaign. La tabla de muestra contiene la columna CustomerID.
TrackingCode	varchar(64)	Valor que pasa el parámetro UACIOfferTrackingCode del método postEvent.
TrackingCodeType	int	Representación numérica del código de seguimiento. El valor debe ser una entrada válida de la tabla UACI_TrackingType.

Columna	Tipo	Descripción
OfferID	bigint	ID de oferta tal como se define en Campaign.
ResponseType	int	Tipo de respuesta para este registro. El valor debe ser una entrada válida de la tabla UA_UsrResponseType.
ResponseTypeCode	varchar(64)	Código de tipo de respuesta para este registro. El valor debe ser una entrada válida de la tabla UA_UsrResponseType.
ResponseDate	datetime	Fecha de la respuesta.
Mark	bigint	El valor de este campo identifica el estado del registro. • 1: en curso • 2: satisfactorio • NULL: reintentar • -1: el registro ha estado en la base de datos durante más de purgeOrphanResponseThresholdInMinutes minutos. Como parte del mantenimiento de esta tabla que realiza el administrador de bases de datos, puede seleccionar este campo para registros que no se correlacionan, es decir, todos los registros con valor de -1. Todos los registros con valor 2 los elimina automáticamente el servicio CrossSessionResponse.
UsrDefinedFields	char(18)	Los campos personalizados que desea incluir cuando correlaciona respuestas de oferta con el historial de contactos y respuestas. Por ejemplo, si desea correlacionar según un código promocional, incluya un campo definido por el usuario de código promocional.

Habilitación del seguimiento de respuestas de sesiones cruzadas

Utilice este procedimiento para habilitar el seguimiento de respuestas de sesiones cruzadas.

Antes de empezar

Debe configurar el módulo del historial de contactos y respuestas para beneficiarse completamente del seguimiento de respuestas de sesiones cruzadas.

Para utilizar seguimiento de respuestas de sesiones cruzadas, debe configurar el entorno de ejecución para tener acceso de lectura a las tablas de historial de contactos y respuestas de Campaign. Puede leer las propias tablas de historial de contactos y respuestas de Campaign en el entorno de diseño, o una copia de las tablas en los orígenes de datos del entorno de ejecución. La configuración del entorno de ejecución para tener acceso de lectura a la tabla de historial de contactos y respuestas es independiente de la configuración del módulo de historial de contactos y respuestas.

Si está realizando la correlación con algún elemento que no es el código de tratamiento o el código de oferta, debe añadirlo a la tabla UACI_TrackingType.

Procedimiento

1. Cree las tablas XSessResponse en las tablas de historial de contactos y respuestas accesibles al entorno de ejecución.
2. Defina las propiedades en la categoría `contactAndResponseHistoryDataSource` para el entorno de ejecución.
3. Defina la propiedad `crossSessionResponseTable` para cada nivel de audiencia.
4. Cree una categoría `OverridePerAudience` para cada nivel de audiencia.

Correlación de ofertas de respuesta entre sesiones

De forma predeterminada, la correlación del seguimiento de respuestas de sesiones cruzadas se realiza según los códigos de tratamiento o los códigos de oferta. El servicio crossSessionResponse utiliza comandos SQL para correlacionar códigos de tratamiento, códigos de oferta o un código personalizado de datos de sesión con tablas de historial de contactos y respuestas de Campaign. Puede editar estos comandos SQL para correlacionar las personalizaciones que se hayan realizado en los códigos de seguimiento, códigos de oferta o códigos personalizados.

Correlación por código de tratamiento

El SQL para correlacionar por código de tratamiento debe devolver todas las columnas de la tabla XSessResponse para este nivel de audiencia además de una columna denominada `CorrelaciónIDOferta`. El valor de la columna `CorrelaciónIDOferta` debe ser el `offerId` que va con el código de tratamiento del registro XSessResponse.

A continuación se muestra un ejemplo del comando SQL generado predeterminado que coincide con los códigos de tratamiento. Interact genera el SQL para utilizar los nombres de tabla correctos para el nivel de audiencia. Este SQL se utiliza si la propiedad `Interact > services > crossSessionResponse > OverridePerAudience > nivelAudiencia > TrackingCodes > byTreatmentCode > SQL` está establecida en **Utilizar SQL generado por el 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
```

Los valores `UACI_XsessResponse`, `UA_DtlContactHist`, `CustomerID` y `UA_ContactHistory` los definen los valores de Interact. Por ejemplo, `UACI_XsessResponse` lo define la propiedad de configuración `Interact > profile > Audience Levels > [NombreNivelAudiencia] > crossSessionResponseTable`.

Si ha personalizado las tablas de historial de contactos y respuestas, es posible que necesite revisar este SQL para trabajar con las tablas. Puede definir sustituciones de SQL en la propiedad `Interact > services > crossSessionResponse > OverridePerAudience > (NivelAudiencia) > TrackingCodes > byTreatmentCode > OverrideSQL`. Si proporciona algún SQL de sustitución, debe cambiar también la propiedad `SQL` a **SQL de sustitución**.

Correlación por código de oferta

El SQL para correlacionar por código de oferta debe devolver todas las columnas de la tabla XSessResponse para este nivel de audiencia además de una columna denominada CorrelaciónCódigoTratamiento. El valor de la columna CorrelaciónCódigoTratamiento es el código de tratamiento que va con el ID de oferta (y código de oferta) en el registro XSessResponse.

A continuación se muestra un ejemplo del comando SQL generado predeterminado que correlaciona los códigos de oferta. Interact genera el SQL para utilizar los nombres de tabla correctos para el nivel de audiencia. Este SQL se utiliza si la propiedad Interact > services > crossSessionResponse > OverridePerAudience > *NivelAudiencia* > TrackingCodes > byOfferCode > SQL está establecida en **Utilizar SQL generado por el 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.*,
   0from UACI_XSessResponse tx
Left Outer Join UA_ContactHistory ch ON tx.CustomerID =ch.CustomerID
Left Outer Join UA_Treatment treatment ON tx.offerId = treatment.offerId
Left Outer Join
   (
      select   max(ch.contactDateTime) as maxDate,
            treatment.offerId, ch.CustomerID
      from UA_ContactHistory ch, UA_Treatment treatment, UACI_XSessResponse tx
      where  tx.CustomerID =ch.CustomerID
      and tx.offerID = treatment.offerId
      and treatment.cellID = ch.cellID
   and treatment.packageID=ch.packageID
      group by ch.CustomerID,  treatment.offerId
      ) ch_by_max_date ON tx.CustomerID =ch_by_max_date.CustomerID
      and tx.offerId = ch_by_max_date.offerId
      and treatment.cellID = ch.cellID
   and treatment.packageID=ch.packageID
where   tx.mark = 1
      and       ch.contactDateTime = ch_by_max_date.maxDate
      and treatment.cellID = ch.cellID
   and       treatment.packageID=ch.packageID
      and tx.offerID = treatment.offerId
   and       tx.trackingCodeType=2
```

Los valores UACI_XsessResponse, UA_DtlContactHist, CustomerID y UA_ContactHistory los definen los valores de Interact. Por ejemplo, UACI_XsessResponse lo define la propiedad de configuración Interact > profile > Audience Levels > [NombreNivelAudiencia] > crossSessionResponseTable.

Si ha personalizado las tablas de historial de contactos y respuestas, es posible que necesite revisar este SQL para trabajar con las tablas. Puede definir sustituciones de SQL en la propiedad Interact > services > crossSessionResponse > OverridePerAudience > *(NivelAudiencia)* > TrackingCodes > byOfferCode > OverrideSQL. Si proporciona algún SQL de sustitución, debe cambiar también la propiedad SQL a **SQL de sustitución**.

Correlación por código alternativo

Puede definir un comando SQL para correlacionar por algún código alternativo que elija. Por ejemplo, podría tener códigos promocionales o códigos de producto además de los códigos de oferta o tratamiento.

Debe definir este código alternativo en la tabla UACI_TrackingType en las tablas de entorno de ejecución de Interact.

Debe proporcionar SQL o un procedimiento almacenado en la propiedad Interact > services > crossSessionResponse > OverridePerAudience > *(nivelAudiencia)* > TrackingCodes > byAlternateCode > OverrideSQL que devuelve todas las columnas de la tabla XSessResponse para este nivel de audiencia además de las columnas CorrelaciónIDTratamiento y CorrelaciónIDOferta. Opcionalmente, puede devolver el códigoOferta en lugar de CorrelaciónIDOferta (con el formato códigoOferta1, códigoOferta2, ... códigoOferta*N* para códigos de oferta de parte N). Los valores de la columna CorrelaciónCódigoTratamiento y CorrelaciónIDOferta (o columnas de código de oferta) deben corresponderse con el CódigoSeguimiento en el registro XSessResponse.

Por ejemplo, el siguiente pseudocódigo SQL de la columna CódigoAlternativo de la tabla 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>
```

Donde <*x*> es el código de seguimiento definido en la tabla UACI_TrackingType.

Utilización de una utilidad de carga de base de datos con el entorno de ejecución

De forma predeterminada, el entorno de ejecución graba los datos del historial de contactos y respuestas de los datos de sesión en las tablas de preparación. Sin embargo, en un sistema de producción muy activo, la cantidad de memoria necesaria para guardar en caché todos los datos antes de que el tiempo de ejecución pueda grabarlos en las tablas de preparación puede ser prohibitiva. Puede configurar el tiempo de ejecución para utilizar una utilidad de carga de base de datos para mejorar el rendimiento.

Cuando habilita una utilidad de carga de base de datos, en lugar de alojar todo el historial de contactos y respuestas en memoria antes de grabar en las tablas de

preparación, el tiempo de ejecución graba los datos en un archivo de preparación. Puede definir la ubicación del directorio que contiene los archivos de preparación con la propiedad `externalLoaderStagingDirectory`. Este directorio contiene varios subdirectorios. El primer subdirectorio es el directorio de instancia de tiempo de ejecución, que contiene los directorios `contactHist` y `respHist`. Los directorios `contactHist` y `respHist` contienen subdirectorios con nombre exclusivo y el formato *nombreNivelAudiencia.ID_exclusivo.estadoActual*, que contienen los archivos de preparación.

Estado Actual	Descripción
CACHE	Contenido del directorio que actualmente se está grabando en un archivo.
READY	Contenido de un directorio listo para su proceso.
RUN	Contenido del directorio que se está grabando actualmente en la base de datos.
PROCESSED	El contenido del directorio se ha grabado en la base de datos.
ERROR	Se ha producido un error al grabar el contenido del directorio en la base de datos.
ATTN	El contenido del directorio requiere su atención. Es decir, es posible que necesite realizar pasos manuales para completar la grabación del contenido de este directorio en la base de datos.
RERUN	El contenido del directorio está listo para grabarse en la base de datos. Debe renombrar un directorio de `ATTN` o `ERROR` a `RERUN` después de haber corregido el problema.

Puede definir el directorio de la instancia de tiempo de ejecución definiendo la propiedad de JVM interact.runtime.instance.name en el script de inicio del servidor de aplicaciones. Por ejemplo, puede añadir `-Dinteract.runtime.instance.name=instance2` al script de inicio del servidor de aplicaciones web. Si no está establecida, el nombre predeterminado es `DefaultInteractRuntimeInstance`.

El directorio `samples` contiene archivos de muestra para ayudarle a escribir sus propios archivos de control de utilidad de carga de base de datos.

Habilitación de una utilidad de carga de base de datos con el entorno de ejecución

Utilice este procedimiento para habilitar una utilidad de carga de base de datos con el entorno de ejecución.

Antes de empezar

Debe definir los archivos de control o comando de la utilidad de carga de base de datos antes de configurar el entorno de ejecución para utilizarlos. Estos archivos deben existir en la misma ubicación en todos los servidores de ejecución en el mismo grupo de servidores.

Interact proporciona archivos de control y comando de muestra en el directorio `loaderService` en la instalación del servidor de ejecución de Interact.

Procedimiento

1. Confirme que el usuario del entorno de ejecución tenga credenciales de inicio de sesión para el origen de datos de tablas de ejecución definido en `Interact > general > systemTablesDataSource` en sus propiedades de configuración.
2. Defina las propiedades de configuración `Interact > general > systemTablesDataSource > loaderProperties`.
3. Defina la propiedad `Interact > services >externalLoaderStagingDirectory`.
4. Revise las propiedades de configuración `Interact > services > responseHist > fileCache`, si es necesario.
5. Revise las propiedades de configuración `Interact > services > contactHist > fileCache`, si es necesario.
6. Reinicie el servidor de ejecución.

Proceso de ETL de patrón de evento

Para procesar grandes cantidades de datos de patrón de evento de IBM Interact y hacer que esos datos estén disponibles para consultas y la creación de informes, puede instalar un proceso de ETL (Extract, Transform, Load) autónomo en cualquier servidor soportado para conseguir un rendimiento óptimo.

En Interact, todos los datos de patrón de evento correspondientes a un ID de audiencia determinado se almacenan como colección individual en las tablas de base de datos de ejecución. La información sobre el ID de audiencia y el estado de patrón se almacena como un objeto grande binario (BLOB). Para realizar consultas SQL o crear informes basándose en patrones de evento, es necesario este nuevo proceso de ETL para dividir el objeto en tablas de una base de datos de destino. Para ello, el proceso de ETL autónomo obtiene datos de patrón de evento de las tablas de base de datos de ejecución de Interact, los procesa de acuerdo con la planificación que especifique el usuario y los almacena en la base de datos de destino, donde pueden utilizarse en consultas SQL o informes adicionales.

Además de mover los datos de patrón de evento a la base de datos de destino y transformarlos, el proceso de ETL autónomo también sincroniza los datos de la base de datos de destino con la información más actual de la base de datos de ejecución de Interact. Por ejemplo, si suprime un patrón de evento en el entorno de ejecución de Interact, los datos procesados de ese patrón de evento se eliminan de la base de datos de destino la próxima vez que se ejecute el proceso de ETL. La información sobre el estado de patrón de evento también se mantiene actualizada. Por lo tanto, la información sobre patrones de evento almacenada en la base de datos de destino consta únicamente de datos actuales, no información histórica.

Ejecución del proceso de ETL autónomo

Cuando inicia el proceso de ETL autónomo en un servidor, se ejecuta continuamente en segundo plano hasta que se detenga. El proceso sigue las instrucciones contenidas en las propiedades de configuración de Marketing Platform para determinar la frecuencia, las conexiones de base de datos y otros detalles durante su operación.

Antes de empezar

Antes de ejecutar el proceso ETL autónomo, no olvide llevar a cabo las tareas siguientes:

- Debe tener permiso de un rol de usuario administrativo de Interact.

- Debe haber instalado el proceso en un servidor, y haber configurado los archivos en el servidor y en Marketing Platform correctamente para su configuración.

Nota:

Si está ejecutando el proceso de ETL en Microsoft Windows para un idioma distinto del inglés americano, ejecute `chcp` en el indicador de comandos para establecer la página de códigos correspondiente al idioma utilizado. Por ejemplo, puede utilizar cualquiera de los códigos siguientes: ja_jp=932, zh_cn=936, ko_kr=949, ru_ru=1251. Para de_de, fr_fr, it_it, es_es, pt_br, utilice 1252. Para garantizar una visualización correcta de los caracteres, ejecute el comando `chcp` en el indicador de comandos de Windows antes de iniciar el proceso de ETL.

Acerca de esta tarea

Después de instalar y configurar el proceso de ETL autónomo, está preparado para iniciar el proceso.

Procedimiento

1. Abra un indicador de comandos en el servidor donde está instalado el proceso de ETL.
2. Vaya al directorio `<directorio_inicial_Interact>/PatternStateETL/bin`, donde residen los archivos ejecutables del proceso de ETL.
3. Ejecute el archivo `command.bat` (en Microsoft Windows) o el archivo `command.sh` (en sistemas operativos tipo UNIX) con los parámetros siguientes:
   - `-u <nombre_usuario>`. Este valor debe ser un usuario válido de Marketing Platform y debe haber configurado ese usuario con acceso a los orígenes de datos **TargetDS** y **RuntimeDS** que utilizará el proceso de ETL.
   - `-p <contraseña>`. Sustituya *<contraseña>* por la contraseña correspondiente al usuario que ha especificado. Si la contraseña de este usuario está en blanco, especifique dos comillas dobles (como en `-p ""`). La contraseña es opcional cuando ejecuta el archivo del comando; si omite la contraseña en el comando, se le solicitará que la especifique cuando se ejecute el comando.
   - `-c <nombre_perfil>`. Sustituya *<nombre_perfil>* por el nombre exacto que ha especificado en Marketing Platform en la configuración de **Interact | PatternStateETL** que ha creado.

     El nombre que especifique aquí debe coincidir con el valor que ha especificado en el campo **Nombre de categoría nuevo** al crear la configuración.
   - `start`. El comando start es necesario para iniciar el proceso.

   Por lo tanto, el comando completo para iniciar el proceso tendrá esta forma:

   ```
   command.bat -u <nombre_usuario> -p <contraseña> -c <nombre_perfil> start
   ```

Resultados

El proceso de ETL autónomo se ejecuta de forma continua en segundo plano hasta que el usuario detiene el proceso o se reinicia el servidor.

Nota:

La primera vez que ejecuta el proceso, los datos acumulados de patrón de evento pueden tardar una cantidad considerable de tiempo en ejecutarse. Las próximas veces que se ejecute el proceso, trabajará sólo con el conjunto de datos de patrón de evento más reciente y tardará menos tiempo en completarse.

Tenga en cuenta que también puede proporcionar el argumento `help` al archivo `command.bat` o `command.sh` para ver todas las opciones disponibles, como en el ejemplo siguiente:

```
command.bat help
```

Detención del proceso de ETL autónomo

Cuando inicia el proceso de ETL autónomo en un servidor, se ejecuta continuamente en segundo plano hasta que se detenga.

Acerca de esta tarea

Procedimiento

1. Abra un indicador de comandos en el servidor donde está instalado el proceso de ETL.
2. Vaya al directorio `<directorio_inicial_Interact>/PatternStateETL/bin`, donde residen los archivos ejecutables del proceso de ETL.
3. Ejecute el archivo `command.bat` (en Microsoft Windows) o el archivo `command.sh` (en sistemas operativos tipo UNIX) con los parámetros siguientes:
   - `-u <nombre_usuario>`. Este valor debe ser un usuario válido de Marketing Platform y debe haber configurado ese usuario con acceso a los orígenes de datos **TargetDS** y **RuntimeDS** que utilizará el proceso de ETL.
   - `-p <contraseña>`. Sustituya *<contraseña>* por la contraseña correspondiente al usuario que ha especificado. Si la contraseña de este usuario está en blanco, especifique dos comillas dobles (como en `-p ""`). La contraseña es opcional cuando ejecuta el archivo del comando; si omite la contraseña en el comando, se le solicitará que la especifique cuando se ejecute el comando.
   - `-c <nombre_perfil>`. Sustituya *<nombre_perfil>* por el nombre exacto que ha especificado en Marketing Platform en la configuración de **Interact | PatternStateETL** que ha creado.

     El nombre que especifique aquí debe coincidir con el valor que ha especificado en el campo **Nombre de categoría nuevo** al crear la configuración.
   - `stop`. El comando stop es necesario para detener el proceso. Si utiliza este comando, se completarán las operaciones de ETL en curso antes de que se cierre el proceso.

     Para cerrar el proceso de ETL sin esperar a que finalicen las operaciones en curso, utilice `forcestop` en lugar de `stop`.

   Por lo tanto, el comando completo para iniciar el proceso tendrá esta forma:

   ```
   command.bat -u <nombre_usuario> -p <contraseña> -c <nombre_perfil> stop
   ```

Resultados

El proceso de ETL autónomo se detiene.

Capítulo 4. Presentación de ofertas

Puede configurar Interact de varias formas para mejorar cómo selecciona las ofertas que se presentan. En las secciones siguientes se describen estas características opcionales con más detalle.

Elegibilidad de una oferta

El objetivo de Interact es presentar las ofertas elegibles. Simplemente, Interact presenta la oferta más óptima entre las elegibles, según el visitante, el canal y la situación.

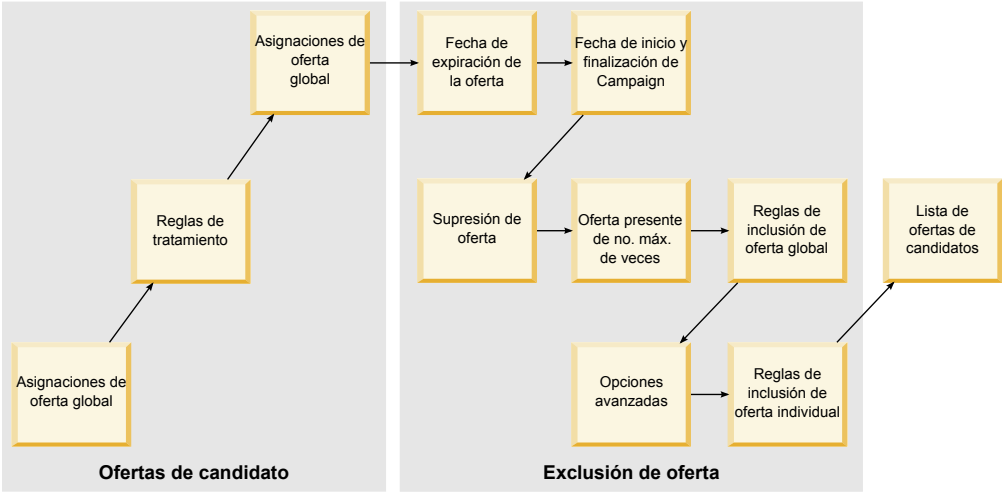
Las reglas de tratamiento son sólo el principio de cómo Interact determina qué ofertas son elegibles para un cliente. Interact tiene varias características opcionales que puede implementar para mejorar cómo determina el entorno de ejecución qué ofertas debe presentar. Ninguna de estas características garantiza que una oferta se presente a un cliente. Estas características influyen en la probabilidad de que una oferta sea elegible para su presentación a un cliente. Puede utilizar tantas o tan pocas características como necesite para implementar la mejor solución para su entorno.

Hay tres áreas principales donde puede influir en la elegibilidad de las ofertas: la generación de la lista de ofertas candidatos, la determinación de la puntuación de marketing y el aprendizaje.

Generación de una lista de ofertas candidatas

La generación de una lista de ofertas candidatas tiene dos etapas principales. La primera etapa es la generación de una lista de todas las posibles ofertas para las que se puede elegir el cliente. La segunda etapa filtra las ofertas para las que ya no se puede elegir el cliente. Hay varios lugares en las dos etapas donde puede influir en la generación de la lista de ofertas candidatas.

Este diagrama muestra las etapas de la generación de una lista de ofertas candidatas. Las flechas indican el orden de preferencia. Por ejemplo, si una oferta pasa el filtro **Número máximo de veces que se presenta una oferta**, pero no pasa el filtro **Reglas de inclusión de ofertas globales**, el entorno de ejecución excluye la oferta.

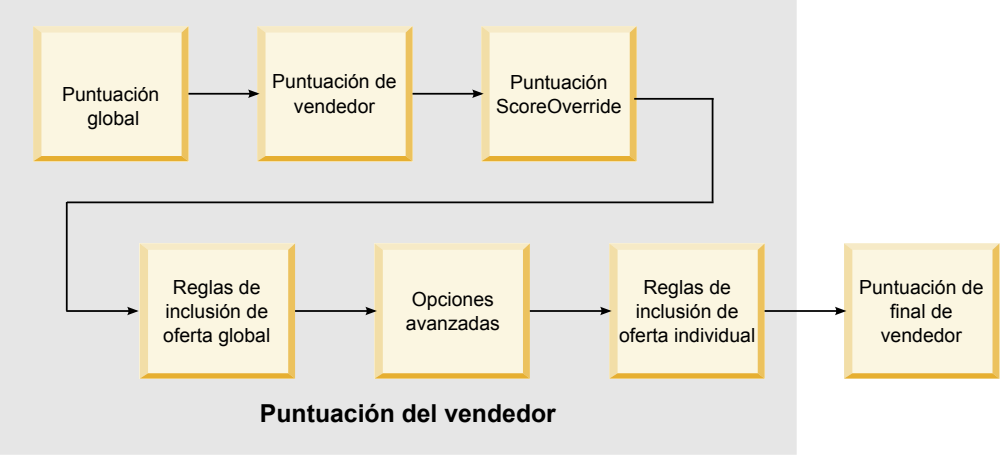


- **Asignaciones de ofertas globales**: puede definir ofertas globales por nivel de audiencia utilizando la tabla de ofertas globales.
- **Reglas de tratamiento**: el método básico para definir ofertas por segmento y por punto de interacción utilizando la pestaña de estrategia de interacción.
- **Asignaciones de ofertas individuales**: puede definir asignaciones de ofertas específicas por cliente utilizando la tabla de anulación de puntuaciones.
- **Fecha de caducidad de la oferta**: cuando crea una oferta en Campaign, puede definir una fecha de caducidad. Si pasa la fecha de caducidad de una oferta, el entorno de ejecución excluye la oferta.
- **Fecha de inicio y finalización de campaña**: cuando crea una campaña en Campaign, puede definir una fecha de inicio y de finalización para la campaña. Si no llega la fecha de inicio de la campaña o si pasa la fecha de finalización, el entorno de ejecución excluye la oferta.
- **Supresión de ofertas**: puede definir la supresión de ofertas para miembros específicos de la audiencia utilizando la tabla de supresión de ofertas.
- **Número máximo de veces que se presenta una oferta**: cuando se define un canal interactivo, debe definir el número máximo de veces que se presenta una oferta a un cliente por sesión. Si la oferta ya se ha presentado este número de veces, el entorno de ejecución excluye la oferta.
- **Reglas de inclusión de ofertas globales**: puede definir una expresión booleana para filtrar las ofertas en un nivel de audiencia utilizando la tabla de ofertas globales. Si el resultado es false, el entorno de ejecución excluye la oferta.
- **Opciones avanzadas**: puede utilizar la opción avanzada **Considerar que esta regla se puede elegir si la siguiente expresión es verdadera** en una regla de tratamiento para filtrar las ofertas en un nivel de segmento. Si el resultado es false, el entorno de ejecución excluye la oferta.
- **Reglas de inclusión de ofertas individuales**: puede definir una expresión booleana para filtrar las ofertas en un nivel de cliente utilizando la tabla de anulación de puntuaciones. Si el resultado es false, el entorno de ejecución excluye la oferta.

Cálculo de la puntuación de marketing

Hay muchas formas de influir (mediante un cálculo) o alterar temporalmente la puntuación de marketing.

Este diagrama muestra las distintas etapas donde puede influir o alterar temporalmente la puntuación de marketing.



Las flechas indican el orden de preferencia. Por ejemplo, si define una expresión para determinar la puntuación de marketing en las opciones avanzadas de una regla de tratamiento y define una expresión en la tabla de anulación de puntuaciones, la expresión en la tabla de anulación de puntuaciones tiene prioridad.

- **Puntuación global**: puede definir una puntuación por nivel de audiencia utilizando la tabla de ofertas globales.
- **Puntuación del vendedor**: puede definir una puntuación por segmento utilizando la barra de desplazamiento en una regla de tratamiento.
- **Puntuación de anulación de puntuaciones**: puede definir una puntuación por cliente utilizando la tabla de anulación de puntuaciones.
- **Reglas de inclusión de oferta global**: puede definir una expresión que calcule una puntuación por nivel de audiencia utilizando la tabla de ofertas globales.
- **Opciones avanzadas**: puede definir una expresión que calcule una puntuación por segmento utilizando la opción avanzada **Usar la siguiente expresión como la puntuación de marketing** en una regla de tratamiento.
- **Reglas de inclusión de oferta de anulación de puntuaciones**: puede definir una expresión que calcule una puntuación por cliente utilizando la tabla de anulación de puntuaciones.

Influencia en el aprendizaje

Si está utilizando el módulo de aprendizaje incorporado de Interact, puede influir en la salida de aprendizaje más allá de las configuraciones de aprendizaje estándar como, por ejemplo, la lista de atributos de aprendizaje o el nivel de confianza. Puede alterar temporalmente los componentes del algoritmo de aprendizaje mientras utiliza el resto de componentes.

Puede alterar temporalmente de aprendizaje utilizando las columnas `LikelihoodScore` y `AdjExploreScore` de las tablas de ofertas predeterminadas y de anulación de puntuaciones. Puede añadir estas columnas a las tablas de ofertas predeterminadas y de anulación de puntuaciones utilizando el script de característica `aci_scoringfeature`. Para utilizar correctamente estas alteraciones temporales, necesita un conocimiento avanzado del aprendizaje incorporado de Interact.

El módulo de aprendizaje utiliza en los cálculos finales la lista de ofertas candidatas y la puntuación de marketing por oferta candidata. La lista de ofertas

se utiliza con los atributos de aprendizaje para calcular la probabilidad (probabilidad de aceptación) de que el cliente acepte la oferta. Utilizando estas probabilidades y el número histórico de presentaciones para conseguir un equilibrio entre la exploración y la explotación, el algoritmo de aprendizaje determina la ponderación de la oferta. Por último, el aprendizaje incorporado multiplica la ponderación de la oferta por la puntuación de marketing final y devuelve una puntuación final. Las ofertas se clasifican por esta puntuación final.

Supresión de ofertas

Puede configurar el entorno de ejecución para suprimir ofertas.

Existen varias formas de que el entorno de ejecución suprima una oferta:

- El elemento **Núm. máximo de veces para mostrar ofertas durante una única visita** de un canal interactivo.

  El **Núm. máximo de veces para mostrar ofertas durante una única visita** se define cuando crea o edita un canal interactivo.
- El uso de una tabla de supresión de ofertas.

  La tabla de supresión de ofertas se crea en la base de datos de perfil.
- Ofertas cuya fecha de caducidad ha pasado.
- Ofertas de campañas caducadas.
- Ofertas excluidas porque no pasan una regla de inclusión de oferta (opción avanzada de regla de tratamiento).
- Ofertas aceptadas o rechazadas explícitamente en una sesión de Interact. Si un cliente acepta o rechaza explícitamente una oferta, dicha oferta se suprime durante la sesión.

Habilitación de la supresión de ofertas

Utilice este procedimiento para habilitar la supresión de ofertas.

Acerca de esta tarea

Puede configurar Interact para que haga referencia a una lista de ofertas suprimidas.

Procedimiento

1. Cree una `offerSuppressionTable`, una nueva tabla para cada audiencia que contiene el ID de audiencia y el ID de oferta.
2. Establezca la propiedad `enableOfferSuppressionLookup` en **true**.
3. Establezca la propiedad `Interact > profile > offerSuppressionTable` en el nombre de la tabla de supresión de ofertas para la audiencia correspondiente.

Tabla de supresión de ofertas

La tabla de supresión de ofertas permite suprimir una oferta para un ID de audiencia específico. Por ejemplo, si su audiencia es Customer, puede suprimir una oferta para el cliente John Smith. Debe existir una versión de esta tabla para al menos un nivel de audiencia en la base de datos de perfil. Puede crear una tabla de supresión de ofertas de muestra, `UACI_BlackList`, ejecutando el script SQL `aci_usertab` en la base de datos de perfil. El script SQL `aci_usertab` se encuentra en el directorio `ddl` del directorio de instalación del entorno de ejecución.

Debe definir los campos AudienceID y OfferCode1 para cada fila. Puede añadir columnas adicionales si el ID de oferta o el código de oferta consta de varias columnas. Estas columnas deben coincidir con los nombres de columna definidos en Campaign. Por ejemplo, si define la audiencia Customer por los campos HHold_ID y MemberNum, debe añadir HHold_ID y MemberNum a la tabla de supresión de ofertas.

Nombre	Descripción
AudienceID	(Necesario) El nombre de esta columna debe coincidir con el nombre de la columna que define el ID de audiencia en Campaign. Si el ID de audiencia está formado por varias columnas, puede añadirlas a esta tabla. Cada fila debe contener el ID de audiencia al que asigna la oferta predeterminada, por ejemplo, customer1.
CódigoOferta1	(Necesario) El código de oferta para la oferta que está alterando temporalmente. Si los códigos de oferta están formados por varios campos, puede añadir las columnas adicionales, por ejemplo, OfferCode2, etc.

Ofertas globales y asignaciones individuales

Puede configurar el entorno de ejecución para asignar ofertas específicas más allá de las reglas de tratamiento configuradas en la pestaña de estrategia de interacción. Puede definir ofertas globales para cualquier miembro de un nivel de audiencia y asignaciones individuales para miembros de audiencia específicos. Por ejemplo, puede definir una oferta global para que la vean todas las unidades familiares cuando no haya otras disponibles y, a continuación, crear una asignación de oferta individual para la unidad familiar específica Smith.

Puede restringir las ofertas globales y las asignaciones individuales por zona, celda y reglas de inclusión de ofertas. Las ofertas globales y las asignaciones individuales se configuran mediante la adición de datos a tablas específicas en la base de datos de perfil de producción.

Para que las ofertas globales y las asignaciones individuales funcionen correctamente, deben existir todos los códigos de oferta y celda referenciados en el despliegue. Para garantizar que los datos necesarios estén disponibles, debe configurar códigos de celda predeterminados y la tabla UACI_ICBatchOffers.

Definición de códigos de celda predeterminados

Si utiliza las tablas de ofertas predeterminadas o de anulación de puntuaciones para las asignaciones de ofertas globales o individuales, debe definir los códigos de celda predeterminados. El DefaultCellCode se utiliza cuando no hay ningún código de celda definido en una determinada fila en las tablas de ofertas predeterminadas o de anulación de puntuaciones. Los informes utilizan este código de celda predeterminado.

Acerca de esta tarea

El DefaultCellCode debe coincidir con el formato de código de celda definido en Campaign. Este código de celda se utiliza para todas las asignaciones de oferta que aparecen en los informes.

Si define códigos de celda predeterminados exclusivos, puede identificar fácilmente las ofertas asignadas por las tablas de ofertas predeterminadas o de anulación de puntuaciones.

Procedimiento

Defina la propiedad `DefaultCellCode` para cada nivel de audiencia y tipo de tabla en la categoría `IndividualTreatment`.

Definición de las ofertas no utilizadas en una regla de tratamiento

Si utiliza las tablas de ofertas predeterminadas o de anulación de puntuaciones, asegúrese de que todos los códigos de oferta existan en el despliegue. Si sabe que todas las ofertas que se utilizan en las tablas de ofertas predeterminadas o de anulación de puntuaciones se utilizan en las reglas de tratamiento, las ofertas existen en el despliegue. No obstante, una oferta que no se utilice en una regla de tratamiento debe definirse en la tabla `UACI_ICBatchOffers`.

Acerca de esta tarea

La tabla `UACI_ICBatchOffers` existe en las tablas del sistema de Campaign.

Procedimiento

Rellene la tabla `UACI_ICBatchOffers` con los códigos de oferta que utiliza en las tablas de ofertas predeterminadas o de anulación de puntuaciones. La tabla tiene el siguiente formato:

Nombre de columna	Tipo	Descripción
ICName	varchar(64)	El nombre del canal interactivo con el que está asociada la oferta. Si utiliza la misma oferta con dos canales interactivos diferentes, debe proporcionar una fila para cada canal interactivo.
CódigoOferta1	varchar(64)	La primera parte del código de oferta.
OfferCode2	varchar(64)	La segunda parte del código de oferta.
OfferCode3	varchar(64)	La tercera parte del código de oferta.
OfferCode4	varchar(64)	La cuarta parte del código de oferta.
OfferCode5	varchar(64)	La quinta parte del código de oferta.

Acerca de la tabla de ofertas globales

La tabla de ofertas globales permite definir tratamientos en el nivel de audiencia. Por ejemplo, puede definir una oferta global para cada miembro de la audiencia Unidad familiar.

Puede definir valores globales para los siguientes elementos de la presentación de ofertas de Interact.

- Asignación de ofertas globales
- Puntuación del usuario de marketing global, mediante un número o una expresión
- Expresión booleana para filtrar las ofertas
- Probabilidad y ponderación de aprendizaje, si utiliza el aprendizaje incorporado de Interact
- Alteración temporal de aprendizaje global

Asignación de ofertas globales

Utilice este procedimiento para configurar el entorno de ejecución para asignar ofertas globales para un nivel de audiencia, aparte de lo definido en las reglas de tratamiento.

Procedimiento

1. Cree una tabla llamada `UACI_DefaultOffers` en la base de datos de perfil.

   Para crear la tabla `UACI_DefaultOffers` con las columnas correctas, utilice el archivo ddl `aci_usrtab`.
2. Establezca la propiedad `enableDefaultOfferLookup` en **true**.

Tabla de ofertas globales

La tabla de ofertas globales debe existir en la base de datos de perfil. Puede crear la tabla de ofertas globales, `UACI_DefaultOffers` ejecutando el script SQL `aci_usertab` en la base de datos de perfil.

El script SQL `aci_usertab` se encuentra en el directorio `ddl` del directorio de instalación del entorno de ejecución.

Debe definir los campos `NivelAudiencia` y `CódigoOferta1` para cada fila. Los otros campos son opcionales para restringir adicionalmente las asignaciones de oferta o influenciar el aprendizaje incorporado en el nivel de audiencia.

Para obtener el mejor rendimiento, debe crear un índice en esta tabla en la columna de nivel de audiencia.

Name	Tipo	Descripción
AudienceLevel	varchar(64)	(Necesario) Nombre del nivel de audiencia al que se asigna la oferta predeterminada, por ejemplo, `cliente` o `unidad familiar`. Este nombre debe coincidir con el nivel de audiencia tal como está definido en Campaign.
OfferCode1	varchar(64)	(Necesario) Código de oferta de la oferta predeterminada. Si los códigos de oferta se componen de varios campos, puede añadir columnas adicionales, por ejemplo, `CódigoOferta2` y así sucesivamente Si está añadiendo esta oferta para proporcionar una asignación de oferta global, debe añadir esta oferta a la tabla `UACI_ICBatchOffers`.
Score	float	Número para definir la puntuación de marketing de esta asignación de oferta.
OverrideTypeID	int	Si se establece en `1`, si la oferta no existe en la lista candidata de ofertas, añada esta oferta a la lista y utilice los datos de puntuación para la oferta. Normalmente, utilice 1 para proporcionar asignaciones de oferta global. Si se establece en `0`, *nulo* o cualquier otro número que no sea 1, utilice los datos de la oferta solo si la oferta existe en la lista candidata de ofertas. En la mayoría de los casos, una regla de tratamiento o asignación individual sustituirá este valor.

Name	Tipo	Descripción
Predicate	varchar(4000)	Puede especificar expresiones en esta columna de la misma forma que para las opciones avanzadas de reglas de tratamiento. Puede utilizar las mismas variables y macros que están disponibles al escribir las opciones avanzadas de las reglas de tratamiento. El comportamiento de esta columna depende del valor de la columna `HabilitarIDestado`. • Si `HabilitarIDestado` es 2, esta columna funciona de la misma forma que la opción **Considerar esta regla eligible si la siguiente expresión es true** en las opciones avanzadas de las reglas de tratamiento que restringen esta asignación de oferta. Esta columna debe contener una expresión booleana y resolverse en true para incluir esta oferta. Si define accidentalmente una expresión que se resuelve en un número, un número distinto de cero se considera true y cero se considera false. • Si `HabilitarIDestado` es 3, esta columna funciona de la misma forma que la opción **Utilizar la siguiente expresión como puntuación de marketing** de las opciones avanzadas de las reglas de tratamiento para restringir esta oferta. Esta columna debe contener una expresión que se resuelva en un número. • Si `HabilitarIDestado` es 1, Interact ignora cualquier valor de esta columna.
FinalScore	float	Número para sustituir la puntuación final utilizada para ordenar la lista final de ofertas devueltas. Esta columna se utiliza si se habilitado el módulo de aprendizaje incorporado. Puede implementar su propio aprendizaje para utilizar esta columna.
CellCode	varchar(64)	Código de celda para un segmento interactivo al que desea asignar esta oferta predeterminada. Si los códigos de celda se componen de varios campos, puede añadir las columnas adicionales. Debe proporcionar un código de celda si `IdTipoAnulación` es 0 o nulo. Si no incluye un código de celda, el entorno de ejecución ignora esta fila de datos. Si `IdTipoAnulación` es 1, no necesita proporcionar un código de celda en esta columna. Si no proporciona un código de celda, el entorno de ejecución utiliza el código de celda definido en la propiedad `DefaultCellCode` para este nivel de audiencia y tabla para fines de creación de informes.
Zone	varchar(64)	Nombre de la zona a la que desea que se aplique esta asignación de oferta. Si es NULL, se aplicará a todas las zonas.

Name	Tipo	Descripción
EnableStateID	int	El valor de esta columna define el comportamiento de la columna `Predicado`. • **1**: no utilizar la columna `Predicado`. • **2**: utilizar `Predicado` como booleano para filtrar la oferta. Permite las mismas reglas que la opción avanzada **Considerar esta regla elegible si la siguiente expresión es true** de una regla de tratamiento. • **3**: utilizar `Predicado` para definir la puntuación del usuario de marketing. Sigue las mismas reglas que la opción avanzada **Utilizar la siguiente expresión como puntuación de marketing** de una regla de tratamiento. Cualquier fila donde esta columna sea nulo o cualquier valor que no sea 2 ó 3 ignora la columna `Predicado`.
Puntuación-Probabilidad	float	Esta columna se utiliza solo para influenciar al aprendizaje incorporado. Puede añadir esta columna con la DDL `aci_scoringfeature`.
Puntuación-ExplorarAdj	float	Esta columna se utiliza solo para influenciar al aprendizaje incorporado. Puede añadir esta columna con la DDL `aci_scoringfeature`.

Acerca de la tabla de anulación de puntuaciones

La tabla de anulación de puntuaciones permite definir los tratamientos en un ID de audiencia o un nivel individual. Por ejemplo, si el nivel de audiencia es Visitante, puede crear alteraciones temporales para visitantes específicos.

Puede definir alteraciones temporal para los siguientes elementos de la presentación de ofertas de Interact.

- Asignación de ofertas individuales
- Puntuación del usuario de marketing individual, mediante un número o una expresión
- Expresión booleana para filtrar las ofertas
- Probabilidad y ponderación de aprendizaje, si utiliza el aprendizaje incorporado
- Alteración temporal de aprendizaje individual

Configuración de las anulaciones de puntuación

Puede configurar Interact para utilizar una puntuación generada a partir de una aplicación de modelado en lugar de la puntuación de marketing.

Procedimiento

1. Cree una tabla de anulación de puntuaciones para cada nivel de audiencia para el que desea proporcionar alteraciones temporales.

   Para crear una tabla de anulación de puntuaciones de muestra con las columnas correctas, utilice el archivo ddl `aci_usrtab`.
2. Establezca la propiedad `enableScoreOverrideLookup` en **true**.
3. Establezca la propiedad `scoreOverrideTable` en el nombre de la tabla de anulación de puntuaciones para cada nivel de audiencia para el que desea proporcionar alteraciones temporales.

No es necesario proporcionar una tabla de anulación de puntuaciones para cada nivel de audiencia.

Tabla de sustituciones de puntuación

La tabla de sustituciones de puntuación debe existir en la base de datos de perfil de producción. Puede crear una tabla de sustituciones de puntuación de muestra, `UACI_ScoreOverride`, ejecutando el script SQL `aci_usertab` en la base de datos de perfil.

El script SQL `aci_usertab` se encuentra en el directorio `ddl` del directorio de instalación del entorno de ejecución.

Debe definir los campos *`IDAudiencia`*, `CódigoOferta1` y `Puntuación` para cada fila. Los valores de los demás campos son opcionales para restringir adicionalmente asignaciones de oferta individuales o proporcionar información de anulación de puntuaciones para aprendizaje incorporado.

Nombre	Tipo	Descripción
IDAudiencia	varchar(64)	(Necesario) El nombre de esta columna debe coincidir con el nombre de la columna que define el ID de audiencia en Campaign. La tabla de muestra creada por el archivo DLL `aci_usertab` crea esta columna como una columna `IDCliente`. Si el ID de audiencia consta de varias columnas, puede añadirlas a esta tabla. Cada fila debe contener el ID de audiencia al que asigna la oferta individual, por ejemplo, `cliente1`. Para obtener un mejor rendimiento, debe crear un índice en esta columna.
CódigoOferta1	varchar(64)	(Necesario) Código de oferta para la oferta. Si los códigos de oferta se componen de varios campos, puede añadir columnas adicionales, por ejemplo, `CódigoOferta2` y así sucesivamente Si está añadiendo esta oferta para proporcionar una asignación de oferta individual, debe añadir esta oferta a la tabla `UACI_ICBatchOffers`.
Puntuación	float	Número para definir la puntuación de marketing de esta asignación de oferta.
IdTipoAnulación	int	Si se establece `0` o *nulo* (o cualquier número distinto de 1), utilice datos de la oferta solo si la oferta existe en la lista candidata de ofertas. Normalmente, utilice `0` para proporcionar sustituciones de puntuación. Debe proporcionar un código de celda. Si se establece en `1`, si la oferta no existe en la lista candidata de ofertas, añada esta oferta a la lista y utilice los datos de puntuación para la oferta. Normalmente, utilice `1` para proporcionar asignaciones de oferta individuales.

Nombre	Tipo	Descripción
Predicado	varchar(4000)	Puede especificar expresiones en esta columna de la misma forma que para las opciones avanzadas de reglas de tratamiento. Puede utilizar las mismas variables y macros que están disponibles al escribir las opciones avanzadas de las reglas de tratamiento. El comportamiento de esta columna depende del valor de la columna `HabilitarIDestado`. • Si `HabilitarIDestado` es `2`, esta columna funciona de la misma forma que la opción **Considerar esta regla eligible si la siguiente expresión es true** en las opciones avanzadas de las reglas de tratamiento que restringen esta asignación de oferta. Esta columna debe contener una expresión booleana y resolverse en true para incluir esta oferta. Si define accidentalmente una expresión que se resuelve en un número, un número distinto de cero se considera true y cero se considera false. • Si `HabilitarIDestado` es `3`, esta columna funciona de la misma forma que la opción **Utilizar la siguiente expresión como puntuación de marketing** de las opciones avanzadas de las reglas de tratamiento para restringir esta oferta. Esta columna debe contener una expresión que se resuelva en un número. • Si `HabilitarIDestado` es `1`, Interact ignora cualquier valor de esta columna.
PuntuaciónFinal	float	Número para sustituir la puntuación final utilizada para ordenar la lista final de ofertas devueltas. Esta columna se utiliza si se habilitado el módulo de aprendizaje incorporado. Puede implementar su propio aprendizaje para utilizar esta columna.
Códigodecelda	varchar(64)	Código de celda de un segmento interactivo al que desea asignar esta oferta. Si los códigos de celda se componen de varios campos, puede añadir las columnas adicionales. Debe proporcionar un código de celda si `IdTipoAnulación` es `0` o nulo. Si no incluye un código de celda, el entorno de ejecución ignora esta fila de datos. Si `IdTipoAnulación` es `1`, no necesita proporcionar un código de celda en esta columna. Si no proporciona un código de celda, el entorno de ejecución utiliza el código de celda definido en la propiedad `DefaultCellCode` para este nivel de audiencia y tabla para fines de creación de informes.
Zona	varchar(64)	Nombre de la zona a la que desea que se aplique esta asignación de oferta. Si es NULL, se aplicará a todas las zonas.

Nombre	Tipo	Descripción
HabilitarIDestado	int	El valor de esta columna define el comportamiento de la columna `Predicado`. • **1**: no utilizar la columna `Predicado`. • **2**: utilizar `Predicado` como booleano para filtrar la oferta. Permite las mismas reglas que la opción avanzada **Considerar esta regla elegible si la siguiente expresión es true** de una regla de tratamiento. • **3**: utilizar `Predicado` para definir la puntuación del usuario de marketing. Sigue las mismas reglas que la opción avanzada **Utilizar la siguiente expresión como puntuación de marketing** de una regla de tratamiento. Cualquier fila donde esta columna sea nulo o cualquier valor que no sea 2 ó 3 ignora la columna `Predicado`.
Puntuación-Probabilidad	float	Esta columna se utiliza solo para influenciar al aprendizaje incorporado. Puede añadir esta columna con la DDL `aci_scoringfeature`.
Puntuación-ExplorarAdj	float	Esta columna se utiliza solo para influenciar al aprendizaje incorporado. Puede añadir esta columna con la DDL `aci_scoringfeature`.

Descripción general del aprendizaje incorporado de Interact

Aunque haga todo lo posible para asegurarse de que propone las ofertas correctas a los segmentos correctos, siempre puede aprender algo de las propias selecciones de los visitantes. El comportamiento real de los visitantes debe influir en su estrategia. Puede ejecutar el historial de respuestas utilizando herramientas de modelado para obtener una puntuación e incluirla en los diagramas de flujo interactivos.

No obstante, estos datos no son en tiempo real.

Interact ofrece dos opciones para aprender de las acciones del visitante en tiempo real:

- Módulo de aprendizaje incorporado: el entorno de ejecución tiene un módulo de aprendizaje basado en Naive Bayesian. Este módulo supervisa los atributos de los clientes que elija y utiliza esos datos para ayudarle a seleccionar qué ofertas debe presentar.
- API de aprendizaje: el entorno de ejecución también tiene una API de aprendizaje para que escriba su propio módulo de aprendizaje.

No es obligatorio utilizar el aprendizaje. De forma predeterminada, el aprendizaje está inhabilitado.

Módulo de aprendizaje de Interact

El módulo de aprendizaje de Interact supervisa las respuestas del visitante a las ofertas y los atributos de visitante.

Modos del módulo de aprendizaje

El módulo de aprendizaje tiene dos modos generales:

- Exploración: el módulo de aprendizaje sirve las ofertas para poder recopilar suficientes datos de respuesta para optimizar el cálculo que se utiliza durante el modo de explotación. Las ofertas servidas durante la exploración no reflejan necesariamente la opción óptima.
- Explotación: una vez recopilados suficientes datos en la fase de exploración, el módulo de aprendizaje utiliza las probabilidades para ayudarle a seleccionar las ofertas que se presentan.

El módulo de aprendizaje utiliza dos propiedades para alternar entre el modo de exploración y el modo de explotación. Las dos propiedades son:

- un nivel de confianza que configura con la propiedad `confidenceLevel`.
- una probabilidad de que el módulo de aprendizaje presente una oferta aleatoria que configura con la propiedad `percentRandomSelection`.

Propiedad de nivel de confianza

Debe establecer `confidenceLevel` en un porcentaje que representa el grado de seguridad (o confianza) que debe tener el módulo de aprendizaje para que se utilicen sus puntuaciones de una oferta en el arbitraje. Al principio, cuando el módulo de aprendizaje no tiene datos a partir de los que trabajar, depende totalmente de la puntuación de marketing. Cuando cada oferta se ha presentado tantas veces como se define en `minPresentCountThreshold`, el módulo de aprendizaje entra en el modo de exploración. Sin una gran cantidad de datos con los que trabajar, el módulo de aprendizaje no está seguro de que los porcentajes que calcula son correctos. Por lo tanto, permanece en el modo de exploración.

El módulo de aprendizaje asigna ponderaciones a cada oferta. Para calcular las ponderaciones, el módulo de aprendizaje utiliza una fórmula que toma como entrada el nivel de confianza configurado, los datos de aceptación históricos y los datos de la sesión actual. La fórmula establece un equilibrio intrínseco entre la exploración y explotación, y devuelve la ponderación adecuada.

Propiedad de selección aleatoria

Para garantizar que el sistema favorece las ofertas con un mayor rendimiento durante las primeras etapas, Interact presenta una oferta aleatoria el `percentRandomSelection` por ciento del tiempo. Este porcentaje de oferta aleatoria obliga al módulo de aprendizaje a recomendar ofertas que no sean las más satisfactorias para determinar si las otras ofertas tendrían un mayor rendimiento si estuvieran más expuestas. Por ejemplo, si configura `percentRandomSelection` en 5, el 5% del tiempo el módulo de aprendizaje presenta una oferta aleatoria y añade los datos de respuesta a sus cálculos.

Puede establecer **% Random** para especificar que el cambio que ha devuelto la oferta se selecciona aleatoriamente, sin tener en cuenta puntuaciones, para cada zona en la pestaña Puntos de interacción de la ventana Canal interactivo.

Cómo determina las ofertas el módulo de aprendizaje

El módulo de aprendizaje determina qué ofertas se presentan de la siguiente forma.

1. Calcula la probabilidad de que un visitante seleccione una oferta.

2. Calcula la ponderación de la oferta basándose en la probabilidad del paso 1 y determina si debe estar en modo de exploración o explotación.
3. Calcula una puntuación final para cada oferta utilizando la puntuación de marketing y la ponderación de oferta del paso 2.
4. Clasifica las ofertas según las puntuaciones determinadas en el paso 3 y devuelve el número solicitado de ofertas principales.

Por ejemplo, el módulo de aprendizaje determina que un visitante tiene un 30% de probabilidades de aceptar una oferta A y un 70% de probabilidades de aceptar una oferta B y explotar esta información. A partir de las reglas de tratamiento, la puntuación de marketing de la oferta A es 75 y 55 para la oferta B. Sin embargo, los cálculos del paso 3 convierten la puntuación final de la oferta B en mayor que la de la oferta A. Por lo tanto, el entorno de ejecución recomienda la oferta B.

Propiedades de factor de ponderación

El aprendizaje también se basa en la propiedad `recencyWeightingFactor` y en la propiedad `recencyWeightingPeriod`. Estas propiedades permiten aumentar la ponderación de los datos más recientes. `recencyWeightingFactor` es el porcentaje de ponderación que se otorga a los datos recientes. `recencyWeightingPeriod` es el período de tiempo que se considera reciente. Por ejemplo, puede configurar `recencyWeightingFactor` en 0,30 y `recencyWeightingPeriod` en 24. Estos valores significan que las últimas 24 horas de datos, se pondera el 30% de todos los datos. Para el equivalente a una semana de datos, todos los datos promediados en los primeros seis días constituyen el 70% de los datos y en el último día constituyen el 30% de los datos.

Datos escritos de la tabla de preparación

Cada sesión escribe los siguientes datos en una tabla de preparación de aprendizaje:

- Contacto de oferta
- Aceptación de la oferta
- Atributos de aprendizaje

En un intervalo configurable, un agregador lee los datos de la tabla de preparación, los compila y los escribe en una tabla. El módulo de aprendizaje lee estos datos agregados y los utiliza en los cálculos.

Habilitación del módulo de aprendizaje

Todos los servidores de ejecución tienen un módulo de aprendizaje incorporado. De forma predeterminada, este módulo de aprendizaje está inhabilitado. Para habilitar el módulo de aprendizaje, cambie una propiedad de configuración.

Procedimiento

En la Marketing Platform del entorno de ejecución, edite las siguientes propiedades de configuración en la categoría `Interact > offerserving`.

Propiedad de configuración	Valor
`optimizationType`	**BuiltInLearning**

Atributos de aprendizaje

El módulo de aprendizaje aprende utilizando los atributos de visitante y los datos de aceptación de ofertas. Puede seleccionar qué atributos de visitante desea supervisar. Estos atributos de visitante pueden ser cualquier elemento en un perfil de cliente, por ejemplo, un parámetro de evento que se recopila en tiempo real.

Los atributos de las tablas dimensionales no están soportados en el aprendizaje.

Aunque puede configurar el número de atributos que desee para supervisarlos, IBM recomienda no configurar más de diez atributos de aprendizaje entre los atributos de aprendizaje estáticos y dinámicos, y seguir estas directrices.

- Seleccione atributos independientes.

  No seleccione atributos que sean similares. Por ejemplo, si crea un atributo denominado HighValue y este atributo se define mediante un cálculo basado en el salario, no seleccione HighValue y Salary. Los atributos similares no ayudan al algoritmo de aprendizaje.
- Seleccione atributos con valores discretos.

  Si un atributo tiene rangos de valores, debe seleccionar un valor exacto. Por ejemplo, si desea utilizar el salario como un atributo, debe dar a cada rango de salarios un valor específico, por ejemplo, el rango 20.000-30.000 debe ser A, 30.001-40.000 debe ser B, etc.
- Limite el número de atributos a los que realiza un seguimiento para no disminuir el rendimiento.

  El número de atributos que puede rastrear depende de los requisitos de rendimiento y de la instalación de Interact. Si puede, utilice otra herramienta de modelado (por ejemplo, PredictiveInsight) para determinar los diez primeros atributos predictivos. Puede configurar el módulo de aprendizaje para podar automáticamente los atributos que no sean predictivos, pero esto también afecta al rendimiento.

Puede gestionar el rendimiento definiendo el número de atributos que desea supervisar y el número de valores por atributo que se van a supervisar. La propiedad `maxAttributeNames` define el número máximo de atributos de visitante a los que puede realizar un seguimiento. La propiedad `maxAttributeValues` define el número máximo de valores que puede rastrear por atributo. Los demás valores se asignan a una categoría definida por el valor de la propiedad `otherAttributeValue`. Sin embargo, el motor de aprendizaje sólo realiza un seguimiento de los primeros valores que encuentra. Por ejemplo, supongamos que realiza un seguimiento del atributo de visitante de color de ojos. Sólo está interesado en los valores azul, marrón y verde, por lo que establece `maxAttributeValues` en 3. Sin embargo, los tres primeros visitantes tiene los valores azul, marrón y avellana. Esto significa que a todos los visitantes con los ojos verdes se les asigna `otherAttributeValue`.

También puede utilizar atributos de aprendizaje dinámicos que permiten definir los criterios de aprendizaje más específicamente. Los atributos de aprendizaje dinámicos permiten aprender de la combinación de dos atributos como una sola entrada. Por ejemplo, supongamos la siguiente información de perfil.

ID del visitante	Tipo de tarjeta	Saldo de tarjeta
1	Tarjeta Oro	$1.000
2	Tarjeta Oro	$9.000
3	Tarjeta Bronce	$1.000

ID del visitante	Tipo de tarjeta	Saldo de tarjeta
4	Tarjeta Bronce	$9.000

Si utiliza los atributos de aprendizaje estándar, sólo puede aprender del tipo de tarjeta y el saldo individualmente. Los visitantes 1 y 2 se agruparán basándose en el Tipo de tarjeta, y los visitantes 2 y 4 se agruparán basándose en el Saldo de tarjeta. Este puede que no sea un indicador preciso de un comportamiento de aceptación de oferta. Si los titulares de la tarjeta Oro tienden a tener saldos mayores, el comportamiento del visitante 2 puede ser radicalmente distinto al del visitante 4, lo que puede sesgar los atributos de aprendizaje estándar. Sin embargo, si se utilizan atributos de aprendizaje dinámicos, se aprende sobre cada uno de estos visitantes individualmente y las predicciones serán más precisas.

Si utiliza atributos de aprendizaje dinámicos y el visitante tiene dos valores válidos para un atributo, el módulo de aprendizaje selecciona el primer valor que encuentra.

Si establece la propiedad `enablePruning` en `yes`, el módulo de aprendizaje determina mediante algoritmos qué atributos no son predictivos y deja de tener en cuenta esos atributos cuando calcula las ponderaciones. Por ejemplo, si realiza un seguimiento de un atributo que representa el color de cabello y el módulo de aprendizaje determina que no existe ningún patrón para aceptar una oferta según el color de cabello del visitante, el módulo de aprendizaje deja de tener en cuenta el atributo de color de cabello. Los atributos se vuelven a evaluar cada vez que se ejecuta el proceso de agregación de aprendizaje (definido por la propiedad `aggregateStatsIntervalInMinutes`). Los atributos de aprendizaje dinámicos también se podan.

Definición de un atributo de aprendizaje

Utilice este procedimiento para definir un atributo de aprendizaje.

Acerca de esta tarea

Puede configurar atributos de visitantes hasta el número especificado en `maxAttributeNames`.

`(learningAttributes)` es una plantilla para crear nuevos atributos de aprendizaje. Debe especificar un nuevo nombre para cada atributo. No puede crear dos categorías con el mismo nombre.

Procedimiento

En la Marketing Platform del entorno de diseño, edite las siguientes propiedades de configuración en la categoría `Campaign > partitions > partition`*`n`* `> Interact > learning`.

Propiedad de configuración	Valor
`attributeName`	`attributeName` debe coincidir con el nombre de un par nombre-valor en los datos del perfil. Este nombre es sensible a mayúsculas y minúsculas.

Definición de atributos de aprendizaje dinámicos

Para definir los atributos de aprendizaje dinámicos, debe completar la tabla `UACI_AttributeList` del origen de datos Aprendizaje.

Todas las columnas de esta tabla tienen el tipo de varchar(64).

Columna	Descripción
`AttributeName`	Nombre del atributo sobre el que desea obtener información. Debe ser un valor real posible en `AttributeNameCol`.
`AttributeNameCol`	Nombre de columna completo (estructura jerárquica, empezando desde la tabla de perfil) donde se puede encontrar `AttributeName`. No es necesario que este nombre de columna sea el atributo de aprendizaje estándar.
`AttributeValueCol`	Nombre de columna completo (estructura jerárquica, empezando desde la tabla de perfil) donde se puede encontrar el valor asociado para `AttributeName`.

Por ejemplo, considere la siguiente tabla de perfil y su tabla de dimensiones asociada.

Tabla 6. MyProfileTable

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

Tabla 7. MyDimensionTable

KeyField	CardType	CardBalance
Key1	Tarjeta Oro	1000
Key2	Tarjeta Oro	9000
Key3	Tarjeta Bronce	1000
Key4	Tarjeta Bronce	9000

A continuación se muestra una tabla `UACI_AttributeList` de muestra que se correlaciona según el tipo de tarjeta y el saldo.

Tabla 8. UACI_AttributeList

AttributeName	AttributeNameCol	AttributeValueCol
Tarjeta Oro	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance
Tarjeta Bronce	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance

Configuración del entorno de ejecución para reconocer módulos de aprendizaje externos

Puede utilizar la API Java™ de aprendizaje para escribir su propio módulo de aprendizaje. Debe configurar el entorno de ejecución para que reconozca su utilidad de aprendizaje en la Marketing Platform.

Acerca de esta tarea

Debe reiniciar el servidor de ejecución de Interact para que estos cambios entren en vigor.

Procedimiento

1. En la Marketing Platform del entorno de ejecución, edite las siguientes propiedades de configuración en la categoría `Interact > offerserving`. Las propiedades de configuración de la API del optimizador de aprendizaje se encuentran en la categoría `Interact > offerserving > External Learning Config`.

Propiedad de configuración	Valor
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Nombre de clase del aprendizaje externo
`externalLearningClassPath`	La ruta de la clase o los archivos JAR en el servidor de ejecución para el aprendizaje externo. Si utiliza un grupo de servidores y todos los servidores de ejecución hacen referencia a la misma instancia de Marketing Platform, cada servidor debe tener una copia de la clase o los archivos JAR en la misma ubicación.

2. Reinicie el servidor de ejecución de Interact para que estos cambios entren en vigor.

Capítulo 5. Entender la API de Interact

Interact proporciona ofertas dinámicamente a una gran variedad de puntos de encuentro. Por ejemplo, puede configurar el entorno de ejecución y el punto de encuentro para enviar mensajes a los empleados del centro de atención al cliente informándoles sobre las mejores posibilidades de aumento de venta o venta cruzada para un cliente que ha llamado con un tipo específico de consulta de servicio. También puede configurar el entorno de ejecución y el punto de encuentro para proporcionar ofertas personalizadas a un cliente (visitante) que ha entrado en un área determinada del sitio web.

La interfaz de programación de aplicaciones (API) de Interact le permite configurar el punto de encuentro y un servidor de ejecución para que funcionen juntos para proporcionar las mejores ofertas posibles. Mediante la API, el punto de encuentro puede solicitar información del servidor de ejecución para asignar el visitante a un grupo (o segmento) y presentar ofertas basadas en ese segmento. También puede registrar datos para su posterior análisis para acotar las estrategias de presentación.

La API de Interact también permite la comunicación de cliente a servidor del usuario final a través de JavaScript.

Para proporcionarle la mayor flexibilidad posible en la integración de Interact con sus entornos, IBM proporciona un servicio web accesible utilizando la API de Interact.

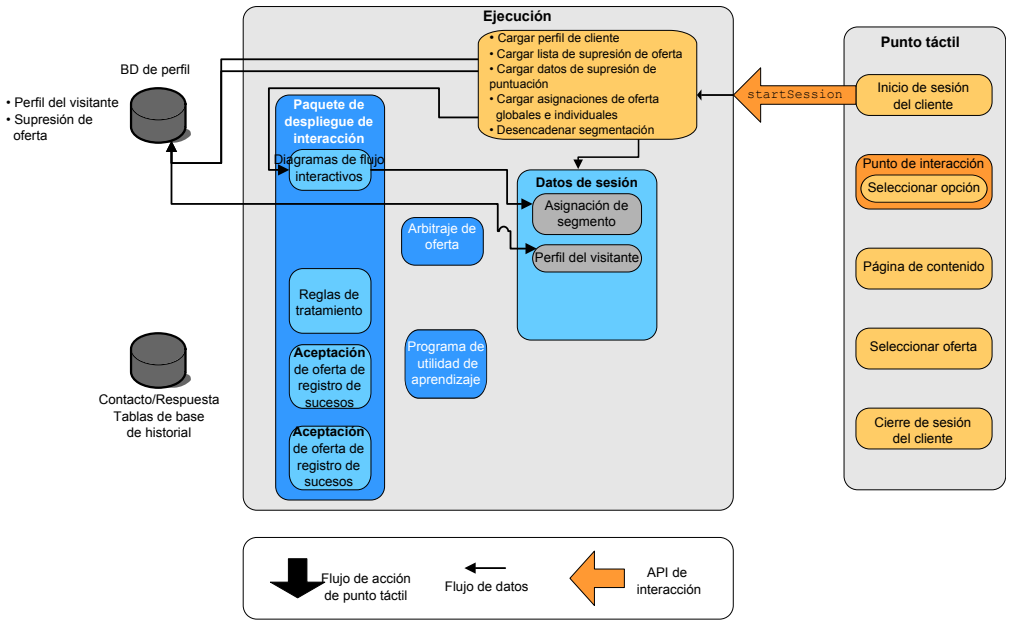
Flujo de datos de la API de Interact

Este ejemplo muestra cómo funciona la API entre el punto de encuentro y el entorno de ejecución. El visitante sólo realiza cuatro acciones: iniciar la sesión, ir a la página que muestra las ofertas, seleccionar una oferta y cerrar la sesión. Puede diseñar la integración para que sea lo complicada que desee, dentro de los límites de sus requisitos de rendimiento.

El diagrama muestra una implementación simple de la API de Interact.

Un visitante inicia una sesión en un sitio web y navega a una página que muestra las ofertas. El visitante selecciona una oferta y cierra la sesión. Mientras la interacción es simple, se producen varios eventos en el punto de encuentro y el servidor de ejecución:

1. Inicio de una sesión
2. Navegación a una página
3. Selección de una oferta
4. Cierre de la sesión



Inicio de la sesión

Cuando el visitante inicia la sesión, desencadena una `startSession`.

El método `startSession` realiza cuatro acciones:

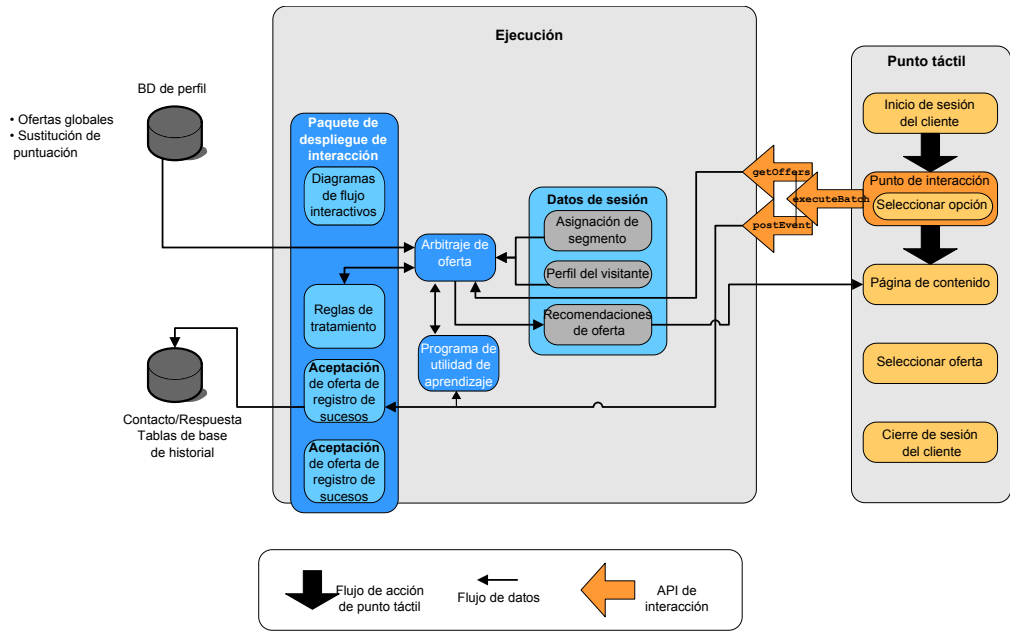
1. Crea una nueva sesión de ejecución
2. Envía una solicitud para los datos de perfil del cliente en la sesión
3. Envía una solicitud para utilizar los datos del perfil e iniciar un diagrama de flujo interactivo para incluir el cliente en segmentos. Esta ejecución de diagrama de flujo es asíncrona.
4. El servidor de ejecución carga la supresión de ofertas y la información de tratamiento de ofertas individuales y globales en la sesión. Los datos de sesión se mantienen en la memoria durante la sesión.

Navegación a una página

El visitante navega por el sitio hasta que llega a un punto de interacción predefinido. En la figura, el segundo punto de interacción (Seleccionar opción) es el lugar donde el visitante pulsa un enlace que presenta un conjunto de ofertas. El gestor de puntos de encuentro ha configurado el enlace para desencadenar un método `executeBatch` para seleccionar una oferta.

Selección de una oferta

Este diagrama muestra la llamada a la API que desencadena el método `executeBatch`.

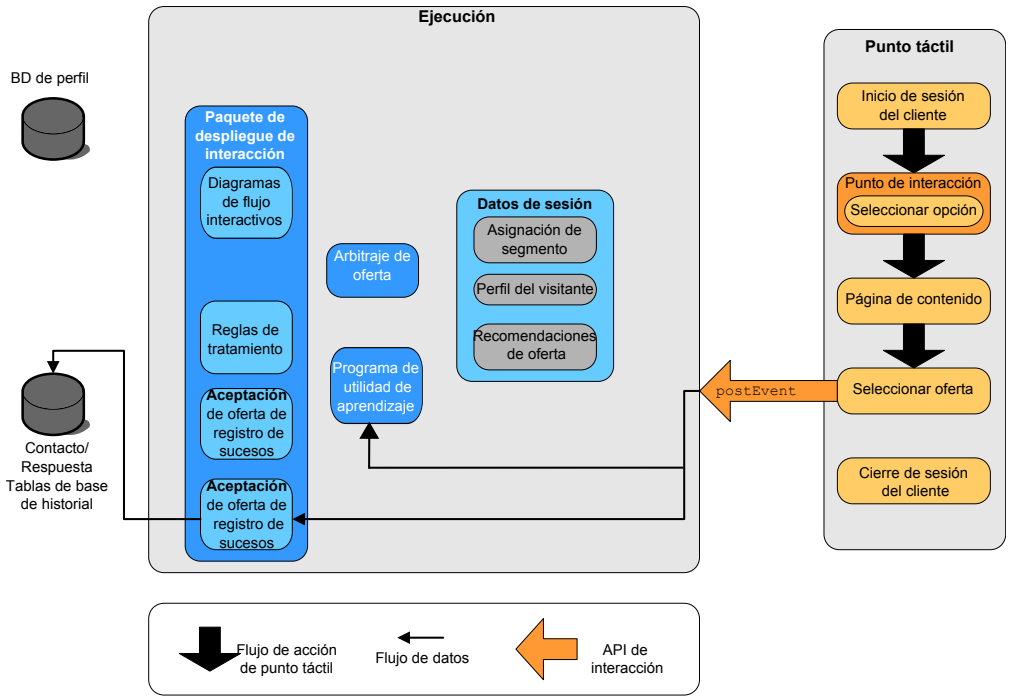


El método `executeBatch` permite llamar a más de un método en una sola llamada al servidor de ejecución. Estas llamadas a executeBatch llaman a dos otros métodos, `getOffers` y `postEvent`. El método `getOffers` solicita una lista de ofertas. El servidor de ejecución utiliza los datos de segmentación, la lista de supresión de ofertas, las reglas de tratamiento y el módulo de aprendizaje para proponer un conjunto de ofertas. El servidor de ejecución devuelve un conjunto de ofertas que se muestran en la página de contenidos.

El método `postEvent` desencadena uno de los eventos definidos en el entorno de diseño. En este caso concreto, el evento envía una solicitud para registrar las ofertas presentadas al historial de contactos.

El visitante selecciona una de las ofertas (Seleccionar oferta).

Este diagrama muestra el método `postEvent`.

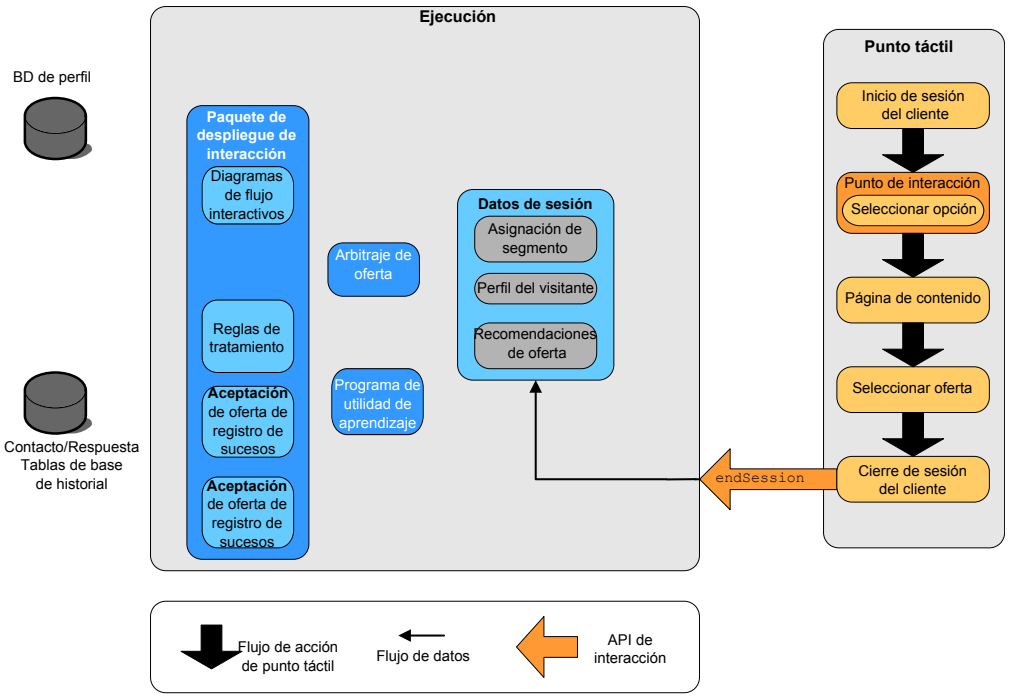


El control de interfaz de usuario que está asociado con la selección de la oferta se configura para enviar otro método `postEvent`. Este evento envía una solicitud para registrar la aceptación de la oferta en el historial de respuestas.

Cierre de la sesión

Una vez que el visitante selecciona la oferta, termina con el sitio web y cierra la sesión. El comando de cerrar sesión está enlazado con el método `endSession`.

Este diagrama muestra el método `endSession`.



El método `endSession` cierra la sesión. Si el visitante olvida cerrar la sesión, existe un tiempo de espera de sesión configurable para garantizar que finalicen todas las sesiones al final. Si desea mantener algunos de los datos pasados a la sesión como, por ejemplo, la información incluida en los parámetros en los métodos `startSession` o `setAudience`, trabaje con la persona que crea los diagramas de flujo interactivos. La persona que crea un diagrama de flujo interactivo puede utilizar el proceso Instantánea para escribir esos datos en una base de datos antes de la sesión finalice y se pierdan los datos. A continuación, puede utilizar el método `postEvent` para llamar al diagrama de flujo interactivo que contiene el proceso Instantánea.

Ejemplo de planificación de interacción simple

En este ejemplo, está diseñando una interacción para el sitio web de una empresa de teléfonos móviles. Crea tres ofertas diferentes, configura el registro de las ofertas, asigna códigos de tratamiento a las ofertas y muestra una serie de imágenes que se enlazan con las ofertas.

Proceso de diseño

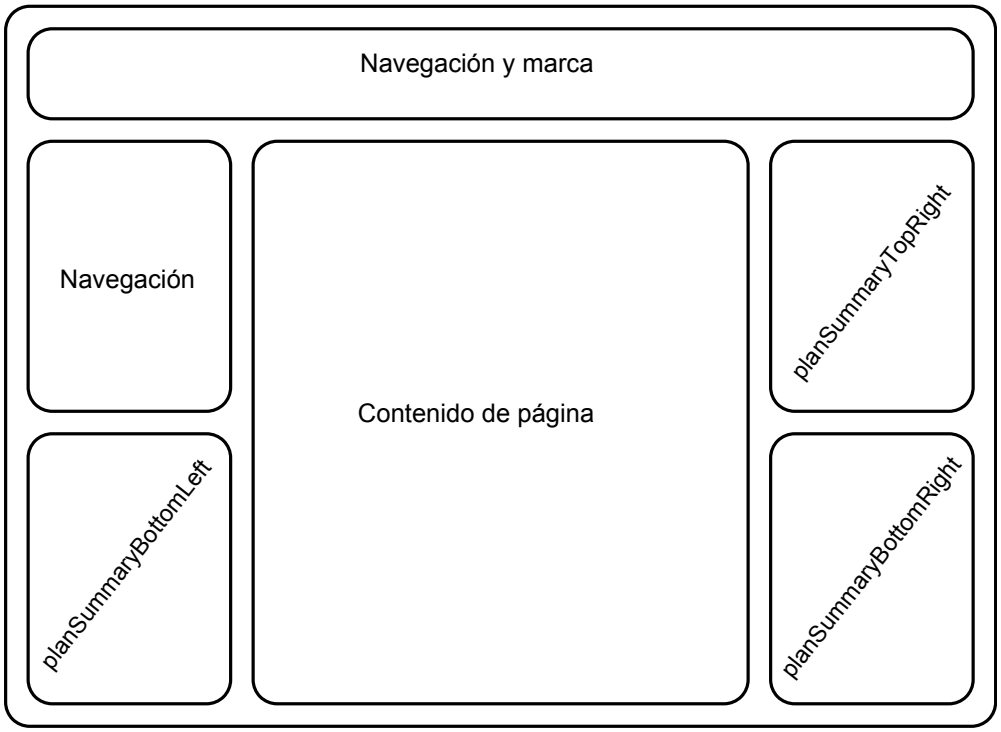
Para diseñar una interacción para este cliente, debe:

1. Identificar los requisitos de la página de resumen del cliente
2. Crear puntos de interacción para los requisitos de oferta
3. Configurar el registro de las ofertas
4. Crear códigos de tratamiento
5. Enlazar una serie de imágenes rotativas a las ofertas

Es un ejemplo básico que no muestra la mejor forma de escribir la integración. Por ejemplo, ninguno de estos ejemplos incluye la comprobación de error que utiliza la clase de respuesta.

Identificación de los requisitos de la página de resumen del plan de teléfonos móviles

El siguiente diagrama muestra el diseño de la página de resumen del plan de teléfonos móviles.



Defina los siguientes elementos para cumplir los requisitos de la página de resumen del plan de teléfonos móviles:

Requisito	Implementación
Una oferta que se va a mostrar en una zona dedicada a ofertas acerca de las actualizaciones Debe definirse el área de la página que muestra la oferta de actualizaciones. Asimismo, después de que Interact selecciona una oferta para visualizarla, la información debe registrarse.	• Punto de interacción: `ip_planSummaryBottomRight` • Evento: `evt_logOffer`
Dos ofertas para actualizaciones de teléfono Debe definirse cada área de la página que muestra las actualizaciones de teléfono.	• Punto de interacción: `ip_planSummaryTopRight` • Punto de interacción: `ip_planSummaryBottomLeft`
Para el análisis, debe registrar qué ofertas se aceptan y qué ofertas se rechazan.	• Evento: `evt_offerAccept` • Evento: `evt_offerReject`
También sabe que debe pasar el código de tratamiento de una oferta siempre que registre un contacto de oferta, aceptación o rechazo.	`NameValuePair`

Requisito	Implementación
Visualice tres imágenes rotativas en la página. Enlace las imágenes con las ofertas.	

Creación de puntos de interacción

Ahora puede pedir al usuario del entorno de diseño que cree los puntos de interacción y los eventos mientras empieza a codificar la integración con su punto de encuentro.

Para cada punto de interacción que muestre una oferta, debe obtener primero una oferta y, a continuación, extraer la información que necesita para mostrar la oferta. Por ejemplo, solicite una oferta para el área inferior derecha de la página web (`planSummaryBottomRight`)

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

Esta llamada de respuesta devuelve un objeto de respuesta que incluye una respuesta `OfferList`. No obstante, la página web no puede utilizar un objeto `OfferList`. Necesita un archivo de imagen de la oferta, que sabe que es uno de los atributos de oferta (`offerImg`). Debe extraer el atributo de oferta que necesita de `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"))
  {
     /* Utilizar este valor en el código de la página, por
      ejemplo: stringHtml = "<img src="+attribute.getValueAsString()+ " >   */
  }
 }
}
```

Configuración del registro

Ahora que está mostrando la oferta, desea registrarla como un contacto.

```
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)
```

En lugar de llamar a cada uno de estos métodos de forma individual, puede utilizar el método `executeBatch`, como se muestra en el siguiente ejemplo para la parte `planSummaryBottomLeft` de la página 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);

/** Crear la matriz de comandos */
```

```
Command[] commands =
{
 getOffersCommand,
 postEventCommand
};

/** Realizar la llamada */
BatchResponse batchResponse = api.executeBatch(sessionId, commands);
```

No es necesario definir el UACIOfferTrackingCode en este ejemplo. El servidor de ejecución de Interact registra automáticamente la última lista de tratamientos recomendada como contactos si no proporciona el UACIOfferTrackingCode.

Creación de códigos de tratamiento

Siempre que sea necesario, debe crear un NameValuePair que contenga el código de tratamiento, como en el siguiente ejemplo.

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

Enlace de imágenes a las ofertas

Para la segunda área en la página que muestra una actualización de teléfono, ha escrito algo para cambiar la imagen mostrada cada 30 segundos. Decide rotar entre tres imágenes y utiliza lo siguiente para recuperar el conjunto de ofertas que se deben guardar en caché para su uso en el código para rotar las imágenes.

```
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)
  {
      // obtener el valor de atributo offerimg y almacenarlo en algún lugar;
      // será la primera imagen que se muestre
  }
    else if(x==1)
  {
      // obtener el valor de atributo offerimg y almacenarlo en algún lugar;
      // será la segunda imagen que se muestre
  }
    else if(x==2)
  {
      // obtener el valor de atributo offerimg y almacenarlo en algún lugar;
      // será la tercera imagen que se muestre
  }
 }
}
```

Debe escribir el código de cliente captado en la memoria caché local y registrarlo en el contacto sólo una vez para cada oferta una vez visualizada su imagen. Para registrar el contacto, el parámetro UACITrackingCode debe publicarse como antes. Cada oferta tiene un código de seguimiento 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, si se pulsa la oferta, registra la oferta aceptada y las ofertas rechazadas. (En este escenario, las ofertas no seleccionadas explícitamente se consideran rechazadas). A continuación, se muestra un ejemplo si se selecciona la oferta `ip_planSummaryTopRight`:

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

En la práctica, será mejor para enviar las tres llamadas `postEvent` con el método `executeBatch`.

Diseño de la integración de la API de Interact

Creación de la integración de la API de Interact con su punto de encuentro requiere algún tipo de diseño antes de iniciar la implementación. Debe trabajar con el equipo de marketing para decidir dónde desea que el entorno de ejecución presente las ofertas en el punto de encuentro (definir los puntos de interacción), y qué otro tipo de seguimiento o funcionalidad interactiva desea utilizar (definir los eventos).

En la fase de diseño, pueden ser meros esquemas. Por ejemplo, para un sitio web de telecomunicaciones, la página de resumen del plan del cliente debe mostrar una oferta sobre la actualización de planes y dos ofertas de actualizaciones de teléfono.

Una vez la empresa ha decidido dónde y cómo desea interactuar con los clientes, debe utilizar Interact para definir los detalles. Un autor de diagrama de flujo debe diseñar los diagramas de flujo interactivos que se utilizarán cuando se produzcan los eventos de resegmentación. Debe decidir el número y los nombres de los puntos de interacción y los eventos, así como qué datos deben pasarse para garantizar una segmentación adecuada, la publicación de eventos y la recuperación de ofertas. El usuario del entorno de diseño define los puntos de interacción y los eventos para el canal interactivo. A continuación, utilice esos nombres cuando codifique la integración con el punto de encuentro en el entorno de ejecución.

También debe definir qué información de métricas es necesaria, para definir cuándo tiene que registrar los contactos y las respuestas de las ofertas.

Puntos a tener en cuenta

Cuando diseñe una interacción, tenga en cuenta el efecto que tienen una oferta no elegible, un servidor de ejecución inalcanzable y el tiempo de proceso en la interacción. Sea específico cuando defina rechazos de ofertas. Tenga en cuenta las características opcionales del producto que pueden mejorar la interacción.

Cuando diseñe la interacción:

Cree un contenido de relleno predeterminado
Cree el contenido de relleno predeterminado, un mensaje de marca positivo o un contenido vacío, para cada punto de interacción donde se puedan presentar ofertas. Este contenido de relleno se utiliza cuando no hay ofertas elegibles que se puedan ofrecer al visitante actual en la situación actual. Asigne este contenido de relleno predeterminado como la cadena predeterminada para el punto de interacción.

Incluya un método alternativo para presentar el contenido
Incluya algún método para presentar el contenido en el caso de que el punto de encuentro no pueda alcanzar el grupo de servidores de ejecución por algún motivo imprevisto.

Tenga en cuenta el tiempo necesario para ejecutar diagramas de flujo
Cuando desencadena eventos que resegmentan el visitante, incluidos `postEvent` y `setAudience`, recuerde que la ejecución de diagramas de flujo necesita un tiempo adicional. El método `getOffers` espera a que la resegmentación finalice antes de ejecutarse. Una resegmentación demasiado frecuente puede dificultar el rendimiento de las respuestas a las llamada de `getOffers`.

Decida qué significa un "rechazo de oferta"
Varios informes como, por ejemplo, el informe Resumen de rendimiento de canal de oferta, muestran el número de veces que se ha rechazado una oferta. Este informe muestra el número de veces que un `postEvent` ha desencadenado una acción Registrar rechazo de oferta. Debe determinar si la acción Registrar rechazo de oferta es para un rechazo real como, por ejemplo, la pulsación de un enlace con la etiqueta **No, gracias**. O bien, si la acción Registrar rechazo de oferta es para una oferta que se ignora como, por ejemplo, una página que muestra tres banners publicitarios distintos, de los cuales no se selecciona ninguno.

Decida qué características de selección de ofertas desea utilizar
Existen varias características opcionales que puede utilizar para mejorar la selección de ofertas de Interact. Estas características incluyen:

- Aprendizaje
- Supresión de la oferta
- Asignaciones de oferta individuales
- Otros elementos de presentación de ofertas.

Debe determinar cuántas, si existen, de estas características opcionales mejorarán sus interacciones.

Capítulo 6. Gestión de la API de IBM Interact

Siempre que utiliza el método `startSession`, crea una sesión de ejecución de Interact en el servidor de ejecución. Puede utilizar las propiedades de configuración para gestionar las sesiones en un servidor de ejecución.

Deberá configurar estos valores a medida que implemente la integración de Interact con el punto de encuentro.

Estas propiedades de configuración están en la categoría `sessionManagement`.

Entorno local y la API de Interact

Puede utilizar Interact para los puntos de encuentro distintos del inglés. El punto de encuentro y todas las cadenas en la API utilizan el entorno local definido para el usuario del entorno de ejecución.

Sólo puede seleccionar un entorno local por grupo de servidores.

Por ejemplo, en el entorno de ejecución, supongamos que crea dos usuarios: asm_admin_en con el entorno local de usuario establecido en inglés y asm_admin_fr con el entorno local de usuario establecido en francés. Si el punto de encuentro está diseñado para francófonos, defina la propiedad `asmUserForDefaultLocale` para el entorno de ejecución como asm_admin_fr.

Acerca de la supervisión JMX

Interact proporciona el servicio de supervisión JMX (Java Management Extensions) al que puede acceder con cualquier aplicación de supervisión JMX. Esta supervisión JMX permite supervisar y gestionar los servidores de ejecución.

Los atributos JMX proporcionan una gran cantidad de información detallada sobre el servidor de ejecución. Por ejemplo, el atributo JMX `ErrorCount` proporciona el número de mensajes de error registrados desde el último restablecimiento o el último inicio del sistema. Puede utilizar esta información para ver con qué frecuencia se producen errores en el sistema. Si ha codificado el sitio web para sólo invocar la finalización de la sesión si alguien completa una transacción, también puede comparar `startSessionCount` con `endSessionCount` para ver cuántas transacciones están incompletas.

Interact da soporte a los protocolos RMI y JMXMP, tal como se define en JSR 160. Puede conectarse al servicio de supervisión JMX con un cliente JMX compatible con JSR160.

Las diagramas de flujo interactivos sólo pueden supervisarse con la supervisión JMX. La información sobre los diagramas de flujo interactivos no aparecen en Campaign Monitoring.

Nota: Si utiliza IBM WebSphere con un gestor de nodos, debe definir el argumento JVM genérico para habilitar la supervisión JMX.

Configuración de Interact para utilizar la supervisión JMX con el protocolo RMI

Utilice este procedimiento para configurar Interact para utilizar la supervisión JMX con el protocolo RMI.

Acerca de esta tarea

La dirección predeterminada de supervisión para el protocolo RMI es `service:jmx:rmi:///jndi/rmi://RuntimeServer:port/interact`.

Procedimiento

En la Marketing Platform del entorno de ejecución, edite las siguientes propiedades de configuración en la categoría `Interact > monitoring`.

Propiedad de configuración	Valor
`protocol`	**RMI**
`port`	El número de puerto del servicio JMX
`enableSecurity`	**False** La implementación de Interact del protocolo RMI no da soporte a la seguridad.

Configuración de Interact para utilizar la supervisión JMX con el protocolo JMXMP

Utilice este procedimiento para configurar Interact para utilizar la supervisión JMX con el protocolo JMXMP.

Antes de empezar

El protocolo JMXMP requiere dos bibliotecas adicionales en el siguiente orden en la ruta de clases: `InteractJMX.jar` y `jmxremote_optional.jar`. Ambos archivos se pueden encontrar en el directorio `lib` de la instalación del entorno de ejecución.

Acerca de esta tarea

Si habilita la seguridad, el nombre de usuario y la contraseña deben coincidir con un usuario en la Marketing Platform para el entorno de ejecución. No puede utilizar una contraseña vacía.

La dirección predeterminada de supervisión para el protocolo JMXMP es `service:jmx:jmxmp://RuntimeServer:port`.

Procedimiento

1. Verifique que las bibliotecas `InteractJMX.jar` y `jmxremote_optional.jar` estén en la ruta de clase en orden. Si no están en la ruta de clase, añádalas a la ruta de clase.
2. En la Marketing Platform del entorno de ejecución, edite las siguientes propiedades de configuración en la categoría `Interact > monitoring`.

Propiedad de configuración	Valor
`protocol`	**JMXMP**

Propiedad de configuración	Valor
port	El número de puerto del servicio JMX
enableSecurity	**False** para inhabilitar la seguridad, o **True** para habilitar la seguridad

Configuración de Interact para utilizar los scripts de jconsole para la supervisión JMX

Si no tiene una aplicación de supervisión JMX aparte, puede utilizar la jconsole que se instala con la JVM. Puede iniciar jconsole con los scripts de inicio en el directorio Interact/tools.

Acerca de esta tarea

El script de jconsole utiliza el protocolo JMXMP para la supervisión de forma predeterminada. Los valores predeterminados de jconsole.bat son:

La conexión 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%
```

La conexión 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
```

Procedimiento

1. Abra Interact\tools\jconsole.bat (Windows) o Interact/tools/jconsole.sh (UNIX) en un editor de texto.
2. Establezca INTERACT_LIB en la ruta completa del directorio *InteractInstallationDirectory*/lib.
3. Establezca HOST en el nombre de host del servidor de ejecución que desea supervisar.
4. Establezca PORT en el puerto que ha configurado para que escuche JMX con la propiedad Interact > monitoring > port.
5. Opcional: Si utiliza el protocolo RMI para la supervisión, añada un comentario antes de la conexión JMXMP y elimínelo antes de la conexión RMI.

Atributos JMX

Hay varios atributos disponibles para la supervisión JMX. Los atributos de entorno de diseño incluyen la supervisión de ETL del historial de respuestas de contacto. Los atributos de entorno de ejecución incluyen las excepciones, varios atributos de diagrama de flujo diferentes, el entorno local, el registrador y las estadísticas de agrupación de subprocesos. También hay disponibles varios atributos de estadísticas de servicio. Todos los datos proporcionados por la supervisión JMX son desde el último restablecimiento o el último inicio del sistema. Por ejemplo, un recuento del número de elementos desde el último restablecimiento o el último inicio del sistema, no desde la instalación.

Atributos de supervisor de ETL del historial de contactos y respuestas

Los atributos del Supervisor de ETL del historial de contactos y respuestas forman parte del entorno de diseño. Todos los siguientes atributos forman parte del entorno de ejecución.

Tabla 9. Supervisor de ETL del historial de contactos y respuestas

Atributo	Descripción
AvgCHExecutionTime	El número promedio de milisegundos que el módulo del historial de contactos y respuestas necesita para escribir en la tabla del historial de contactos. Este promedio sólo se calcula para las operaciones que han sido satisfactorias y para las que se ha escrito un registro como mínimo en la tabla del historial de contactos.
AvgETLExecutionTime	El número promedio de milisegundos que el módulo del historial de contactos y respuestas necesita para leer datos del entorno de ejecución. El promedio incluye el tiempo de las operaciones satisfactorias y anómalas.
AvgRHExecutionTime	El número promedio de milisegundos que el módulo del historial de contactos y respuestas necesita para escribir en la tabla del historial de respuestas. Este promedio sólo se calcula para las operaciones que han sido satisfactorias y para las que se ha escrito un registro como mínimo en la tabla del historial de respuestas.
ErrorCount	El número de mensajes de error registrados desde el último restablecimiento o el último inicio del sistema, si existen.
HighWaterMarkCHExecutionTime	El número máximo de milisegundos que el módulo del historial de contactos y respuestas ha necesitado para escribir en la tabla del historial de contactos. Este valor sólo se calcula para las operaciones que han sido satisfactorias y para las que se ha escrito un registro como mínimo en la tabla del historial de contactos.
HighWaterMarkETLExecutionTime	El número máximo de milisegundos que el módulo del historial de contactos y respuestas ha necesitado para leer datos del entorno de ejecución. El cálculo incluye las operaciones satisfactorias y anómalas.

Tabla 9. Supervisor de ETL del historial de contactos y respuestas (continuación)

Atributo	Descripción
HighWaterMarkRHExecutionTime	El número máximo de milisegundos que el módulo del historial de contactos y respuestas ha necesitado para escribir en la tabla del historial de respuestas. Este valor sólo se calcula para las operaciones que han sido satisfactorias y para las que se ha escrito un registro como mínimo en la tabla del historial de respuestas.
LastExecutionDuration	El número de milisegundos que el módulo del historial de contactos y respuestas ha necesitado para realizar la última copia.
NumberOfExecutions	El número de veces que el módulo del historial de contactos y respuestas se ha ejecutado desde la última inicialización.
LastExecutionStart	La hora a la que se ha iniciado la última ejecución del módulo del historial de contactos y respuestas.
LastExecutionSuccessful	Si es true, la última ejecución del módulo del historial de contactos y respuestas ha sido satisfactoria. Si es false, se ha producido un error.
NumberOfContactHistoryRecordsMarked	El número de registros del historial de contactos en la tabla `UACI_CHStaging` que se mueven durante la ejecución actual del módulo del historial de contactos y respuestas. Este valor sólo es mayor que cero si se está ejecutando el módulo del historial de contactos y respuestas.
NumberOfResponseHistoryRecordsMarked	El número de registros del historial de respuestas en la tabla `UACI_RHStaging` que se mueven durante la ejecución actual del módulo del historial de contactos y respuestas. Este valor sólo es mayor que cero si se está ejecutando el módulo del historial de contactos y respuestas.

Atributos de excepción

Los atributos de excepción forman parte del entorno de ejecución.

Tabla 10. Excepciones

Atributo	Descripción
errorCount	El número de mensajes de error que se han registrado desde el último restablecimiento o el último inicio del sistema.

Tabla 10. Excepciones (continuación)

Atributo	Descripción
warningCount	El número de mensajes de aviso que se han registrado desde el último restablecimiento o el último inicio del sistema.

Atributos de estadísticas de motor de diagrama de flujo

Los atributos de estadísticas de motor de diagrama de flujo forman parte del entorno de ejecución.

Tabla 11. Estadísticas de motor de diagrama de flujo

Atributo	Descripción
activeProcessBoxThreads	Recuento activo de subprocesos del proceso de diagrama de flujo (compartidos entre todas las ejecuciones) que están actualmente en ejecución.
activeSchedulerThreads	Recuento activo de subprocesos del planificador de diagrama de flujo que están actualmente en ejecución.
avgExecutionTimeMillis	Promedio de tiempo de ejecución de diagrama de flujo en milisegundos.
CurrentJobsInProcessBoxQueue	El número de trabajos que están a la espera de ser ejecutados por los subprocesos del proceso de diagrama de flujo.
CurrentJobsInSchedulerQueue	El número de trabajos que están a la espera de ser ejecutados por los subprocesos del planificador de diagrama de flujo.
maximumProcessBoxThreads	Número máximo de subprocesos del proceso de diagrama de flujo (compartidos entre todas las ejecuciones) que pueden ejecutarse.
maximumSchedulerThreads	Número máximo de subprocesos del planificador de diagrama de flujo (un subproceso por ejecución) que pueden ejecutarse.
numExecutionsCompleted	Número total de ejecuciones de diagrama de flujo que se han completado.
numExecutionsStarted	Número total de ejecuciones de diagrama de flujo que se han iniciado.

Atributos de diagramas de flujo específicos por canal interactivo

Los atributos de diagramas de flujo específicos por canal interactivo forman parte del entorno de ejecución.

Tabla 12. Diagramas de flujo específicos por canal interactivo

Atributo	Descripción
AvgExecutionTimeMillis	Promedio de tiempo de ejecución en milisegundos para este diagrama de flujo en este canal interactivo.
HighWaterMarkForExecutionTime	Tiempo de ejecución máximo en milisegundos para este diagrama de flujo en este canal interactivo.
LastCompletedExecutionTimeMillis	Tiempo de ejecución en milisegundos para la última finalización de este diagrama de flujo en este canal interactivo.
NumExecutionsCompleted	Número total de ejecuciones que se han completado para este diagrama de flujo en este canal interactivo.
NumExecutionsStarted	Número total de ejecuciones que se han iniciado para este diagrama de flujo en este canal interactivo.

Atributos de entorno local

Los atributos de entorno local forman parte del entorno de ejecución.

Tabla 13. Entorno local

Atributo	Descripción
locale	Valor de entorno local del cliente JMX.

Atributos de configuración del registrador

Los atributos de configuración del registrador forman parte del entorno de ejecución.

Tabla 14. Configuración del registrador

Atributo	Descripción
categoría	Cambia la categoría de registro en la que puede manipularse el nivel de registro.

Atributos de estadísticas de agrupación de subprocesos de servicios

Los atributos de estadísticas de agrupación de subprocesos de servicios forman parte del entorno de ejecución.

Tabla 15. Estadísticas de agrupación de subprocesos de servicios

Atributo	Descripción
activeContactHistThreads	El número aproximado de subprocesos que están ejecutando activamente tareas para el historial de contactos y el historial de respuestas.

Tabla 15. Estadísticas de agrupación de subprocesos de servicios (continuación)

Atributo	Descripción
activeFlushCacheToDBThreads	El número aproximado de subprocesos que están ejecutando activamente tareas para vaciar las estadísticas guardadas en caché en el almacén de datos.
activeOtherStatsThreads	El número aproximado de subprocesos que están ejecutando activamente tareas para Estadísticas elegibles, Actividades de eventos y Estadísticas predeterminadas.
CurrentHighWaterMarkInContactHistQueue	Número máximo de entradas en cola que va a registrar el servicio que recopila los datos del historial de contactos y respuestas.
CurrentHighWaterMark InFlushCachetoDBQueue	Número máximo de entradas en cola que va a registrar el servicio que escribe los datos de la memoria caché en las tablas de base de datos.
CurrentHighWaterMarkInOtherStatsQueue	Número máximo de entradas en cola que va a registrar el servicio que recopila las estadísticas de elegibilidad de ofertas, las estadísticas de uso de cadenas predeterminadas, las estadísticas de actividades de eventos y el registro personalizado en los datos de la tabla.
currentMsgsInContactHistQueue	El número de trabajos en la cola de la agrupación de subprocesos que se utiliza para el historial de contactos y el historial de respuestas.
currentMsgsInFlushCacheToDBQueue	El número de trabajos en la cola de la agrupación de subprocesos que se utiliza para vaciar las estadísticas guardadas en caché en el almacén de datos.
currentMsgsInOtherStatsQueue	El número de trabajos en la cola de la agrupación de subprocesos que se utiliza para Estadísticas elegibles, Actividades de eventos y Estadísticas predeterminadas.
maximumContactHistThreads	El número máximo de subprocesos que han estado simultáneamente en la agrupación que se utiliza para el historial de contactos y el historial de respuestas.
maximumFlushCacheToDBThreads	El número máximo de subprocesos que han estado simultáneamente en la agrupación que se utiliza para vaciar las estadísticas guardadas en caché en el almacén de datos.

Tabla 15. Estadísticas de agrupación de subprocesos de servicios (continuación)

Atributo	Descripción
maximumOtherStatsThreads	El número máximo de subprocesos que han estado simultáneamente en la agrupación que se utiliza para Estadísticas elegibles, Actividades de eventos y Estadísticas predeterminadas.

Atributos de estadísticas de servicio

Las estadísticas de servicio están formadas por un conjunto de atributos para cada servicio.

- ContactHistoryMemoryCacheStatistics: el servicio que recopila datos para las tablas de preparación del historial de contactos.
- CustomLoggerStatistics: el servicio que recopila datos personalizados para escribir en una tabla (un evento que utiliza el parámetro de evento `UACICustomLoggerTableName`).
- Estadísticas predeterminadas: el servicio que recopila las estadísticas sobre el número de veces que se ha utilizado la cadena predeterminada para el punto de interacción.
- Estadísticas de elegibilidad: el servicio que escribe estadísticas para las ofertas elegibles.
- Estadísticas de actividades de eventos: el servicio que recopila las estadísticas de eventos, de eventos del sistema como `getOffer` o `startSession` y eventos de usuario desencadenados por `postEvent`.
- Estadísticas de memoria caché del historial de respuestas: el servicio que escribe en las tablas de preparación del historial de respuestas.
- Estadísticas de respuestas de sesiones cruzadas: el servicio que recopila los datos de seguimiento de respuestas de sesiones cruzadas.

Tabla 16. Estadísticas de servicios

Atributo	Descripción
Count	El número de mensajes procesados.
ExecTimeInsideMutex	La cantidad de tiempo necesaria para procesar mensajes para este servicio, excepto el tiempo a la espera de otros subprocesos, en milisegundos. Si hay una gran diferencia entre ExecTimeInsidMutex y ExecTimeMillis, deberá cambiar el tamaño de la agrupación de subprocesos para el servicio.
ExecTimeMillis	La cantidad de tiempo necesaria para procesar mensajes para este servicio, incluido el tiempo a la espera de otros subprocesos, en milisegundos.
ExecTimeOfDBInsertOnly	La cantidad de tiempo en milisegundos necesaria para procesar sólo la parte de inserción por lotes.
HighWaterMark	El número máximo de mensajes procesados para este servicio.

Tabla 16. Estadísticas de servicios (continuación)

Atributo	Descripción
NumberOfDBInserts	El número total de inserciones por lotes ejecutadas.
TotalRowsInserted	El número total de filas insertadas en la base de datos.

Atributos de Estadísticas de servicios — Utilidad de carga de base de datos

Los atributos de Estadísticas de servicios — Utilidad de carga de base de datos forman parte del entorno de ejecución.

Tabla 17. Estadísticas de servicios — Utilidad de carga de base de datos

Atributo	Descripción
ExecTimeOfWriteToCache	La cantidad de tiempo en milisegundos necesaria para escribir en la memoria caché de archivos, incluida la escritura en los archivos y la obtención de la clave primaria de la base de datos cuando sea necesario.
ExecTimeOfLoaderDBAccessOnly	La cantidad de tiempo en milisegundos necesaria para ejecutar sólo la parte de cargador de base de datos.
ExecTimeOfLoaderThreads	La cantidad de tiempo en milisegundos necesaria para los subprocesos del cargador de base de datos.
ExecTimeOfFlushCacheFiles	La cantidad de tiempo en milisegundos necesaria para vaciar la memoria caché y volver a crear las nuevas.
ExecTimeOfRetrievePKDBAccess	La cantidad de tiempo en milisegundos necesaria para recuperar el acceso de base de datos de clave primaria.
NumberOfDBLoaderRuns	El número total de ejecuciones del cargador de base de datos.
NumberOfLoaderStagingDirCreated	El número total de directorios de preparación creados.
NumberOfLoaderStagingDirRemoved	El número total de directorios de preparación eliminados.
NumberOfLoaderStaging DirMovedToAttention	El número total de directorios de preparación renombrados como de atención.
NumberOfLoaderStagingDirMovedToError	El número total de directorios de preparación renombrados como de error.
NumberOfLoaderStagingDirRecovered	El número total de directorios de preparación recuperados, incluido en el inicio y los que se han vuelto a ejecutar en los subprocesos de fondo.
NumberOfTimesRetrievePKFromDB	El número total de veces que se recupera la clave primaria de la base de datos.
NumberOfLoaderThreadsRuns	El número total de ejecuciones de subprocesos del cargador de base de datos.

Tabla 17. Estadísticas de servicios — Utilidad de carga de base de datos (continuación)

Atributo	Descripción
NumberOfFlushCacheFiles	El número total de veces que se vacía la memoria caché de archivos.

Atributos de estadísticas de API

Los atributos de estadísticas de API forman parte del entorno de ejecución.

Tabla 18. Estadísticas de API

Atributo	Descripción
endSessionCount	El número de llamadas a la API `endSession` desde el último restablecimiento o el último inicio del sistema.
endSessionDuration	Tiempo transcurrido para la última llamada a la API `endSession`.
executeBatchCount	El número de llamadas a la API `executeBatch` desde el último restablecimiento o el último inicio del sistema.
executeBatchDuration	Tiempo transcurrido para la última llamada a la API `executeBatch`.
getOffersCount	El número de llamadas a la API `getOffers` desde el último restablecimiento o el último inicio del sistema.
getOffersDuration	Tiempo transcurrido para la última llamada a la API `getOffer`.
getProfileCount	El número de llamadas a la API `getProfile` desde el último restablecimiento o el último inicio del sistema.
getProfileDuration	Tiempo transcurrido para la última llamada a la API `getProfileDuration`.
getVersionCount	El número de llamadas a la API `getVersion` desde el último restablecimiento o el último inicio del sistema.
getVersionDuration	Tiempo transcurrido para la última llamada a la API `getVersion`.
loadOfferSuppressionDuration	Tiempo transcurrido para la última llamada a la API `loadOfferSuppression`.
LoadOffersBySQLCount	El número de llamadas a la API `LoadOffersBySQL` desde el último restablecimiento o el último inicio del sistema.
LoadOffersBySQLDuration	Tiempo transcurrido para la última llamada a la API `LoadOffersBySQL`.
loadProfileDuration	Tiempo transcurrido para la última llamada a la API `loadProfile`.
loadScoreOverrideDuration	Tiempo transcurrido para la última llamada a la API `loadScoreOverride`.

Tabla 18. Estadísticas de API (continuación)

Atributo	Descripción
postEventCount	El número de llamadas a la API `postEvent` desde el último restablecimiento o el último inicio del sistema.
postEventDuration	Tiempo transcurrido para la última llamada a la API `postEvent`.
runSegmentationDuration	Tiempo transcurrido para la última llamada a la API `runSegmentation`.
setAudienceCount	El número de llamadas a la API `setAudience` desde el último restablecimiento o el último inicio del sistema.
setAudienceDuration	Tiempo transcurrido para la última llamada a la API `setAudience`.
setDebugCount	El número de llamadas a la API `setDebug` desde el último restablecimiento o el último inicio del sistema.
setDebugDuration	Tiempo transcurrido para la última llamada a la API `setDebug`.
startSessionCount	El número de llamadas a la API `startSession` desde el último restablecimiento o el último inicio del sistema.
startSessionAverage	El Promedio de tiempo transcurrido para la última llamada a la API `startSession`.
ActiveSessionCount	El número de sesiones que están activas simultáneamente en la instancia de tiempo de ejecución de Interact. **Nota:** El valor de ActiveSessionCount en JMX MBean com.unicacorp.interact:type=api, group=Statistics no tiene en cuentas los sucesos con tiempo de espera agotado y, por lo tanto, podría ser que se mostraran recuentos activos incorrectos.

Atributos de estadísticas de optimizador de aprendizaje

Los atributos de estadísticas de optimizador de aprendizaje forman parte del entorno de ejecución.

Tabla 19. Estadísticas de optimizador de aprendizaje

Atributo	Descripción
LearningOptimizerAcceptCalls	El número de eventos de aceptación pasados al módulo de aprendizaje.
LearningOptimizer AcceptTrackingDuration	El número total de milisegundos necesarios para registrar los eventos de aceptación en el módulo de aprendizaje.
LearningOptimizerContactCalls	El número de eventos de contacto pasados al módulo de aprendizaje.
LearningOptimizer ContactTrackingDuration	El número total de milisegundos necesarios para registrar los eventos de contacto en el módulo de aprendizaje.

Tabla 19. Estadísticas de optimizador de aprendizaje (continuación)

Atributo	Descripción
LearningOptimizerLogOtherCalls	El número de eventos de no aceptación y de no contacto pasados al módulo de aprendizaje.
LearningOptimizer LogOtherTrackingDuration	La duración en milisegundos necesaria para registrar otros eventos (de no contacto y no aceptación) en el módulo de aprendizaje.
LearningOptimizer NonRandomCalls	El número de veces que se ha aplicado la implementación de aprendizaje configurada.
LearningOptimizer RandomCalls	El número de veces que se ha omitido la implementación de aprendizaje configurada y se ha aplicado la selección aleatoria.
LearningOptimizer RecommendCalls	El número de solicitudes de recomendación pasadas al módulo de aprendizaje.
LearningOptimizer RecommendDuration	El número total de milisegundos invertidos en la lógica de recomendación de aprendizaje.

Atributos de estadísticas de ofertas predeterminadas

Los atributos de estadísticas de ofertas predeterminadas forman parte del entorno de ejecución.

Tabla 20. Estadísticas de ofertas predeterminadas

Atributo	Descripción
LoadDefaultOffersDuration	Tiempo transcurrido en la carga de ofertas predeterminadas.
DefaultOffersCalls	Número de veces que se realiza la carga de ofertas predeterminadas.

Atributos de asignadores de mensajes desencadenados

Los atributos de asignadores de mensajes desencadenados forman parte del entorno de ejecución.

Tabla 21. Asignadores de mensajes desencadenados

Atributo	Descripción
NumRequested	El número total de ofertas solicitado para su asignación mediante este asignador.
NumDispatched	El número total de ofertas que ha asignado correctamente este asignador.
AvgExecutionTime	El promedio de tiempo en milisegundos que utiliza este asignador para asignar una oferta. Solo se incluyen en el cálculo las ofertas asignadas correctamente a las pasarelas.
CurrentQueueSize	El número de ofertas que están a la espera de ser asignadas actualmente.

Tabla 21. Asignadores de mensajes desencadenados (continuación)

Atributo	Descripción
GatewayInvocation	El número de ofertas asignadas por este asignador a cada pasarela, y el promedio de tiempo de asignación en milisegundos. El formato del valor es `{gateway name=[número de ofertas, tiempo promedio de asignación]}`.

Atributos de pasarelas de mensajes desencadenados

Los atributos de pasarelas de mensajes desencadenados forman parte del entorno de ejecución.

Tabla 22. Pasarelas de mensajes desencadenados

Atributo	Descripción
NumValidationRequested	El número total de ofertas que ha solicitado esta pasarela para su validación.
NumValidated	El número total de ofertas que ha validado correctamente esta pasarela.
AvgValidationTime	El promedio de tiempo en milisegundos que utiliza esta pasarela para validar una oferta. Solo se incluyen en el cálculo las ofertas validadas correctamente.
NumDeliveryRequested	El número total de ofertas que ha solicitado esta pasarela para su entrega.
NumDelivered	El número total de ofertas que ha entregado correctamente esta pasarela.
AvgDeliveryTime	El promedio de tiempo en milisegundos que utiliza esta pasarela para entregar una oferta. Solo se incluyen en el cálculo las ofertas entregadas correctamente.

Atributos de mensajes sobre mensajes desencadenados

Los atributos de mensajes sobre mensajes desencadenados forman parte del entorno de ejecución.

Tabla 23. Mensajes sobre mensajes desencadenados

Atributo	Descripción
ProcessSuccessCount	El número total de veces que se ha ejecutado correctamente este mensaje desencadenado.
AvgSuccessProcessTime	El promedio de tiempo en milisegundos que ha empleado este mensaje desencadenado en cada ejecución correcta.

Tabla 23. Mensajes sobre mensajes desencadenados (continuación)

Atributo	Descripción
ProcessErrorCount	El número total de veces que este mensaje desencadenado no se ha ejecutado correctamente.
AvgErrorProcessTime	El promedio de tiempo en milisegundos que ha empleado este mensaje desencadenado en cada ejecución incorrecta.
SelectBranchCount	El número total de veces que se ha ejecutado la selección de ramas durante el proceso de mensajes desencadenados.
AvgSelectBranchTime	El promedio de tiempo en milisegundos que utiliza la ejecución de la selección de ramas durante el proceso de mensajes desencadenados.
SelectOfferCount	El número total de veces que se ha ejecutado la selección de ofertas durante el proceso de mensajes desencadenados.
AvgSelectOfferTime	El promedio de tiempo en milisegundos que utiliza la ejecución de la selección de ofertas durante el proceso de mensajes desencadenados.
SelectChannelCount	El número total de veces que se ha ejecutado la selección de canales durante el proceso de mensajes desencadenados.
AvgSelectChannelTime	El promedio de tiempo en milisegundos que utiliza la ejecución de la selección de canales durante el proceso de mensajes desencadenados.
FlowchartWaitCount	El número total de veces que este mensaje desencadenado ha esperado a que se complete correctamente la segmentación.
AvgFlowchartWaitTime	El promedio de tiempo en milisegundos que ha esperado este mensaje desencadenado a que se complete la segmentación en cada ejecución.
WaitFlowchartTimeoutCount	El número total de veces que este mensaje desencadenado ha excedido el tiempo de espera hasta que se complete la segmentación.

Operaciones de JMX

Hay varias operaciones disponibles para la supervisión JMX.

En la siguiente tabla se describen las operaciones disponibles para la supervisión JMX.

Grupo	Atributo	Descripción
Configuración del registrador	activateDebug	Establece el nivel de registro del archivo de registro definido en `Interact/conf/interact_log4j.properties` en Depurar.
Configuración del registrador	activateError	Establece el nivel de registro del archivo de registro definido en `Interact/conf/interact_log4j.properties` en Error.
Configuración del registrador	activateFatal	Establece el nivel de registro del archivo de registro definido en `Interact/conf/interact_log4j.properties` en Muy grave.
Configuración del registrador	activateInfo	Establece el nivel de registro del archivo de registro definido en `Interact/conf/interact_log4j.properties` en Información.
Configuración del registrador	activateTrace	Establece el nivel de registro del archivo de registro definido en `Interact/conf/interact_log4j.properties` en Rastreo.
Configuración del registrador	activateWarn	Establece el nivel de registro del archivo de registro definido en `Interact/conf/interact_log4j.properties` en Aviso.
Entorno local	changeLocale	Cambia el entorno local del cliente JMX. Los entornos locales soportados de Interact son `de`, `en`, `es` y `fr`.
ContactResponseHistory ETLMonitor	reset	Restablece todos los contadores.
Estadísticas de ofertas predeterminadas	updatePollPeriod	Actualiza defaultOfferUpdatePollPeriod. Este valor, en segundos, indica al sistema cuánto tiempo debe esperar antes de actualizar las ofertas predeterminadas en la memoria caché. Si se establece en `-1`, el sistema lee el número de ofertas predeterminadas sólo en el inicio.

Capítulo 7. Clases y métodos para IBM Interact Java, SOAP y API de REST

En las siguientes secciones se muestran de requisitos y otros detalles que debe conocer antes de empezar a trabajar con la API de Interact.

Nota: En esta sección se supone que está familiarizado con el punto de encuentro, el lenguaje de programación Java y el trabajo con una API basada en Java.

La API de Interact tiene un adaptador de cliente Java que utiliza la serialización Java a través de HTTP. Asimismo, Interact proporciona un WSDL para dar soporte a los clientes SOAP. El WSDL muestra el mismo conjunto de funciones que el adaptador de cliente Java, por lo que continúan aplicándose las siguientes secciones, excepto los ejemplos.

Nota: No se permite tener varias apariciones de un parámetro en una única llamada a la API.

Clases de API de Interact

La API de Interact se basa en la clase `InteractAPI`.

Existen 6 interfaces de soporte.

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

Estas interfaces tienen 3 clases de soporte concretas. Se debe crear una instancia de las siguientes dos clases concretas y se deben pasar como argumentos a los métodos de API de Interact:

- NameValuePairImpl
- CommandImpl

Una tercera clase concreta, denominada `AdvisoryMessageCode`, está disponible para proporcionar las constantes utilizadas para distinguir los códigos de mensaje devueltos del servidor siempre que sea aplicable.

El resto de esta sección describe los métodos que componen la API de Interact.

Serialización Java a través de los requisitos previos HTTP

El adaptador de cliente Java utiliza la serialización Java a través de HTTP.

Los requisitos previos para utilizar el adaptador de cliente Java para la serialización Java a través de HTTP son:

1. Añada el archivo siguiente a la variable `CLASSPATH`:

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

2. Todos los objetos que se pasan entre el cliente y el servidor pueden encontrarse en el paquete `com.unicacorp.interact.api`. Para obtener información más detallada, consulte el Javadoc de la API de Interact instalado en el servidor de ejecución, en `Interact_Home/docs/apiJavaDoc`. Para visualizar el Javadoc, abra el archivo `index.html` que encontrará en esta ubicación con cualquier navegador web.
3. Para obtener una instancia de la clase `InteractAPI`, llame al método estático `getInstance` con el URL del servidor de ejecución de Interact.

Requisitos previos de SOAP

Para poder acceder al servidor de ejecución con SOAP, debe realizar varias tareas de requisito previo para configurar el entorno.

Importante: La prueba de rendimiento muestra que el adaptador de serialización Java tiene un rendimiento con una tasa mucho mayor que un cliente SOAP generado. Para garantizar el máximo rendimiento, utilice el adaptador de serialización Java siempre que sea posible.

Para acceder al servidor de ejecución con SOAP, debe hacer lo siguiente:

1. Convierta el WSDL de la API de Interact con el kit de herramientas SOAP que prefiera.

   El WSDL de la API de Interact se instala con Interact en el directorio `Interact/conf`.

   Cuando configura SOAP utilizando los archivos XML de WSDL, debe modificar los URL con el nombre de host y el puerto del servidor de ejecución.

   El texto del WSDL está disponible al final de la Guía de administración de Interact.
2. Instale y configure el servidor de ejecución.

   El servidor de ejecución debe estar ejecutándose para probar completamente la integración.
3. Verifique que está utilizando la versión de SOAP correcta.

   Interact utiliza axis2 1.3 como la infraestructura SOAP en los servidores de ejecución de Interact. Para obtener información detallada sobre qué versiones de SOAP admite axis2 1.3, consulte el siguiente sitio web:

   Apache Axis2

   Interact se ha probado con los clientes SOAP de axis2, XFire, JAX-WS-Ri, DotNet, SOAPUI e IBM RAD.

Requisitos previos de REST

Un método para llamar a la API de Interact es utilizar llamadas de formato JSON (JavaScript Object Notation) sobre HTTP, aquí denominadas la API REST. La API REST API tiene la ventaja de que tiene un mejor rendimiento que SOAP, aunque el adaptador de serialización de Java sigue siendo el método más rápido para las llamadas a la API de Interact.

Antes de empezar a utilizar la API REST, debe tener presente lo siguiente:

- El URL que da soporte a las llamadas REST a la API de Interact es:

  `http://Interact_Runtime_Server:PORT/interact/servlet/RestServlet`, y sustituye al nombre de host o dirección IP real del servidor de ejecución de Interact y al puerto donde está desplegado Interact.

- Hay dos clases de Interact específicas para la API REST: `RestClientConnector`, que ayuda a establecer conexión con una instancia de tiempo de ejecución de Interact vía REST con el formato JSON, y`RestFieldConstants`, que describe el formato subyacente del mensaje JSON que se utiliza para las solicitudes y respuestas de la API.
- En `Interact _Home/samples/javaApi/InteractRestClient.java` se proporciona un cliente REST de muestra. Aunque el código de muestra es simplemente un ejemplo, debería ofrecer un punto de partida para mostrar cómo se utiliza la API REST.
- Para ver una descripción completa de las clases de la API REST junto con toda la demás información de la API de Interact información el Javadoc instalado en el servidor de ejecución, en `Interact_Home/docs/apiJavaDoc`.
- La API REST devuelve valores de SessionID y mensajes en formato de caracteres especiales HTML, no en formato Unicode.

Además de la información mencionada, la API REST admite todos los métodos que admiten los demás protocolos para utilizar la API de Interact.

JavaDoc de la API

Además de la Guía del administrador de Interact, el Javadoc de la API de Interact se instala con el servidor de ejecución. El Javadoc se instala, para realizar consultas, en el directorio `Interact_Home/docs/apiJavaDoc`.

Ejemplo de API

Todos los ejemplos de la guía se han creado con la serialización Java a través del adaptador HTTP. Las clases generadas a partir de WSDL pueden variar según el kit de herramientas SOAP y las opciones que seleccione. Si utiliza SOAP, estos ejemplos no funcionarán exactamente igual en su entorno.

Cómo trabajar con datos de sesión

Cuando inicia una sesión con el método `startSession`, los datos de sesión se cargan en la memoria. Durante la sesión, puede leer y escribir en los datos de sesión (que son un superconjunto de los datos de perfil estáticos).

La sesión contiene los siguientes datos:

- Datos de perfil estáticos
- Asignaciones de segmento
- Datos en tiempo real
- Recomendaciones de ofertas

Todos los datos de sesión están disponibles hasta que invoca el método `endSession` o transcurre el tiempo de `sessionTimeout`. Una vez finalizada la sesión, todos los datos que no se han guardado de forma explícita en el historial de contactos o respuestas o en otra tabla de base de datos se pierden.

Los datos se almacenan como un conjunto de pares nombre-valor. Si los datos se leen de una tabla de base de datos, el nombre es la columna de la tabla.

Puede crear estos pares nombre-valor cuando trabaje con la API de Interact. No es necesario declarar todos los pares nombre-valor en un área global. Si establece nuevos parámetros de eventos como pares nombre-valor, el entorno de ejecución añade los pares nombre-valor a los datos de sesión. Por ejemplo, si utiliza los

parámetros de evento con el método postEvent, el entorno de ejecución añade los parámetros de evento a los datos de sesión, aunque los parámetros de evento no estén disponibles en los datos del perfil. Estos datos sólo existen en los datos de sesión.

Puede sobrescribir datos de sesión en cualquier momento. Por ejemplo, si parte del perfil de cliente incluye creditScore, puede pasar en un parámetro de evento utilizando el tipo personalizado NameValuePair. En la clase NameValuePair, puede utilizar los métodos setName y setValueAsNumeric para cambiar el valor. El nombre debe coincidir. En los datos de sesión, el nombre no es sensible a las mayúsculas y minúsculas. Por lo tanto, los nombres creditscore o CrEdItScOrE deben sobrescribir creditScore.

Sólo se mantienen los últimos datos escritos en los datos de sesión. Por ejemplo, startSession carga los datos del perfil del valor de lastOffer. Un método postEvent sobrescribe lastOffer. A continuación, un segundo método postEvent sobrescribe lastOffer. El entorno de ejecución sólo conserva los datos escritos por el segundo método postEvent en los datos de sesión.

Cuando la sesión finaliza, los datos se pierden, a menos que haya creado consideraciones especiales como, por ejemplo, la utilización de un proceso Instantánea en el diagrama de flujo interactivo para escribir los datos en una tabla de base de datos. Si tiene previsto utilizar procesos Instantánea, recuerde que los nombres deben coincidir con las limitaciones de la base de datos. Por ejemplo, si sólo tiene permiso para 256 caracteres en el nombre de una columna, el nombre del par nombre-valor no debe exceder los 256 caracteres.

Acerca de la clase InteractAPI

La clase InteractAPI contiene los métodos que se utilizan para integrar el punto de encuentro con el servidor de ejecución. Las demás clases y métodos de la API de Interact dan soporte a los métodos de esta clase.

Debe compilar su implementación en interact_client.jar, que se encuentra en el directorio lib de la instalación del entorno de ejecución de Interact.

endSession

El método endSession marca el final de la sesión de ejecución. Cuando el servidor de ejecución recibe este método, el servidor de ejecución se registra en el historial, borra la memoria, etc.

```
endSession(String sessionID)
```

- **ID_sesión**: cadena exclusiva que identifica la sesión.

Si no se invoca el método endSession, las sesiones de ejecución exceden el tiempo de espera. El período de tiempo de espera puede configurarse con la propiedad sessionTimeout.

Valor de retorno

El servidor de ejecución responde al método endSession con el objeto Respose con los siguientes atributos completados:

- SessionID
- ApiVersion
- StatusCode

- AdvisoryMessages

Ejemplo

El siguiente ejemplo muestra el método endSession y cómo puede analizar la respuesta. sessionID es la misma cadena que identifica la sesión utilizada por la llamada startSession que ha iniciado esta sesión.

```
response = api.endSession(sessionId);
    // comprobar si la respuesta es satisfactoria o no
    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");
    }
    // Para los errores, deben aparecer mensajes de aviso explicando por qué
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("endSession",
      response.getAdvisoryMessages());
```

executeBatch

El método executeBatch permite ejecutar varios métodos con una sola solicitud en el servidor de ejecución.

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

- **ID_sesión**: una cadena que identifica el ID de sesión. Este ID de sesión se utiliza para todos los comandos ejecutados por esta llamada de método.
- **commandImpl[]**: una matriz de objetos CommandImpl, uno para cada comando que desea ejecutar.

El resultado de invocar este método equivale a invocar explícitamente cada método en la matriz de comandos. Este método minimiza el número de solicitudes reales en el servidor de ejecución. El servidor de ejecución ejecuta cada método en serie; para cada llamada, se capturan los errores o avisos en el objeto de respuesta correspondiente a esa llamada de método. Si se encuentra un error, executeBatch continúa con el resto de las llamadas del lote. Si la ejecución de un método da como resultado un error, el estado de nivel superior del objeto BatchResponse refleja ese error. Si no se produce ningún error, el estado de nivel superior refleja todos los avisos que se han producido. Si no se produce ningún aviso, el estado de nivel superior refleja una ejecución satisfactoria del lote.

Valor de retorno

El servidor de ejecución responde a executeBatch con un objeto BatchResponse.

Ejemplo

El siguiente ejemplo muestra cómo llamar a todos los métodos getOffer y postEvent con una sola llamada executeBatch, y una sugerencia sobre cómo manejar la respuesta.

```
/** Definir todas las variables para todos los miembros de executeBatch*/
String sessionId="MySessionID-123";
String interactionPoint = "Overview Page Banner 1";
```

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

/** compilar el comando getOffers */
Command getOffersCommand = new CommandImpl();
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(interactionPoint);
getOffersCommand.setNumberRequested(numberRequested);

/** compilar el comando postEvent */
Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEventParameters(postEventParameters);
postEventCommand.setEvent(eventName);

/** Crear la matriz de comandos */
Command[] commands =
{
  getOffersCommand,
  postEventCommand,
};

/** Realizar la llamada */
BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Procesar la respuesta según corresponda */
// El código de estado de nivel superior es un atajo para determinar si hay
// errores en la matriz de objetos de respuesta
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");
}

// Iterar por la matriz e imprimir el mensaje de los errores
for(Response response : batchResponse.getResponses())
{
  if(response.getStatusCode()!=Response.STATUS_SUCCESS)
  {
  printDetailMessageOfWarningOrError("executeBatchCommand",
    response.getAdvisoryMessages());  }
}
```

Escribir solicitudes XML executeBatch() para la API SOAP de interacción

Utilice estos pasos para escribir solicitudes XML `executeBatch()` para la API SOAP Interact.

Acerca de esta tarea

El XML de solicitud para una sola API SOAP de operación llama (`startSession`, `getOffers`, `setAudience`, `endSession`, y etc.) no se debe copiar ni pegar directamente en una llamada `executeBatch()` de varias operaciones. Los submandatos en las llamadas `executeBatch()` tienen unas estructuras de solicitud WSDL y XML ligeramente diferentes respecto a las llamadas de API de una sola operación. Las diferencias estructurales provocan respuestas de error del servidor

si los elementos XML se copian y pegan de solicitudes de API de una sola operación en varias solicitudes `executeBatch`.

Respuestas de error de ejemplo:

```
** Elemento de respuesta XML: <ns0:faultstring>org.apache.axis2.databinding.ADBException:
ID de audiencia de subelemento inesperado</ns0:faultstring>
** Excepción del servidor de interacción: java.lang.Exception: org.apache.axis2.databinding.
ADBException: ID de audiencia de subelemento inesperado en
*** ... com.unicacorp.interact.api.soap.service.v1.xsd.CommandImpl$Factory.parse
(CommandImpl.java:1917) at
```

Utilice estos pasos para escribir una solicitud XML `executeBatch()`. Puede hacer referencia a solicitudes de llamada de API una sola operación para valores de parámetro durante estos pasos, pero no copiar ni pegar elementos XML.

Procedimiento

1. Utilice una herramienta de procesamiento WSDL (por ejemplo, SoapUI) para crear una solicitud XML `executeBatch()` bien formada a partir del archivo WSDL Interact.
2. Añada submandatos a la solicitud después de la definición WSDL para elementos hijo `executeBatch()`.
3. Complete los argumentos del submandato después de la definición WSDL para elementos hijo `executeBatch()`.

getInstance

El método `getInstance` crea una instancia de la API de Interact que se comunica con el servidor de ejecución especificado.

```
getInstance(String URL)
```

Importante: Cada aplicación que escriba utilizando la API de Interact debe invocar `getInstance` para crear una instancia de un objeto InteractAPI que se correlaciona con un servidor de ejecución especificado por el parámetro de URL.

Para los grupos de servidores, si está utilizando un equilibrador de carga, utilice el nombre de host y el puerto que ha configurado con el equilibrador de carga. Si no tiene un equilibrador de carga, deberá incluir la lógica para rotar entre los servidores de ejecución disponibles.

Este método sólo es aplicable para la serialización Java con el adaptador HTTP. No hay ningún método correspondiente definido en el WSDL de SOAP. Cada implementación de cliente SOAP tiene su propia forma de establecer el URL de punto final.

- **URL**: una cadena que identifica el URL de la instancia de tiempo de ejecución. Por ejemplo, `http://localhost:7001/Interact/servlet/InteractJSService`.

Valor de retorno

El servidor de ejecución devuelve la InteractAPI.

Ejemplo

El siguiente ejemplo muestra cómo crear una instancia de un objeto InteractAPI que apunta a una instancia de servidor de ejecución que se ejecuta en la misma máquina que su punto de encuentro.

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

obtenerOfertas

El método `getOffers` permite solicitar ofertas del servidor de ejecución.

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

- **ID_sesión**: una cadena que identifica la sesión actual.
- **puntoInteracción**: una cadena que identifica el nombre del punto de interacción al que hace referencia este método.

  Nota: Este nombre debe coincidir exactamente con el nombre del punto de interacción definido en el canal interactivo.
- **númeroOfertas**: un entero que identifica el número de ofertas solicitadas.

El método `getOffers` espera el número de milisegundos definido en la propiedad `segmentationMaxWaitTimeInMS` a que finalice toda la resegmentación antes de ejecutarse. Por lo tanto, si invoca un método `postEvent` que desencadena una resegmentación o un método `setAudience` inmediatamente antes de una llamada `getOffers`, puede producirse un retardo.

Valor de retorno

El servidor de ejecución responde a `getOffers` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

Ejemplo

Este ejemplo muestra la solicitud una oferta individual para el punto de interacción Banner de la página de descripción general 1 y una forma de manejar la respuesta.

`sessionID` es la misma cadena que identifica la sesión de ejecución utilizada por la llamada `startSession` que ha iniciado esta sesión.

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

/** Realizar la llamada */
response = api.getOffers(sessionId, interactionPoint, numberRequested);

/** Procesar la respuesta según corresponda */
 // comprobar si la respuesta es satisfactoria o no
 if(response.getStatusCode() == Response.STATUS_SUCCESS)
 {
    System.out.println("getOffers call processed with no warnings or errors");

        /** Comprobar si hay alguna oferta */
        OfferList offerList=response.getOfferList();

        if(offerList.getRecommendedOffers() != null)
        {
            for(Offer offer : offerList.getRecommendedOffers())
            {
                // imprimir oferta
                System.out.println("Nombre de oferta:"+offer.getOfferName());
            }
```

```
    }
    else // basarse en la serie de oferta predeterminada
    System.out.println("Oferta predeterminada:"+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");
}
// Para los errores, deben aparecer mensajes de aviso explicando por qué
if(response.getStatusCode() != Response.STATUS_SUCCESS)
    printDetailMessageOfWarningOrError("getOffers",
  response.getAdvisoryMessages());
```

getOffersForMultipleInteractionPoints

El método `getOffersForMultipleInteractionPoints` permite solicitar ofertas del servidor de ejecución para varios IP con desduplicación.

`getOffersForMultipleInteractionPoints(String sessionID, String requestStr)`

- **ID_sesión**: una cadena que identifica la sesión actual.
- **cadenaSolicitud**: una cadena que proporciona una matriz de objetos `GetOfferRequest`.

  Cada objeto `GetOfferRequest` especifica:

  – **nombreIP**: el nombre del punto de interacción (IP) para el que el objeto solicita ofertas

  – **númeroSolicitado**: el número de ofertas exclusivas necesarias para el IP especificado

  – **atributosOferta**: requisitos de los atributos de las ofertas entregadas utilizando una instancia de `OfferAttributeRequirements`

  – **políticaDuplicación**: ID de política de duplicación de las ofertas que se entregarán

    Las políticas de duplicación determinan si se devolverán ofertas duplicadas en los distintos puntos de interacción en una única llamada de método. (*Dentro* de un punto de interacción individual, nunca se devuelven ofertas duplicadas). Actualmente, se da soporte a dos políticas de duplicación.

    - NO_DUPLICATION (Valor de ID = 1). Ninguna de las ofertas que se han incluido en las instancias de `GetOfferRequest` anteriores se incluirán en esta instancia de `GetOfferRequest` (es decir, Interact aplicará la desduplicación).
    - ALLOW_DUPLICATION (Valor de ID = 2). Se incluirán todas las ofertas que cumplan los requisitos especificados en esta instancia de `GetOfferRequest`. Las ofertas que se han incluido en las instancias de `GetOfferRequest` anteriores no se reconciliarán.

  El orden de las solicitudes en el parámetro de matriz es también el orden de prioridad cuando se entregan las ofertas.

  Por ejemplo, supongamos que los IP en la solicitud son IP1 e IP2, que no se permiten ofertas duplicadas (un ID de política de duplicación = 1) y que cada uno solicita cada dos ofertas. Si Interact encuentra las ofertas A, B y C para IP1 y las ofertas A y D para IP2, la respuesta contendrá las ofertas A y B para IP1, y sólo la oferta D para IP2.

  Asimismo, tenga en cuenta que cuando el ID de política de duplicación es 1, las ofertas que se hayan entregado con un IP con una prioridad mayor no se entregará con este IP.

El método getOffersForMultipleInteractionPoints espera el número de milisegundos definido en la propiedad segmentationMaxWaitTimeInMS a que finalice toda la resegmentación antes de ejecutarse. Por lo tanto, si invoca un método postEvent que desencadena una resegmentación o un método setAudience inmediatamente antes de una llamada getOffers, puede producirse un retardo.

Valor de retorno

El servidor de ejecución responde a getOffersForMultipleInteractionPoints con un objeto de respuesta con los siguientes atributos completados:

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

Ejemplo

```
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) {
    // Comprobar si hay alguna oferta
  OfferList[] allOfferLists = response.getAllOfferLists();
   if (allOfferLists != null) {
     for (OfferList ol : allOfferLists) {
    System.out.println
("Las ofertas siguientes se entregan para el punto de
               interacción " + 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());
 }
```

Tenga en cuenta que la sintaxis de requestStr es la siguiente:

```
requests_for_IP[<requests_for_IP]
```

donde

```
<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
```

En el ejemplo anterior, `requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))})` significa que, para el punto de interacción denominado IP1, se entregan como máximo 5 ofertas diferentes que no pueden devolverse para otros puntos de interacción durante la misma llamada de método. Las 5 ofertas deben tener un atributo numérico denominado `attr1` con el valor 1, y deben tener un atributo de cadena denominado `attr2` con el valor *value2*. Aparte de esas 5 ofertas, un máximo de 3 deben tener un atributo de cadena denominado `attr3` con el valor *value3*, y un máximo de 3 deben tener un atributo numérico denominado `attr4` con el valor *4*.

Los tipos de atributo permitidos son numéricos, cadena, y fecha y hora; el formato de un valor de atributo de fecha y hora debe ser `MM/dd/yyyy HH:mm:ss`. Para recuperar las ofertas devueltas, utilice el método `Response.getAllOfferLists()`. Para entender mejor la sintaxis, el ejemplo de `setGetOfferRequests` crea la misma instancia de `GetOfferRequests` utilizando objetos Java, que es la opción preferida.

getProfile

El método `getProfile` permite recuperar la información de perfil y temporal sobre el visitante que visita el punto de encuentro.

`getProfile(String sessionID)`

- **ID_sesión**: una cadena que identifica el ID de sesión.

Valor de retorno

El servidor de ejecución responde a `getProfile` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

Ejemplo

El siguiente es un ejemplo de utilización de `getProfile` y una forma de manejar la respuesta.

`sessionID` es la misma cadena que identifica la sesión utilizada por la llamada `startSession` que ha iniciado esta sesión.

```
response = api.getProfile(sessionId);
/** Procesar la respuesta según corresponda */
    // comprobar si la respuesta es satisfactoria o no
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("getProfile call processed with no warnings or errors");
        // Imprimir el perfil - es sólo una matriz de objetos NameValuePair
        for(NameValuePair nvp : response.getProfileRecord())
        {
            System.out.println("Nombre:"+nvp.getName());
            if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
            {
                System.out.println("Valor:"+nvp.getValueAsDate());
            }
            else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
            {
                System.out.println("Valor:"+nvp.getValueAsNumeric());
```

```
            }
            else
            {
                System.out.println("Valor:"+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");
    }
    // Para los errores, deben aparecer mensajes de aviso explicando por qué
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getProfile",
      response.getAdvisoryMessages());
```

getVersion

El método `getVersion` devuelve la versión de la implementación actual del servidor de ejecución de Interact.

```
getVersion()
```

Se recomienda utilizar este método cuando inicialice el punto de encuentro con la API de Interact.

Valor de retorno

El servidor de ejecución responde a `getVersion` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- StatusCode

Ejemplo

Este ejemplo muestra una forma simple de invocar `getVersion` y procesar los resultados.

```
response = api.getVersion();
/** Procesar la respuesta según corresponda */
    // comprobar si la respuesta es satisfactoria o no
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("llamada a getVersion procesada sin avisos o errores");
        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");
    }

    // Para los errores, deben aparecer mensajes de aviso explicando por qué
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getVersion",
      response.getAdvisoryMessages());
```

postEvent

El método `postEvent` permite ejecutar cualquier evento definido en el canal interactivo.

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

- **sessionID**: una cadena que identifica el ID de sesión.
- **eventName**: una cadena que identifica el nombre del evento.

  Nota: El nombre del evento debe coincidir con el nombre del evento tal como está definido en el canal interactivo. Este nombre es sensible a mayúsculas y minúsculas.
- **eventParameters**. Los objetos `NameValuePairImpl` identifican los parámetros que deben pasarse con el evento. Estos valores se almacenan en los datos de sesión.

  Si este evento desencadena la resegmentación, debe asegurarse de que todos los datos necesarios para los diagramas de flujo interactivos estén disponibles en los datos de sesión. Si alguno de estos valores no se ha completado mediante las acciones anteriores (por ejemplo, a partir de `startSession` o `setAudience`, o al cargar la tabla de perfil), debe incluir un eventParameter para cada valor que falte. Por ejemplo, si ha configurado todas las tablas de perfiles para que se carguen en la memoria, debe incluir un NameValuePair para los datos temporales necesarios para los diagramas de flujo interactivos.

  Si utiliza más de un nivel de audiencia, lo más probable es que tenga distintos conjuntos de eventParameters para cada nivel de audiencia. Debe incluir alguna lógica para asegurarse de que está seleccionando el conjunto de parámetros correcto para el nivel de audiencia.

  Importante: Si este evento se registra en un historial de respuestas, debe pasar el código de tratamiento para la oferta. Debe definir el nombre de NameValuePair como "`UACIOfferTrackingCode`".

  Sólo puede pasar un código de tratamiento por evento. Si no pasa el código de tratamiento para un contacto de oferta, Interact registra un contacto de oferta para cada oferta en la última lista de ofertas recomendada. Si no pasa el código de tratamiento para una respuesta, Interact devuelve un error.
- Existen otros parámetros reservados que se utilizan con `postEvent` y otros métodos, que se describen más adelante en esta sección.

Toda solicitud de resegmentación o escritura en el historial de contactos o de respuestas no espera una respuesta.

La resegmentación no borra los resultados de una segmentación anterior para el nivel de audiencia actual. Puede utilizar el parámetro `UACIExecuteFlowchartByName` para definir diagramas de flujo específicos para ejecutar. El método `getOffers` espera a que la resegmentación finalice antes de ejecutarse. Por lo tanto, si llama a un método `postEvent`, que desencadena una resegmentación inmediatamente antes de una llamada a `getOffers`, podría haber una retardo.

Valor de retorno

El servidor de ejecución responde a `postEvent` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion

- SessionID
- StatusCode

Ejemplo

El siguiente ejemplo de `postEvent` muestra el envío de nuevos parámetros para un evento que desencadena la resegmentación, y una forma de manejar la respuesta.

`sessionID` es la misma cadena que identifica la sesión utilizada por la llamada `startSession` que ha iniciado esta sesión.

```
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
};

/** Realizar la llamada */
response = api.postEvent(sessionId, eventName, postEventParameters);

/** Procesar la respuesta según corresponda */
    // comprobar si la respuesta es satisfactoria o no
    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");
    }

    // Para los errores, deben aparecer mensajes de aviso explicando por qué
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("postEvent",
      response.getAdvisoryMessages());
```

setAudience

El método `setAudience` permite establecer el ID y el nivel de audiencia para un visitante.

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

- **ID_sesión**: una cadena que identifica el ID de sesión.
- **IDAudiencia**: matriz de objetos `NameValuePairImpl` que define el ID de audiencia.
- **nivelAudiencia**: cadena que define el nivel de audiencia.
- **parámetros**: objetos `NameValuePairImpl` que identifican los parámetros que se deben pasar con `setAudience`. Estos valores se almacenan en los datos de sesión y se pueden utilizar para la segmentación.

  Debe tener un valor para cada columna de su perfil. Este es un superconjunto de todas las columnas de todas las tablas definidas para el canal interactivo y los datos en tiempo real. Si ya ha completado todos los datos de sesión con `startSession` o `postEvent`, no es necesario enviar nuevos parámetros.

El método `setAudience` desencadena una resegmentación. El método `getOffers` espera a que la resegmentación finalice antes de ejecutarse. Por lo tanto, si invoca un método `setAudience` inmediatamente antes de una llamada `getOffers`, puede producirse un retardo.

El método `setAudience` también carga los datos del perfil para el ID de audiencia. Puede utilizar el método `setAudience` para forzar que se vuelvan a cargar los mismos datos del perfil cargados por el método `startSession`.

Valor de retorno

El servidor de ejecución responde a `setAudience` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Ejemplo

En este ejemplo, el nivel de audiencia permanece igual, pero el ID cambia, como si un usuario anónimo inicia una sesión y pasa a ser conocido.

`sessionId` y `audienceLevel` son las mismas cadenas que identifican la sesión y el nivel de audiencia utilizados por la llamada `startSession` que ha iniciado esta sesión.

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

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

/** Los parámetros también pueden pasarse. Para este ejemplo, no hay parámetros,
* por lo tanto se pasa null */
NameValuePair[] noParameters=null;

/** Realizar la llamada */
response = api.setAudience(sessionId, newAudienceId, audienceLevel, noParameters);

 /** Procesar la respuesta según corresponda */
    // comprobar si la respuesta es satisfactoria o no
    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");
    }

    // Para los errores, deben aparecer mensajes de aviso explicando por qué
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setAudience",
      response.getAdvisoryMessages());
```

setDebug

El método setDebug le permite establecer el nivel de detalle de registro para todas las rutas de código de la sesión.

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

- **ID_sesión**: cadena que identifica el ID de sesión.
- **debug**: booleano que habilita o inhabilita la información de depuración. Los valores válidos son true o false. Si es true, Interact registra información de depuración en el registro del servidor de ejecución.

Valor de retorno

El servidor de ejecución responde a setDebug con un objeto Response con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Ejemplo

El ejemplo siguiente muestra el nivel de depuración de la sesión.

sessionID es la misma cadena que identifica la sesión utilizada por la llamada startSession que ha iniciado esta sesión.

```
boolean newDebugFlag=false;
/** realizar la llamada */
response = api.setDebug(sessionId, newDebugFlag);

/** Procesar la respuesta según corresponda */
    // comprobar si la respuesta es satisfactoria o no
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("llamada a setDebug procesada sin avisos o errores");
    }
```

```
else if(response.getStatusCode() == Response.STATUS_WARNING)
{
    System.out.println("llamada a setDebug procesada con un aviso");
}
else
{
    System.out.println("llamada a setDebug procesada con un error");
}

// Para los errores, deben aparecer mensajes de aviso explicando por qué
if(response.getStatusCode() != Response.STATUS_SUCCESS)
    printDetailMessageOfWarningOrError("setDebug",
  response.getAdvisoryMessages());
```

startSession

El método `startSession` crea y define una sesión de ejecución.

```
startSession(String ID_sesión,
      boolean relyOnExistingSession,
      boolean debug,
      String canalInteractivo,
      NameValuePairImpl[] IDAudiencia,
      String nivelAudiencia,
      NameValuePairImpl[] parámetros)
```

`startSession` puede desencadenar hasta cinco acciones:

- crear una sesión de ejecución.
- cargar los datos de perfil del visitante para el nivel de audiencia actual en la sesión de ejecución, incluidas las tablas de dimensiones marcadas para la carga en la correlación de tablas definida para el canal interactivo.
- desencadenar segmentación, ejecutando todos los diagramas de flujo interactivos para el nivel de audiencia actual.
- cargar datos de supresión de oferta en la sesión, si la propiedad `enableOfferSuppressionLookup` está establecida en true.
- cargar los datos de anulación de puntuaciones en la sesión, si la propiedad `enableScoreOverrideLookup` está establecida en true.

El método `startSession` requiere los parámetros siguientes:

- **ID_sesión**: cadena que identifica el ID de sesión. Debe definir el ID de sesión. Por ejemplo, podría utilizar una combinación de ID de cliente y marca de hora.

  Para definir qué compone una sesión de ejecución, se debe especificar el ID de sesión. Este valor lo gestiona el cliente. El cliente debe sincronizar todas las llamadas a método para el mismo ID de sesión. El comportamiento para las llamadas a la API simultáneas con el mismo ID de sesión no está definido.
- **relyOnExistingSession**: booleano que define si esta sesión utiliza una sesión nueva o existente. Los valores válidos son `true` o `false`. Si es `true`, debe proporcionar un ID de sesión existente con el método `startSession`. Si es `false`, debe proporcionar un nuevo ID de sesión.

  Si se establece `relyOnExistingSession` en `true` y ya existe una sesión, el entorno de ejecución utiliza los datos de sesión existentes y no vuelve a cargar datos o desencadena segmentación. Si la sesión no existe, el entorno de ejecución crea una nueva sesión, lo que incluye cargar datos y desencadenar segmentación. Establecer `relyOnExistingSession` en true y utilizarlo con todas las llamadas a `startSession` resulta útil si el punto de encuentro tiene una duración de sesión superior a la sesión de ejecución. Por ejemplo, una sesión de sitio web está activa durante 2 horas, pero la sesión de ejecución solo está activa 20 minutos.

Si llama a `startSession` dos veces con el mismo ID de sesión, todos los datos de sesión de la primera llamada a `startSession` se perderán si `relyOnExistingSession` es false.

- **debug**: booleano que habilita o inhabilita la información de depuración. Los valores válidos son `true` o `false`. Si es true, Interact registra la información de depuración en los registros del servidor de ejecución. El indicador de depuración se establece individualmente para cada sesión. Por lo tanto, puede rastrear los datos de depuración para una sesión individual.
- **canalInteractivo**: cadena que define el nombre del canal interactivo al que hace referencia esta sesión. Este nombre debe coincidir con el nombre del canal interactivo tal como está definido exactamente en Campaign.
- **IDAudiencia**: matriz de objetos NameValuePairImpl donde los nombres deben coincidir con los nombres de columna física de cualquier tabla que contenga el ID de audiencia.
- **nivelAudiencia**: cadena que define el nivel de audiencia.
- **parámetros**: objetos NameValuePairImpl que identifican los parámetros que es necesario pasar con `startSession`. Estos valores se almacenan en los datos de sesión y se pueden utilizar para la segmentación.

  Si tiene varios diagramas de flujo interactivos para el mismo nivel de audiencia, debe incluir un superconjunto de todas las columnas de todas las tablas. Si configura el tiempo de ejecución para cargar la tabla de perfil, y la tabla de perfil contiene todas las columnas que necesita, no es necesario pasar ningún parámetro, a menos que desee sobrescribir los datos de la tabla de perfil. Si la tabla de perfil contiene un subconjunto de las columnas necesarias, debe incluir las columnas que faltan como parámetros.

Si `IDAudiencia` o `nivelAudiencia` no son válidos y `relyOnExistingSession` es false, la llamada a `startSession` fallará. Si `interactiveChannel` no es válido, `startSession` fallará, independientemente de si `relyOnExistingSession` es true o false.

Si `relyOnExistingSession` es true y realiza una segunda llamada a `startSession` utilizando el mismo `ID_sesión`, pero la primera sesión ha caducado, Interact creará una nueva sesión.

Si `relyOnExistingSession` es true y realiza una segunda llamada a `startSession` utilizando el mismo `ID_sesión` pero un `IDAudiencia` o `nivelAudiencia` distinto, el servidor de ejecución cambiará la audiencia para la sesión existente.

Si `relyOnExistingSession` es true y realiza una segunda llamada a `startSession` utilizando el mismo `ID_sesión` pero un `canalInteractivo` distinto, el servidor de ejecución creará una nueva sesión.

Valor de retorno

El servidor de ejecución responde a `startSession` con un objeto Response que tiene los siguientes atributos completados:

- AdvisoryMessages (si StatusCode no es igual a 0)
- ApiVersion
- SessionID
- StatusCode

Ejemplo

El ejemplo siguiente muestra una forma de llamar a `startSession`.

```
String sessionId="MySessionID-123";
String audienceLevel="Cliente";
NameValuePair custId = new NameValuePairImpl();
custId.setName("IDCliente");
custId.setValueAsNumeric(1.0);
custId.setValueDataType(NameValuePair.DATA_TYPE_NUMERIC);
NameValuePair[] initialAudienceId = { custId };
boolean relyOnExistingSession=false;
boolean initialDebugFlag=true;
String interactiveChannel="Sitio web Cuentas";
NameValuePair parm1 = new NameValuePairImpl();
parm1.setName("SerieBúsqueda");
parm1.setValueAsString("");
parm1.setValueDataType(NameValuePair.DATA_TYPE_STRING);

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

NameValuePair parm3 = new NameValuePairImpl();
parm3.setName("Navegador");
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("TemaPágina");
parm6.setValueAsString("");
parm6.setValueDataType(NameValuePair.DATA_TYPE_STRING);

/** Especificación de parámetros (opcional) */
NameValuePair[] initialParameters = { parm1,
 parm2,
 parm3,
 parm4,
 parm5,
 parm6
 };

/** Realizar la llamada */
response = api.startSession(sessionId, relyOnExistingSession, initialDebugFlag,
  interactiveChannel, initialAudienceId, audienceLevel, initialParameters);

/** Procesar la respuesta según corresponda */
processStartSessionResponse(response);
```

`processStartSessionResponse` es un método que maneja el objeto de respuesta devuelto por `startSession`.

```
public static void processStartSessionResponse(Response response)
{
  // comprobar si la respuesta es satisfactoria o no
  if(response.getStatusCode() == Response.STATUS_SUCCESS)
```

```
{
 System.out.println("llamada a startSession procesada sin avisos o errores");
}
 else if(response.getStatusCode() == Response.STATUS_WARNING)
{
 System.out.println("llamada a startSession procesada con un aviso");
}
 else
{
 System.out.println("llamada a startSession procesada con un error");
}

// Para los errores, deben aparecer mensajes de aviso explicando por qué
if(response.getStatusCode() != Response.STATUS_SUCCESS)
 printDetailMessageOfWarningOrError("StartSession",
     response.getAdvisoryMessages());}
```

Desduplicación de la oferta para varios atributos de oferta

Mediante la API (interfaz de programación de aplicaciones) de Interact, dos llamadas de API proporcionan las ofertas `getOffers` y `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` puede impedir la devolución de ofertas duplicadas a nivel de *OfferID*, pero no puede desduplicar ofertas para varias categorías de oferta. Así, por ejemplo, para que Interact devuelva una única oferta de cada categoría de oferta, anteriormente era necesaria una solución temporal. Con la adición de dos parámetros a la llamada de API `startSession`, ahora es posible la desduplicación de ofertas para varios atributos de oferta como, por ejemplo, la categoría.

Esta lista resume los parámetros que se han añadido a la llamada de API `startSession`. Para obtener más información sobre estos parámetros u otros aspectos de la API de Interact, consulte la *Guía del administrador de IBM Interact* o los archivos Javadoc que se incluyen con la instalación de Interact en `<Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Para crear una llamada de API `startSession` con desduplicación de oferta para que las llamadas `getOffer` subsiguientes devuelvan una única oferta de cada categoría, debe incluir el parámetro `UACIOfferDedupeAttribute` como parte de la llamada de API. Puede especificar un parámetro con el formato `nombre,valor,tipo`, como se muestra a continuación:

  `UACIOfferDedupeAttribute,<nombre_atributo>,string`

  En este ejemplo, sustituiría *<nombre_atributo>* por el nombre del atributo de oferta que desee utilizar como criterio para la desduplicación, por ejemplo, Categoría.

  Nota: Interact examina las ofertas que tienen el mismo valor de atributo que especifique (por ejemplo, Categoría) y elimina las ofertas duplicadas excepto la oferta que tenga la mayor puntuación. Si las ofertas que tienen el atributo duplicado también tienen puntuaciones idénticas, Interact devuelve una selección aleatoria entre las ofertas coincidentes.

- `UACINoAttributeDedupeIfFewerOf`. Cuando incluye el parámetro `UACIOfferDedupeAttribute` en la llamada a `startSession`, también puede definir el parámetro `UACINoAttributeDedupeIfFewerOf` para especificar el comportamiento cuando la lista de ofertas después de la desduplicación ya no contiene ofertas suficientes para satisfacer la solicitud original.

Por ejemplo, si define `UACIOfferDedupeAttribute` para utilizar la categoría de oferta para desduplicar ofertas y la llamada a getOffers posterior solicita que se devuelvan ocho ofertas, la desduplicación puede producir menos de ocho ofertas elegibles. En ese caso, el establecimiento el parámetro `UACINoAttributeDedupeIfFewerOf` en true dará como resultado la adición de algunos de los duplicados a la lista de ofertas elegibles para satisfacer el número solicitado de ofertas. En este ejemplo, si establece el parámetro en false, el número de ofertas que se devuelven será menor que el número solicitado.

`UACINoAttributeDedupeIfFewerOf` se establece en true de forma predeterminada.

Por ejemplo, supongamos que ha especificado como parámetro `startSession` que el criterio de desduplicación sea la oferta Categoría, como se muestra a continuación:

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

Utilizados conjuntamente, estos parámetros hacen que Interact desduplique las ofertas basándose en el atributo de oferta "Categoría" y que se devuelvan solamente las ofertas desduplicadas, aunque el número resultante de ofertas sea menor que el solicitado (`UACINoAttributeDedupeIfFewerOffer` es false).

Cuando emite una llamada de API `getOffers`, el conjunto original de ofertas elegibles puede incluir estas ofertas:

- Category=Electronics: oferta A1 con una puntuación de 100 y oferta A2 con una puntuación de 50.
- Category=Smartphones: oferta B1 con una puntuación de 100, oferta B2 con una puntuación de 80 y oferta B3 con una puntuación de 50.
- Category=MP3Players: oferta C1 con una puntuación de 100 y oferta C2 con una puntuación de 50.

En este caso, había dos ofertas duplicadas que coinciden con la primera categoría, tres ofertas duplicadas que coinciden con la segunda categoría y dos ofertas duplicadas que coinciden con la tercera categoría. Las ofertas que se devuelven son las ofertas con mayor puntuación de cada categoría, es decir, las ofertas A1, B1 y C1.

Si la llamada de API `getOffers` solicita seis ofertas, este ejemplo establece `UACINoAttributeDedupeIfFewerOffer` en false para devolver solamente tres ofertas.

Si la llamada de API `getOffers` solicita seis ofertas y este ejemplo omite el parámetro `UACINoAttributeDedupeIfFewerOffer` o lo establece explícitamente en true, algunas de las ofertas duplicadas se incluirán en el resultado para satisfacer el número solicitado.

Parámetros reservados

Hay algunos parámetros reservados que se utilizan con la API de Interact. Algunos son necesarios para el servidor de ejecución y otros se pueden utilizar para características adicionales.

Características de postEvent

Característica	Parámetro	Descripción
Registrar en tabla personalizada	UACICustomLoggerTableName	Nombre de una tabla del origen de datos de tablas de ejecución. Si proporciona este parámetro con un nombre de tabla válido, el entorno de ejecución graba todos los datos de sesión en la tabla seleccionada. Todos los nombres de columna de la tabla que coinciden con los datos de sesión NameValuePair se completan. El entorno de ejecución completa con un nulo las columnas que no coincidan con el par nombre-valor de sesión. Puede gestionar el proceso que graba en la base de datos con las propiedades de configuración customLogger.
Varios tipos de respuesta	UACILogToLearning	Un entero con el valor 1 o 0. 1 indica que el entorno de ejecución deberá registrar el evento como aceptado en el sistema de aprendizaje o habilitar la supresión de ofertas dentro de una sesión. 0 indica que el entorno de ejecución no deberá registrar el evento como aceptado en el sistema de aprendizaje o habilitar la supresión de ofertas dentro de una sesión. Este parámetro le permite crear varios métodos postEvent registrando distintos tipos de respuesta sin influenciar el aprendizaje. No necesita definir este parámetro para eventos establecidos para registrar un contacto, una aceptación o un rechazo. Debe utilizar este parámetro junto con UACIResponseTypeCode. Si no define UACILOGTOLEARNING, el entorno de ejecución asume el valor predeterminado de 0 (a menos que el evento desencadene un contacto, aceptación o rechazo de registro).
	UACIResponseTypeCode	Valor que representa un código de tipo de respuesta. El valor debe ser una entrada válida en la tabla UA_UsrResponseType
Seguim. respuestas	UACIOfferTrackingCode	Código de tratamiento de la oferta. Debe definir este parámetro si el evento se registra en el historial de respuestas o contactos. Sólo puede pasar un código de tratamiento por evento. Si no pasa el código de tratamiento para un contacto de oferta, el entorno de ejecución registra un contacto de oferta para cada oferta en la última lista recomendada de ofertas. Si no pasa el código de tratamiento para una respuesta, el entorno de ejecución devuelve un error. Si configura el seguimiento de respuestas de sesiones cruzadas, puede utilizar el parámetro UACIOfferTrackingcodeType para definir qué tipo de código de seguimiento utiliza que no es el código de tratamiento.

Característica	Parámetro	Descripción
Seguimiento de respuestas de sesiones cruzadas	UACIOfferTrackingCodeType	Número que define el tipo de código de seguimiento. 1 es el código de tratamiento predeterminado y 2 es el código de oferta. Todos los códigos deben ser entradas válidas de la tabla UACI_TrackingType. Puede añadir otros códigos personalizadas a esta tabla.
Ejecución de diagrama de flujo específica	UACIExecuteFlowchartByName	Si define este parámetro para cualquier método que desencadene segmentación (startSession, setAudience o postEvent que desencadena re-segmentación), en lugar de ejecutar todos los diagramas de flujo para el nivel de audiencia actual, Interact ejecuta solo los diagramas de flujo con nombre. Puede proporcionar una lista de diagramas de flujo separados por un carácter de barra vertical (|).

Parámetros reservados del entorno de ejecución

El entorno de ejecución utiliza los siguientes parámetros reservados. No utilice estos parámetros para sus parámetros de evento.

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

Acerca de la clase AdvisoryMessage

La clase advisoryMessage contiene métodos que definen el objeto de mensaje de aviso. El objeto de mensaje de aviso está contenido en el objeto de respuesta. Cada método de la InteractAPI devuelve un objeto de respuesta.(Excepto el método executeBatch, que devuelve un objeto batchResponse.)

Si se produce un error o un aviso, el servidor de Interact completa el objeto de mensaje de aviso. El objeto de mensaje de aviso contiene los siguientes atributos:

- **DetailMessage**: una descripción detallada del mensaje de aviso. Este atributo no estará disponibles para todos los mensajes de aviso. Si está disponible, DetailMessage no puede localizarse.
- **Message**: una descripción breve del mensaje de aviso.
- **MessageCode**: un número de código para el mensaje de aviso.
- **StatusLevel**: un número de código para la gravedad del mensaje de aviso.

Los objetos advisoryMessage se recuperan utilizando el método getAdvisoryMessages.

getDetailMessage

El método getDetailMessage devuelve la descripción detallada de un objeto de mensaje de aviso. No todos los mensajes tienen un mensaje detallado.

```
getDetailMessage()
```

Valor de retorno

El objeto de mensaje de aviso devuelve una cadena.

Ejemplo

```
// Para los errores, deben aparecer mensajes de aviso explicando por qué
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
  for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
  System.out.println(msg.getMessage());
  // Algunos mensajes de aviso pueden tener detalles adicionales:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessage

El método getMessage devuelve una descripción breve de un objeto de mensaje de aviso.

```
getMessage()
```

Valor de retorno

El objeto de mensaje de aviso devuelve una cadena.

Ejemplo

El siguiente método imprime el mensaje y el mensaje detallado de un objeto AdvisoryMessage.

```
// Para los errores, deben aparecer mensajes de aviso explicando por qué
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
  for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
  System.out.println(msg.getMessage());
  // Algunos mensajes de aviso pueden tener detalles adicionales:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessageCode

El método getMessageCode devuelve el código de error interno asociado con un objeto de mensaje de aviso si el nivel de estado es 2 (STATUS_LEVEL_ERROR).

```
getMessageCode()
```

Valor de retorno

El objeto AdvisoryMessage devuelve un entero.

Ejemplo

El siguiente método imprime el código de mensaje de un 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

El método `getStatusLevel` devuelve el nivel de estado de un objeto de mensaje de aviso.

```
getStatusLevel()
```

Valor de retorno

El objeto de mensaje de aviso devuelve un entero.

- 0 - STATUS_LEVEL_SUCCESS: el método invocado ha finalizado sin errores.
- 1 - STATUS_LEVEL_WARNING: el método invocado ha finalizado con al menos un aviso (pero sin errores).
- 2 - STATUS_LEVEL_ERROR: el método invocado no ha finalizado satisfactoriamente y tiene al menos un error.

Ejemplo

El siguiente método imprime el nivel de estado de un objeto AdvisoryMessage.

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

Acerca de la clase AdvisoryMessageCode

La clase `advisoryMessageCode` contiene métodos que definen los códigos de mensaje de aviso. Puede recuperar los códigos de mensajes de aviso con el método `getMessageCode`.

Códigos de mensaje de aviso

Puede recuperar los códigos de mensajes de aviso con el método `getMessageCode`.

Esta tabla muestra y describe los códigos de mensaje de aviso.

Cód.	Texto del mensaje	Descripción
1	INVALID_SESSION_ID	El ID de sesión no hace referencia a una sesión válida.
2	ERROR_TRYING_TO_ABORT_ SEGMENTATION	Se ha producido un error cuando se intentaba abortar la segmentación durante una llamada a `endSession`.
3	INVALID_INTERACTIVE_CHANNEL	El argumento pasado para el canal interactivo no hace referencia a un canal interactivo válido.
4	INVALID_EVENT_NAME	El argumento pasado para el evento no hace referencia a un evento válido para el canal interactivo actual.

Cód.	Texto del mensaje	Descripción
5	INVALID_INTERACTION_POINT	El argumento pasado para el punto de interacción no hace referencia a un punto de interacción válido para el canal interactivo actual.
6	ERROR_WHILE_MAKING_INITIAL_ SEGMENTATION_REQUEST	Se ha producido un error al enviar una solicitud de segmentación.
7	SEGMENTATION_RUN_FAILED	La segmentación se ha ejecutado parcialmente, pero ha dado como resultado un error.
8	PROFILE_LOAD_FAILED	El intento de cargar las tablas de dimensiones o perfiles ha fallado.
9	OFFER_SUPPRESSION_LOAD_FAILED	El intento de cargar la tabla de supresión de ofertas ha fallado.
10	COMMAND_METHOD_ UNRECOGNIZED	El método de comando especificado para un comando en una llamada a `executeBatch` no es válido.
11	ERROR_TRYING_TO_POST_EVENT _PARAMETERS	Se ha producido un error al publicar los parámetros de evento.
12	LOG_SYSTEM_EVENT_EXCEPTION	Se ha producido una excepción al intentar enviar el evento de sistema (Finalizar sesión, Obtener oferta, Obtener perfil, Establecer audiencia, Establecer depuración o Iniciar sesión) para el registro.
13	LOG_USER_EVENT_EXCEPTION	Se ha producido una excepción al intentar enviar el evento de registro para el registro.
14	ERROR_TRYING_TO_LOOK_UP_EVENT	Se ha producido un error al intentar buscar el nombre de evento.
15	ERROR_TRYING_TO_LOOK_UP _INTERACTIVE_CHANNEL	Se ha producido un error al intentar buscar el nombre de canal interactivo.
16	ERROR_TRYING_TO_LOOK_UP _INTERACTION_POINT	Se ha producido un error al intentar buscar el nombre de punto de interacción.
17	RUNTIME_EXCEPTION_ ENCOUNTERED	Se ha encontrado excepción de tiempo de ejecución inesperada.
18	ERROR_TRYING_TO_EXECUTE_ ASSOCIATED_ACTION	Se ha producido error al intentar ejecutar la acción asociada (Desencadenar resegmentación, Registrar contacto de oferta, Registrar aceptación de oferta, Registrar rechazo de oferta).
19	ERROR_TRYING_RUN_FLOWCHART	Se ha producido un error al intentar ejecutar el diagrama de flujo.
20	FLOWCHART_FAILED	Una ejecución de diagrama de flujo ha fallado.
21	FLOWCHART_ABORTED	Una ejecución de diagrama de flujo ha terminado anormalmente.
22	FLOWCHART_NEVER_RUN	El diagrama de flujo especificado nunca se ha ejecutado.
23	FLOWCHART_STILL_RUNNING	El diagrama de flujo todavía está ejecutándose.

Cód.	Texto del mensaje	Descripción
24	ERROR_WHILE_READING_ PARAMETERS	Se ha producido un error al leer los parámetros.
25	ERROR_WHILE_LOADING_ RECOMMENDED_OFFERS	Se ha producido un error al cargar las ofertas recomendadas.
26	ERROR_WHILE_LOGGING_DEFAULT _TEXT_STATISTICS	Se ha producido un error al registrar las estadísticas de texto predeterminadas (el número de veces que aparece la cadena predeterminada para el punto de interacción).
27	SCORE_OVERRIDE_LOAD_FAILED	La tabla de anulación de puntuación no se ha podido cargar.
28	NULL_AUDIENCE_ID	El identificador de audiencia está vacío.
29	UNRECOGNIZED_AUDIENCE_LEVEL	Se ha especificado un nivel de audiencia no reconocido.
30	MISSING_AUDIENCE_FIELD	Falta un campo de audiencia.
31	INVALID_AUDIENCE_FIELD_TYPE	Se ha especificado un tipo de campo de audiencia no válido.
32	UNSUPPORTED_AUDIENCE_FIELD _TYPE	Tipo de campo de audiencia no soportado.
33	TIMEOUT_REACHED_ON_GET_ OFFERS_CALL	La llamada a `getOffers` ha alcanzado el tiempo de espera sin devolver ofertas.
34	INTERACT_INITIALIZATION_NOT_ COMPLETED_SUCCESSFULLY	La inicialización del servidor de ejecución no ha finalizado satisfactoriamente.
35	SESSION_ID_UNDEFINED	El identificador de sesión está sin definir.
36	INVALID_NUMBER_OF_OFFERS_ REQUESTED	Se ha solicitado un número de ofertas no válido.
37	NO_SESSION_EXIST_BUT_WILL_ CREATE_NEW_ONE	No existía ninguna sesión, pero se ha creado una.
38	AUDIENCE_ID_NOT_FOUND_IN_ PROFILE_TABLE	El identificador de audiencia especificado no está en la tabla de perfil.
39	LOG_CUSTOM_LOGGER_EVENT_ EXCEPTION	Se ha producido una excepción al intentar enviar el evento de datos de registro personalizado.
40	SPECIFIED_FLOWCHART_FOR_ EXECUTION_DOES_NOT_EXIST	El diagrama de flujo especificado no puede ejecutarse porque no existe.
41	AUDIENCE_NOT_DEFINED_IN_ CONFIGURATION	La audiencia especificada no está definida en la configuración.

Acerca de la clase BatchResponse

La clase `BatchResponse` contiene métodos que definen los resultados del método `executeBatch`.

El objeto de respuesta por lotes contiene los siguientes atributos:

- **BatchStatusCode**: el valor máximo de Código de estado para todas las respuestas solicitadas por el método `executeBatch`.
- **Responses**: una matriz de los objetos de respuesta solicitados por el método `executeBatch`.

getBatchStatusCode

El método getBatchStatusCode devuelve el código de estado máximo de la matriz de comandos ejecutados por el método executeBatch.

```
getBatchStatusCode()
```

Valor de retorno

El método getBatchStatusCode devuelve un entero.

- 0 - STATUS_SUCCESS: el método invocado ha finalizado sin errores.
- 1 - STATUS_WARNING: el método invocado ha finalizado con al menos un aviso (pero sin errores).
- 2 - STATUS_ERROR: el método invocado no ha finalizado satisfactoriamente y tiene al menos un error.

Ejemplo

El siguiente código de muestra proporciona un ejemplo de cómo recuperar el BatchStatusCode.

```
// El código de estado de nivel superior es un atajo para determinar si hay
// errores en la matriz de objetos de respuesta
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");
}

// Iterar por la matriz e imprimir el mensaje de los errores
for(Response response : batchResponse.getResponses())
{
  if(response.getStatusCode()!=Response.STATUS_SUCCESS)
  {
      printDetailMessageOfWarningOrError("executeBatchCommand",
      response.getAdvisoryMessages());  }
}
```

getResponses

El método getResponses devuelve la matriz de objetos de respuesta correspondiente a la matriz de comandos ejecutados por el método executeBatch.

```
getResponses()
```

Valor de retorno

El método getResponses devuelve una matriz de objetos Response.

Ejemplo

El siguiente ejemplo selecciona todas las respuestas e imprime los mensajes de aviso si el comando no ha sido satisfactorio.

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

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

Acerca de la interfaz Comando

El método `executeBatch` requiere que pase una matriz de objetos que implemente la interfaz Comando. Debe utilizar la implementación predeterminada, `CommandImpl` para pasar objetos Comando.

La tabla siguiente lista el comando, el método de la clase InteractAPI que representa el comando, y los métodos de la interfaz Comando que se deben utilizar para cada comando. No es necesario incluir un ID de sesión porque el método `executeBatch` ya incluye el ID de sesión.

Comando	Método de la API de Interact	Métodos de la interfaz Comando
COMMAND_ENDSESSION	endSession	Ninguno.
COMMAND_GETOFFERS	obtenerOfertas	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	Ninguno.
COMMAND_GETVERSION	getVersion	Ninguno.
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

El método `setAudienceID` define el IDAudiencia para los comandos `setAudience` y `startSession`.

`setAudienceID(`*`IDAudiencia`*`)`

- **IDAudiencia**: matriz de objetos NameValuePair que definen el IDAudiencia.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método `executeBatch` que llama a `startSession` y `setAudience`.

```
NameValuePair custId = new NameValuePairImpl();
custId.setName("IDCliente");
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);
. . .
/** Crear la matriz de comandos */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Realizar la llamada */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Procesar la respuesta según corresponda */
    processExecuteBatchResponse(batchResponse);
```

setAudienceLevel

El método `setAudienceLevel` define el Nivel de audiencia de los comandos `setAudience` y `startSession`.

`setAudienceLevel(`*`nivelAudiencia`*`)`

- *nivelAudiencia*: cadena que contiene el Nivel de audiencia.

  Importante: El nombre del *nivelAudiencia* debe coincidir con el nombre del nivel de audiencia tal como está definido exactamente en Campaign. Distingue entre mayúsculas y minúsculas.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método `executeBatch` que llama a `startSession` y `setAudience`.

```
String audienceLevel="Cliente";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceLevel(audienceLevel);
. . .
/** Crear la matriz de comandos */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Realizar la llamada */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);
```

```
/** Procesar la respuesta según corresponda */
    processExecuteBatchResponse(batchResponse);
```

setDebug

El método setDebug define el nivel de depuración del comando startSession.

setDebug(*debug*)

Si es true, el servidor de ejecución registra la información de depuración en el registro del servidor de ejecución. Si es false, el servidor de ejecución no registra información de depuración. El indicador de depuración se establece individualmente para cada sesión. Por lo tanto, puede rastrear los datos de depuración para una sesión de ejecución individual.

- **debug**: un valor booleano (true o false).

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método executeBatch que llama a startSession y setDebug.

```
boolean initialDebugFlag=true;
boolean newDebugFlag=false;
. . .
/* crear el comando startSession */
Command startSessionCommand = new CommandImpl();
startSessionCommand.setDebug(initialDebugFlag);
. . .

/* crear el comando setDebug */
Command setDebugCommand = new CommandImpl();
setDebugCommand.setMethodIdentifier(Command.COMMAND_SETDEBUG);
setDebugCommand.setDebug(newDebugFlag);

/** Crear la matriz de comandos */
Command[] commands =
    {
        startSessionCommand,
        setDebugCommand,
    };
/** Realizar la llamada */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Procesar la respuesta según corresponda */
    processExecuteBatchResponse(batchResponse);
```

setEvent

El método setEvent define el nombre del evento utilizado por el comando postEvent.

setEvent(*evento*)

- **evento**: cadena que contiene el nombre del evento.

Importante: El nombre del *evento* debe coincidir con el nombre del evento tal como está definido exactamente en el canal interactivo. Distingue entre mayúsculas y minúsculas.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método executeBatch que llama a postEvent.

```
String eventName = "SearchExecution";

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

setEventParameters

El método setEventParameters define los parámetros de evento utilizados por el comando postEvent. Estos valores se almacenan en los datos de la sesión.

setEventParameters(*parámetrosEvento*)

- **parámetrosEvento**: matriz de objetos NameValuePair que define los parámetros del evento.

Por ejemplo, si el evento está registrando una oferta en el historial de contactos, debe incluir el código de tratamiento de la oferta.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método executeBatch que llama a 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

El método **setGetOfferRequests** establece el parámetro para recuperar ofertas utilizadas por el comando getOffersForMultipleInteractionPoints.

setGetOfferRequests(*númeroSolicitado*)

- **númeroSolicitado**: matriz de objetos de GetOfferRequest que definen el parámetro para recuperar ofertas.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método GetOfferRequest que llama a 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

El método `setInteractiveChannel` define el nombre del canal interactivo utilizado por el comando `startSession`.

`setInteractiveChannel(`*`canalInteractivo`*`)`

- **canalInteractivo**: cadena que contiene el nombre de canal interactivo.

Importante: *canalInteractivo* debe coincidir con el nombre del canal interactivo tal como está definido exactamente en Campaign. Distingue entre mayúsculas y minúsculas.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método `executeBatch` que llama a `startSession`.

```
String interactiveChannel="Sitio web Cuentas";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setInteractiveChannel(interactiveChannel);
```

setInteractionPoint

El método `setInteractionPoint` define el nombre del punto de interacción utilizado por los comandos `getOffers` y `postEvent`.

`setInteractionPoint(`*`puntoInteracción`*`)`

- **puntoInteracción**: cadena que contiene el nombre de punto de interacción.

Importante: *puntoInteracción* debe coincidir con el punto de interacción tal como está definido exactamente en el canal interactivo. Distingue entre mayúsculas y minúsculas.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método `executeBatch` que llama a `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

El método `setMethodIdentifier` define el tipo de comando contenido en el objeto de comando.

`setMethodIdentifier(`*`métodoIdentificador`*`)`

- **métodoIdentificador**: cadena que contiene el tipo de comando.

  Los valores válidos son:
  - **COMMAND_ENDSESSION**: representa el método `endSession`.
  - **COMMAND_GETOFFERS**: representa el método `getOffers`.
  - **COMMAND_GETPROFILE**: representa el método `getProfile`.
  - **COMMAND_GETVERSION**: representa el método `getVersion`.
  - **COMMAND_POSTEVENT**: representa el método `postEvent`.
  - **COMMAND_SETAUDIENCE**: representa el método `setAudience`.
  - **COMMAND_SETDEBUG**: representa el método `setDebug`.
  - **COMMAND_STARTSESSION**: representa el método `startSession`.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método `executeBatch` que llama a `getVersion` y `endSession`.

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

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

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

setNumberRequested

El método `setNumberRequested` define el número de ofertas solicitadas por el comando `getOffers`.

`setNumberRequested(`*`numberRequested`*`)`

- **númeroSolicitado**: un entero que define el número de ofertas solicitadas por el comando `getOffers`.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método `executeBatch` que llama a `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

El método setRelyOnExistingSession define un booleano que define si el comando startSession utiliza una sesión existente o no.

```
setRelyOnExistingSession(relyOnExistingSession)
```

Si es true, el ID de sesión de executeBatch debe coincidir con un ID de sesión existente. Si es false, debe proporcionar un nuevo ID de sesión con el método executeBatch.

- **relyOnExistingSession**: un valor booleano (true o false).

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente es un fragmento de un método executeBatch que llama a startSession.

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

Acerca de la interfaz NameValuePair

Muchos métodos de la API de Interact devuelven objetos NameValuePair o requieren que pase objetos NameValuePair como argumentos. Cuando se pasan como argumentos a un método, debe utilizar la implementación predeterminada NameValuePairImpl.

getName

El método getName devuelve el componente de nombre de un objeto NameValuePair.

```
getName()
```

Valor de retorno

El método getName devuelve una serie.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa el objeto de respuesta para getProfile.

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

getValueAsDate

El método getValueAsDate devuelve el valor de un objeto NameValuePair.

```
getValueAsDate()
```

Debe utilizar getValueDataType antes de utilizar getValueAsDate para confirmar que está haciendo referencia al tipo de datos correcto.

Valor de retorno

El método getValueAsDate devuelve una fecha.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa un NameValuePair e imprime el valor si es una fecha.

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

getValueAsNumeric

El método getValueAsNumeric devuelve el valor de un objeto NameValuePair.

```
getValueAsNumeric()
```

Debe utilizar getValueDataType antes de utilizar getValueAsNumeric para confirmar que está haciendo referencia al tipo de datos correcto.

Valor de retorno

El método getValueAsNumeric devuelve un valor doble. Si, por ejemplo, está recuperando un valor almacenado originalmente en la tabla de perfil como entero, getValueAsNumeric devuelve un valor doble.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa un NameValuePair e imprime el valor si es numérico.

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

getValueAsString

El método getValueAsString devuelve el valor de un objeto NameValuePair.

```
getValueAsString()
```

Debe utilizar getValueDataType antes de utilizar getValueAsString para confirmar que está haciendo referencia al tipo de datos correcto.

Valor de retorno

El método getValueAsString devuelve una serie. Si, por ejemplo, está recuperando un valor almacenado originalmente en la tabla de perfil como char, varchar o char[10], getValueAsString devuelve un serie.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa NameValuePair e imprime el valor si es una serie.

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

getValueDataType

El método `getValueDataType` devuelve el tipo de datos de un objeto NameValuePair.

```
getValueDataType()
```

Debe utilizar `getValueDataType` antes de utilizar `getValueAsDate`, `getValueAsNumeric` o `getValueAsString` para confirmar que está haciendo referencia al tipo de datos correcto.

Valor de retorno

El método `getValueDataType` devuelve una serie que indica si NameValuePair contiene datos, un número o una serie.

Los valores válidos son:

- **DATA_TYPE_DATETIME**: fecha que contiene un valor de fecha y hora.
- **DATA_TYPE_NUMERIC**: valor doble que contiene un valor de número.
- **DATA_TYPE_STRING**: cadena que contiene un valor de texto.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa el objeto de respuesta de un método `getProfile`.

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

setName

El método `setName` define el componente de nombre de un objeto NameValuePair.

```
setName(nombre)
```

- **nombre**: cadena que contiene el componente de nombre de un objeto NameValuePair.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente muestra cómo definir el componente de nombre de un NameValuePair.

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

setValueAsDate

El método `setValueAsDate` define el valor de un objeto NameValuePair.

`setValueAsDate(`*`valorComoFecha`*`)`

- **valorComoFecha**: fecha que contiene el valor de fecha y hora de un objeto NameValuePair.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente muestra cómo definir el componente de valor de un NameValuePair si el valor es una fecha.

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

setValueAsNumeric

El método `setValueAsNumeric` define el valor de un objeto NameValuePair.

`setValueAsNumeric(`*`valorComoNumérico`*`)`

- **valorComoNumérico**: valor doble que contiene el valor numérico de un objeto NameValuePair.

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente muestra cómo definir el componente de valor de un NameValuePair si el valor es numérico.

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

setValueAsString

El método `setValueAsString` define el valor de un objeto NameValuePair.

`setValueAsString(`*`valorComoSerie`*`)`

- **valorComoSerie**: serie que contiene el valor de un objeto NameValuePair

Valor de retorno

Ninguno.

Ejemplo

El ejemplo siguiente muestra cómo definir el componente de valor de un NameValuePair si el valor es numérico.

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

setValueDataType

El método `setValueDataType` define el tipo de datos de un objeto NameValuePair.

`getValueDataType(`*`valorTipoDatos`*`)`

Los valores válidos son:

- **DATA_TYPE_DATETIME**: fecha que contiene un valor de fecha y hora.
- **DATA_TYPE_NUMERIC**: valor doble que contiene un valor de número.
- **DATA_TYPE_STRING**: cadena que contiene un valor de texto.

Valor de retorno

Ninguno.

Ejemplo

Los ejemplos siguientes muestran cómo establecer el tipo de datos del valor de un NameValuePair.

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

NameValuePair parm3 = new NameValuePairImpl();
parm3.setName("Navegador");
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);
```

Acerca de la clase Offer

La clase `Offer` contiene métodos que definen un objeto de oferta. Este objeto de oferta contiene muchas de las mismas propiedades que una oferta en Campaign.

El objeto de oferta contiene los siguientes atributos:

- **AdditionalAttributes**: NameValuePairs que contienen los atributos de oferta personalizados que ha definido en Campaign.
- **Description**: la descripción de la oferta.
- **EffectiveDate**: la fecha efectiva de la oferta.
- **ExpirationDate**: la fecha de caducidad de la oferta.

- **OfferCode**: el código de oferta de la oferta.
- **OfferName**: el nombre de la oferta.
- **TreatmentCode**: el código de tratamiento de la oferta.
- **Score**: la puntuación de marketing de la oferta, o la puntuación definida por `ScoreOverrideTable` si la propiedad `enableScoreOverrideLookup` es true.

getAdditionalAttributes

El método `getAdditionalAttributes` devuelve los atributos de oferta personalizados definidos en Campaign.

```
getAdditionalAttributes()
```

Valor de retorno

El método `getAdditionalAttributes` devuelve una matriz de objetos NameValuePair.

Ejemplo

El ejemplo siguiente ordena todos los atributos adicionales, comprobando la fecha efectiva y la fecha de caducidad, e imprimiendo los demás parámetros.

```
for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
{
    // comprobar si existe la fecha efectiva
    if(offerAttribute.getName().equalsIgnoreCase("fechaEfectiva"))
    {
        System.out.println("Se ha encontrado la fecha efectiva");
    }
    // comprobar si existe la fecha de caducidad
    else if(offerAttribute.getName().equalsIgnoreCase("fechaCaducidad"))
    {
        System.out.println("Se ha encontrado la fecha de caducidad");
    }
    printNameValuePair(offerAttribute);
    }
}
public static void printNameValuePair(NameValuePair nvp)
{
    // imprimir el nombre:
    System.out.println("Nombre:"+nvp.getName());

    // en función del tipo de datos, llame al método adecuado para obtener el valor
    if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_DATETIME)
        System.out.println("Valor de fecha:"+nvp.getValueAsDate());
    else if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_NUMERIC)
        System.out.println("Valor numérico:"+nvp.getValueAsNumeric());
    else
        System.out.println("Valor de serie:"+nvp.getValueAsString());
}
```

getDescription

El método `getDescription` devuelve la descripción de la oferta definida en Campaign.

```
getDescription()
```

Valor de retorno

El método `getDescription` devuelve una serie.

Ejemplo

El ejemplo siguiente imprime la descripción de una oferta.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // imprimir oferta
     System.out.println("Descripción de oferta:"+offer.getDescription());
}
```

getOfferCode

El método `getOfferCode` devuelve el código de oferta de la oferta tal como está definido en Campaign.

```
getOfferCode()
```

Valor de retorno

El método `getOfferCode` devuelve una matriz de series que contiene el código de oferta de la oferta.

Ejemplo

El ejemplo siguiente imprime el código de oferta de una oferta.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // imprimir oferta
   System.out.println("Código de oferta:"+offer.getOfferCode());
}
```

getOfferName

El método `getOfferName` devuelve el nombre de la oferta tal como está definido en Campaign.

```
getOfferName()
```

Valor de retorno

El método `getOfferName` devuelve una serie.

Ejemplo

El ejemplo siguiente imprime el nombre de una oferta.

```
for(Offer offer : offerList.getRecommendedOffers())
{
// imprimir oferta
System.out.println("Nombre de oferta:"+offer.getOfferName());
}
```

getScore

El método `getScore` devuelve una puntuación, que se basa en las ofertas que ha configurado.

```
getScore()
```

El método `getScore` devuelve uno de los elementos siguientes:

- Si no ha habilitado la tabla de ofertas predeterminadas, la tabla de sustituciones de puntuación o el aprendizaje incorporado, este método devuelve la puntuación de marketing de la oferta tal como está definida en la pestaña de estrategia de interacción.
- Si ha habilitado las ofertas predeterminadas o la tabla de sustituciones de puntuación y no ha habilitado el aprendizaje incorporado, este método devuelve la puntuación de la oferta tal como está definida por el orden de prioridad entre la tabla de ofertas predeterminadas, la puntuación del usuario de marketing y la tabla de sustituciones de puntuación.
- Si ha habilitado el aprendizaje incorporado, este método devuelve la puntuación final que ha utilizado el aprendizaje incorporado para ordenar las ofertas.

Valor de retorno

El método `getScore` devuelve un entero que representa la puntuación de la oferta.

Ejemplo

El ejemplo siguiente imprime la puntuación de una oferta.

```
for(Offer offer : offerList.getRecommendedOffers())
{
// imprimir oferta
System.out.println("Puntuación de oferta:"+offer.getOfferScore());
}
```

getTreatmentCode

El método `getTreatmentCode` devuelve el código de tratamiento de la oferta tal como está definido en Campaign.

```
getTreatmentCode()
```

Dado que Campaign utiliza el código de tratamiento para identificar la instancia de la oferta ofrecida, este código se debe devolver como parámetro de evento cuando se utilice el método `postEvent` para registrar un evento de contacto, aceptación o rechazo de la oferta. Si está registrando la aceptación o el rechazo de una oferta, debe establecer el valor de nombre de NameValuePair que representa el código de tratamiento en `UACIOfferTrackingCode`.

Valor de retorno

El método `getTreatmentCode` devuelve una serie.

Ejemplo

El ejemplo siguiente imprime el código de tratamiento de una oferta.

```
for(Offer offer : offerList.getRecommendedOffers())
{
    // imprimir oferta
    System.out.println("Código de tratamiento de oferta:"+offer.getTreatmentCode());
}
```

Acerca de la clase OfferList

La clase `OfferList` contiene métodos que definen los resultados del método `getOffers`.

El objeto OfferList contiene los siguientes atributos:

- **DefaultString**: la cadena predeterminada definida para el punto de interacción en el canal interactivo.
- **RecommendedOffers**: una matriz de los objetos de oferta solicitados por el método `getOffers`.

La clase `OfferList` trabaja con listas de ofertas. Esta clase no está relacionada con las listas de ofertas de Campaign.

getDefaultString

El método `getDefaultString` devuelve la serie predeterminada para el punto de interacción tal como está definido en Campaign.

```
getDefaultString()
```

Si el objeto RecommendedOffers está vacío, debe configurar el punto de encuentro para presentar esta serie para asegurarse de que el contenido se presenta. Interact completa el objeto DefaultString solo si el objeto RecommendedOffers está vacío.

Valor de retorno

El método `getDefaultString` devuelve una serie.

Ejemplo

El ejemplo siguiente obtiene la serie predeterminada si el objeto offerList no contiene ofertas.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
  System.out.println("Nombre de oferta:"+offer.getOfferName());
    }
}
else // basarse en la serie de oferta predeterminada
    System.out.println("Oferta predeterminada:"+offerList.getDefaultString());
```

getRecommendedOffers

El método `getRecommendedOffers` devuelve una matriz de objetos Offer solicitados por el método `getOffers`.

```
getRecommendedOffers()
```

Si la respuesta a `getRecommendedOffer` está vacía, el punto de encuentro debe presentar el resultado de `getDefaultString`.

Valor de retorno

El método `getRecommendedOffers` devuelve un objeto Offer.

Ejemplo

El ejemplo siguiente procesa el objeto OfferList e imprime el nombre de oferta para todas las ofertas recomendadas.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
```

```
    {
        // imprimir oferta
        System.out.println("Nombre de oferta:"+offer.getOfferName());
    }
}
else // basarse en la serie de oferta predeterminada
System.out.println("Oferta predeterminada:"+offerList.getDefaultString());
```

Acerca de la clase Response

La clase `Response` contiene métodos que definen los resultados de cualquiera de los métodos de la clase InteractAPI.

El objeto Response contiene los atributos siguientes:

- **AdvisoryMessages**: matriz de mensajes de recomendación. Este atributo se completa solo si hay avisos o errores al ejecutar el método.
- **ApiVersion**: cadena que contiene la versión de la API. Este atributo lo completa el método `getVersion`.
- **OfferList**: objeto OfferList que contiene las ofertas que solicita el método `getOffers`.
- **ProfileRecord**: matriz de NameValuePairs que contiene datos de perfil. Este atributo lo completa el método `getProfile`.
- **SessionID**: cadena que define el ID de sesión. Lo devuelven todos los métodos de clase `InteractAPI`.
- **StatusCode**: número que indica si el método se ha ejecutado sin error, con un aviso o con errores. Lo devuelven todos los métodos de clase `InteractAPI`.

getAdvisoryMessages

El método `getAdvisoryMessages` devuelve una matriz de Mensajes de recomendación del objeto Response.

```
getAdvisoryMessages()
```

Valor de retorno

El método `getAdvisoryMessages` devuelve una matriz de objetos Advisory Message.

Ejemplo

El ejemplo siguiente obtiene los objetos AdvisoryMessage de un objeto Response e itera a través de ellos, imprimiendo los mensajes.

```
AdvisoryMessage[] messages = response.getAdvisoryMessages();
    for(AdvisoryMessage msg : messages)
    {
        System.out.println(msg.getMessage());
        // Algunos mensajes de aviso pueden tener detalles adicionales:
        System.out.println(msg.getDetailMessage());
    }
```

getApiVersion

El método `getApiVersion` devuelve la versión de la API de un objeto Response.

```
getApiVersion()
```

El método `getVersion` completa el atributo ApiVersion de un objeto Response.

Valor de retorno

EL objeto Response devuelve una serie.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa el objeto de respuesta de `getVersion`.

```
if(response.getStatusCode() == Response.STATUS_SUCCESS)
{
    System.out.println("llamada a getVersion procesada sin avisos o errores");
    System.out.println("Versión de la API:" + response.getApiVersion());
}
```

getOfferList

El método `getOfferList` devuelve el objeto OfferList de un objeto Response.

```
getOfferList()
```

El método `getOffers` completa el objeto OfferList de un objeto Response.

Valor de retorno

El objeto Response devuelve un objeto OfferList.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa el objeto de respuesta para `getOffers`.

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

getAllOfferLists

El método `getAllOfferLists` devuelve una matriz de todas las OfferLists de un objeto Response.

```
getAllOfferLists()
```

Lo utiliza el método `getOffersForMultipleInteractionPoints` que completa el objeto de matriz OfferList de un objeto Response.

Valor de retorno

El objeto Response devuelve una matriz OfferList.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa el objeto de respuesta para `getOffers`.

```
OfferList[] allOfferLists = response.getAllOfferLists();
  if (allOfferLists != null) {
    for (OfferList ol : allOfferLists) {
      System.out.println("Las ofertas siguientes se entregan
                          para el punto de interacción"
          + ol.getInteractionPointName() + ":");
      for (Offer o : ol.getRecommendedOffers()) {
        System.out.println(o.getOfferName());
   }
  }
 }
```

getProfileRecord

El método `getProfileRecord` devuelve los registros de perfil para la sesión actual como una matriz de objetos NameValuePair. Estos registros de perfil incluyen también los `parámetrosEvento` añadidos anteriormente en la sesión de ejecución.

```
getProfileRecord()
```

El método `getProfile` completa los objetos NameValuePair del registro del perfil de un objeto Response.

Valor de retorno

El objeto Response devuelve una matriz de objetos NameValuePair.

Ejemplo

El ejemplo siguiente es un fragmento de un método que procesa el objeto de respuesta para `getOffers`.

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

getSessionID

El método `getSessionID` devuelve el ID de sesión.

```
getSessionID()
```

Valor de retorno

El método `getSessionID` devuelve una serie.

Ejemplo

El ejemplo siguiente muestra un mensaje que se puede visualizar al final o al inicio del error para indicar a qué sesión corresponde cualquier error.

```
System.out.println("Esta respuesta corresponde a sessionId:"+response.getSessionID());
```

getStatusCode

El método `getStatusCode` devuelve el código de estado de un objeto Response.

```
getStatusCode()
```

Valor de retorno

El objeto Response devuelve un entero.

- 0 - STATUS_SUCCESS: el método al que se ha llamado se ha completado sin errores. Puede haber Mensajes de recomendación o no.
- 1 - STATUS_WARNING: el método al que se ha llamado se ha completado con un mensaje de aviso como mínimo (pero sin errores). Consulte Mensajes de recomendación para ver más detalles.
- 2 - STATUS_ERROR: el método al que se ha llamado no se ha completado satisfactoriamente y tiene como mínimo un mensaje de error. Consulte Mensajes de recomendación para ver más detalles.

Ejemplo

A continuación se muestra un ejemplo de cómo puede utilizar `getStatusCode` en el manejo de errores.

```
public static void processSetDebugResponse(Response response)
{
    // comprobar si la respuesta es satisfactoria o no
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("llamada a setDebug procesada sin avisos o errores");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("llamada a setDebug procesada con un aviso");
    }
    else
    {
        System.out.println("llamada a setDebug procesada con un error");
    }

    // Para los errores, deben aparecer mensajes de aviso explicando por qué
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setDebug",
       response.getAdvisoryMessages());}
```

Capítulo 8. Clases y métodos para la API de JavaScript IBM Interact

Las secciones siguientes listan los requisitos y otros detalles que debería saber antes de trabajar con la API de JavaScript Interact.

La API Interact soporta un tipo de javascript para permitir la comunicación del cliente de usuario final (navegador) con el servidor.

Nota: Esta sección da por supuesto que está familiarizado con una API basada en JavaScript.

Nota: No se permite tener varias apariciones de un parámetro en una única llamada a la API.

Requisitos previos de JavaScript

Antes de utilizar la API JavaScript de Interact en un sitio web, deberá incluir el archivo `interactapi.js` en las páginas web.

Cómo trabajar con datos de sesión

Cuando inicia una sesión con el método `startSession`, los datos de sesión se cargan en la memoria. Durante la sesión, puede leer y escribir en los datos de sesión (que son un superconjunto de los datos de perfil estáticos).

La sesión contiene los siguientes datos:

- Datos de perfil estáticos
- Asignaciones de segmento
- Datos en tiempo real
- Recomendaciones de ofertas

Todos los datos de sesión están disponibles hasta que invoca el método `endSession` o transcurre el tiempo de `sessionTimeout`. Una vez finalizada la sesión, todos los datos que no se han guardado de forma explícita en el historial de contactos o respuestas o en otra tabla de base de datos se pierden.

Los datos se almacenan como un conjunto de pares nombre-valor. Si los datos se leen de una tabla de base de datos, el nombre es la columna de la tabla.

Puede crear estos pares nombre-valor cuando trabaje con la API de Interact. No es necesario declarar todos los pares nombre-valor en un área global. Si establece nuevos parámetros de eventos como pares nombre-valor, el entorno de ejecución añade los pares nombre-valor a los datos de sesión. Por ejemplo, si utiliza los parámetros de evento con el método `postEvent`, el entorno de ejecución añade los parámetros de evento a los datos de sesión, aunque los parámetros de evento no estén disponibles en los datos del perfil. Estos datos sólo existen en los datos de sesión.

Puede sobrescribir datos de sesión en cualquier momento. Por ejemplo, si parte del perfil de cliente incluye `creditScore`, puede pasar en un parámetro de evento

utilizando el tipo personalizado `NameValuePair`. En la clase `NameValuePair`, puede utilizar los métodos `setName` y `setValueAsNumeric` para cambiar el valor. El nombre debe coincidir. En los datos de sesión, el nombre no es sensible a las mayúsculas y minúsculas. Por lo tanto, los nombres `creditscore` o `CrEdItScOrE` deben sobrescribir `creditScore`.

Sólo se mantienen los últimos datos escritos en los datos de sesión. Por ejemplo, `startSession` carga los datos del perfil del valor de `lastOffer`. Un método `postEvent` sobrescribe `lastOffer`. A continuación, un segundo método `postEvent` sobrescribe `lastOffer`. El entorno de ejecución sólo conserva los datos escritos por el segundo método `postEvent` en los datos de sesión.

Cuando la sesión finaliza, los datos se pierden, a menos que haya creado consideraciones especiales como, por ejemplo, la utilización de un proceso Instantánea en el diagrama de flujo interactivo para escribir los datos en una tabla de base de datos. Si tiene previsto utilizar procesos Instantánea, recuerde que los nombres deben coincidir con las limitaciones de la base de datos. Por ejemplo, si sólo tiene permiso para 256 caracteres en el nombre de una columna, el nombre del par nombre-valor no debe exceder los 256 caracteres.

Uso del parámetro de devolución de llamada

La función de devolución de llamada es un parámetro adicional de cada método de la API de JavaScript Interact.

El proceso principal del navegador es un bucle de evento con una sola hebra. La ejecución de una operación de larga ejecución dentro de un bucle de evento con una sola hebra bloquea el proceso. Esto impide que el proceso procese otros eventos mientras espera a que se complete la operación. Para poder evitar bloqueos en operaciones de larga ejecución, XMLHttpRequest proporciona una interfaz asíncrona. Le pasa una devolución de llamada para ejecutar una vez completada la operación, y mientras realiza el proceso, devuelve el control al bucle principal del evento, en lugar de bloquear el proceso.

Si el método se ha realizado correctamente, la función de devolución de llamada llama a `onSuccess`. Si el método ha fallado, la función de devolución de llamada llama a `onError`.

Por ejemplo, si deseaba mostrar ofertas en la página web, deberá utilizar el método `getOffers` y utilizar la devolución de llamada para mostrar en la página. La página web se comporta normalmente y no espera a que Interact devuelva las ofertas. En lugar de esto, cuando Interact devuelve las ofertas, la respuesta se vuelve a enviar en el parámetro de devolución de llamada. Puede analizar los datos de devolución de llamada y mostrar ofertas en la página.

Puede utilizar una devolución de llamada genérica para todas las funciones o también puede utilizar devoluciones de llamada específicas para funciones específicas.

Puede utilizar `var callback = InteractAPI.Callback.create(onSuccess, onError);` para crear una función de devolución de llamada genérica.

Puede utilizar la función siguiente para crear una función de devolución de llamada específica para el método getOffers.

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

Acerca de la clase InteractAPI

La clase InteractAPI contiene los métodos que se utilizan para integrar el punto de encuentro con el servidor de ejecución. Las demás clases y métodos de la API de Interact dan soporte a los métodos de esta clase.

Debe compilar su implementación en `interact_client.jar`, que se encuentra en el directorio `lib` de la instalación del entorno de ejecución de Interact.

startSession

El método `startSession` crea y define una sesión de ejecución.

```
function callStartSession(commandsToExecute, callback) {

        //leer sesión de inicio configurada
        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` puede desencadenar hasta cinco acciones:

- crear una sesión de ejecución.
- cargar los datos de perfil del visitante para el nivel de audiencia actual en la sesión de ejecución, incluidas las tablas de dimensiones marcadas para la carga en la correlación de tablas definida para el canal interactivo.
- desencadenar segmentación, ejecutando todos los diagramas de flujo interactivos para el nivel de audiencia actual.
- cargar datos de supresión de oferta en la sesión, si la propiedad `enableOfferSuppressionLookup` está establecida en true.
- cargar los datos de anulación de puntuaciones en la sesión, si la propiedad `enableScoreOverrideLookup` está establecida en true.

El método `startSession` requiere los parámetros siguientes:

- **ID_sesión**: cadena que identifica el ID de sesión. Debe definir el ID de sesión. Por ejemplo, podría utilizar una combinación de ID de cliente y marca de hora.

  Para definir qué compone una sesión de ejecución, se debe especificar el ID de sesión. Este valor lo gestiona el cliente. El cliente debe sincronizar todas las llamadas a método para el mismo ID de sesión. El comportamiento para las llamadas a la API simultáneas con el mismo ID de sesión no está definido.
- **relyOnExistingSession**: booleano que define si esta sesión utiliza una sesión nueva o existente. Los valores válidos son `true` o `false`. Si es `true`, debe proporcionar un ID de sesión existente con el método `startSession`. Si es `false`, debe proporcionar un nuevo ID de sesión.

  Si se establece `relyOnExistingSession` en `true` y ya existe una sesión, el entorno de ejecución utiliza los datos de sesión existentes y no vuelve a cargar datos o desencadena segmentación. Si la sesión no existe, el entorno de ejecución crea una nueva sesión, lo que incluye cargar datos y desencadenar segmentación.

Establecer relyOnExistingSession en true y utilizarlo con todas las llamadas a startSession resulta útil si el punto de encuentro tiene una duración de sesión superior a la sesión de ejecución. Por ejemplo, una sesión de sitio web está activa durante 2 horas, pero la sesión de ejecución solo está activa 20 minutos.

Si llama a startSession dos veces con el mismo ID de sesión, todos los datos de sesión de la primera llamada a startSession se perderán si relyOnExistingSession es false.

- **debug**: booleano que habilita o inhabilita la información de depuración. Los valores válidos son true o false. Si es true, Interact registra la información de depuración en los registros del servidor de ejecución. El indicador de depuración se establece individualmente para cada sesión. Por lo tanto, puede rastrear los datos de depuración para una sesión individual.
- **canalInteractivo**: cadena que define el nombre del canal interactivo al que hace referencia esta sesión. Este nombre debe coincidir con el nombre del canal interactivo tal como está definido exactamente en Campaign.
- **IDAudiencia**: matriz de objetos NameValuePairImpl donde los nombres deben coincidir con los nombres de columna física de cualquier tabla que contenga el ID de audiencia.
- **nivelAudiencia**: cadena que define el nivel de audiencia.
- **parámetros**: objetos NameValuePairImpl que identifican los parámetros que es necesario pasar con startSession. Estos valores se almacenan en los datos de sesión y pueden utilizarse para la segmentación.

  Si tiene varios diagramas de flujo interactivos para el mismo nivel de audiencia, debe incluir un superconjunto de todas las columnas de todas las tablas. Si configura el tiempo de ejecución para cargar la tabla de perfil, y la tabla de perfil contiene todas las columnas que necesita, no es necesario pasar ningún parámetro, a menos que desee sobrescribir los datos de la tabla de perfil. Si la tabla de perfil contiene un subconjunto de las columnas necesarias, debe incluir las columnas que faltan como parámetros.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a onSuccess. Si el método ha fallado, la función de devolución de llamada llama a onError.

Si IDAudiencia o nivelAudiencia no son válidos y relyOnExistingSession es false, la llamada a startSession fallará. Si interactiveChannel no es válido, startSession fallará, independientemente de si relyOnExistingSession es true o false.

Si relyOnExistingSession es true y realiza una segunda llamada a startSession utilizando el mismo ID_sesión, pero la primera sesión ha caducado, Interact creará una nueva sesión.

Si relyOnExistingSession es true y realiza una segunda llamada a startSession utilizando el mismo ID_sesión pero un IDAudiencia o nivelAudiencia distinto, el servidor de ejecución cambiará la audiencia para la sesión existente.

Si relyOnExistingSession es true y realiza una segunda llamada a startSession utilizando el mismo ID_sesión pero un canalInteractivo distinto, el servidor de ejecución creará una nueva sesión.

Valor de retorno

El servidor de ejecución responde a startSession con un objeto Response que tiene los siguientes atributos completados:

- AdvisoryMessages (si StatusCode no es igual a 0)
- ApiVersion
- SessionID
- StatusCode

Desduplicación de la oferta para varios atributos de oferta

Mediante la API (interfaz de programación de aplicaciones) de Interact, dos llamadas de API proporcionan las ofertas `getOffers` y `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` puede impedir la devolución de ofertas duplicadas a nivel de *OfferID*, pero no puede desduplicar ofertas para varias categorías de oferta. Así, por ejemplo, para que Interact devuelva una única oferta de cada categoría de oferta, anteriormente era necesaria una solución temporal. Con la adición de dos parámetros a la llamada de API `startSession`, ahora es posible la desduplicación de ofertas para varios atributos de oferta como, por ejemplo, la categoría.

Esta lista resume los parámetros que se han añadido a la llamada de API `startSession`. Para obtener más información sobre estos parámetros u otros aspectos de la API de Interact, consulte la *Guía del administrador de IBM Interact* o los archivos Javadoc que se incluyen con la instalación de Interact en `<Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Para crear una llamada de API `startSession` con desduplicación de oferta para que las llamadas `getOffer` subsiguientes devuelvan una única oferta de cada categoría, debe incluir el parámetro `UACIOfferDedupeAttribute` como parte de la llamada de API. Puede especificar un parámetro con el formato `nombre,valor,tipo`, como se muestra a continuación:

  `UACIOfferDedupeAttribute,<nombre_atributo>,string`

  En este ejemplo, sustituiría *<nombre_atributo>* por el nombre del atributo de oferta que desee utilizar como criterio para la desduplicación, por ejemplo, Categoría.

  Nota: Interact examina las ofertas que tienen el mismo valor de atributo que especifique (por ejemplo, Categoría) y elimina las ofertas duplicadas excepto la oferta que tenga la mayor puntuación. Si las ofertas que tienen el atributo duplicado también tienen puntuaciones idénticas, Interact devuelve una selección aleatoria entre las ofertas coincidentes.

- `UACINoAttributeDedupeIfFewerOffer`. Cuando incluye el parámetro `UACIOfferDedupeAttribute` en la llamada a `startSession`, también puede establecer el parámetro `UACINoAttributeDedupeIfFewerOffer` para especificar el comportamiento cuando la lista de ofertas después de la desduplicación ya no contiene ofertas suficientes para satisfacer la solicitud original.

  Por ejemplo, si define `UACIOfferDedupeAttribute` para utilizar la categoría de oferta para desduplicar ofertas y la llamada a getOffers posterior solicita que se devuelvan ocho ofertas, la desduplicación puede producir menos de ocho ofertas elegibles. En este caso, el establecimiento el parámetro `UACINoAttributeDedupeIfFewerOffer` en true dará como resultado la adición de algunos de los duplicados a la lista de ofertas elegibles para satisfacer el número solicitado de ofertas. En este ejemplo, si establece el parámetro en false, el número de ofertas que se devuelven será menor que el número solicitado.

UACINoAttributeDedupeIfFewerOffer se establece en true de forma predeterminada.

Por ejemplo, supongamos que ha especificado como parámetro startSession que el criterio de desduplicación sea la oferta Categoría, como se muestra a continuación:

```
UACIOfferDedupeAttribute,Category,string;

UACINoAttributeDedupeIfFewerOffer,1,string
```

De forma predeterminada, UACIOfferDedupeAttribute no eliminará duplicados de ofertas si se devuelve un número inferior a la cantidad solicitada. Sin embargo, para garantizar que se produce la eliminación de duplicados cuando se devuelven menos ofertas de las solicitadas, se debe proporcionar el parámetro UACINoAttributeDedupeIfFewerOffer y se debe establecer en 1.

Utilizados conjuntamente, estos parámetros hacen que Interact desduplique las ofertas basándose en el atributo de oferta "Categoría" y que se devuelvan solamente las ofertas desduplicadas, aunque el número resultante de ofertas sea menor que el solicitado (UACINoAttributeDedupeIfFewerOffer es false).

Cuando emite una llamada de API getOffers, el conjunto original de ofertas elegibles puede incluir estas ofertas:

- Category=Electronics: oferta A1 con una puntuación de 100 y oferta A2 con una puntuación de 50.
- Category=Smartphones: oferta B1 con una puntuación de 100, oferta B2 con una puntuación de 80 y oferta B3 con una puntuación de 50.
- Category=MP3Players: oferta C1 con una puntuación de 100 y oferta C2 con una puntuación de 50.

En este caso, había dos ofertas duplicadas que coinciden con la primera categoría, tres ofertas duplicadas que coinciden con la segunda categoría y dos ofertas duplicadas que coinciden con la tercera categoría. Las ofertas que se devuelven son las ofertas con mayor puntuación de cada categoría, es decir, las ofertas A1, B1 y C1.

Si la llamada de API getOffers solicita seis ofertas, este ejemplo establece UACINoAttributeDedupeIfFewerOffer en false para devolver solamente tres ofertas.

Si la llamada de API getOffers solicita seis ofertas y este ejemplo omite el parámetro UACINoAttributeDedupeIfFewerOffer o lo establece explícitamente en true, algunas de las ofertas duplicadas se incluirán en el resultado para satisfacer el número solicitado.

postEvent

El método postEvent permite ejecutar cualquier evento definido en el canal interactivo.

```
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**: una cadena que identifica el ID de sesión.
- **eventName**: una cadena que identifica el nombre del evento.

  Nota: El nombre del evento debe coincidir con el nombre del evento tal como está definido en el canal interactivo. Este nombre es sensible a mayúsculas y minúsculas.
- **eventParameters**. Los objetos `NameValuePairImpl` identifican los parámetros que deben pasarse con el evento. Estos valores se almacenan en los datos de sesión.

  Si este evento desencadena la resegmentación, debe asegurarse de que todos los datos necesarios para los diagramas de flujo interactivos estén disponibles en los datos de sesión. Si alguno de estos valores no se ha completado mediante las acciones anteriores (por ejemplo, a partir de `startSession` o `setAudience`, o al cargar la tabla de perfil), debe incluir un eventParameter para cada valor que falte. Por ejemplo, si ha configurado todas las tablas de perfiles para que se carguen en la memoria, debe incluir un NameValuePair para los datos temporales necesarios para los diagramas de flujo interactivos.

  Si utiliza más de un nivel de audiencia, lo más probable es que tenga distintos conjuntos de eventParameters para cada nivel de audiencia. Debe incluir alguna lógica para asegurarse de que está seleccionando el conjunto de parámetros correcto para el nivel de audiencia.

  Importante: Si este evento se registra en un historial de respuestas, debe pasar el código de tratamiento para la oferta. Debe definir el nombre de NameValuePair como `"UACIOfferTrackingCode"`.

  Sólo puede pasar un código de tratamiento por evento. Si no pasa el código de tratamiento para un contacto de oferta, Interact registra un contacto de oferta para cada oferta en la última lista de ofertas recomendada. Si no pasa el código de tratamiento para una respuesta, Interact devuelve un error.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a `onSuccess`. Si el método ha fallado, la función de devolución de llamada llama a `onError`.
- Existen otros parámetros reservados que se utilizan con `postEvent` y otros métodos, que se describen más adelante en esta sección.

Toda solicitud de resegmentación o escritura en el historial de contactos o de respuestas no espera una respuesta.

La resegmentación no borra los resultados de una segmentación anterior para el nivel de audiencia actual. Puede utilizar el parámetro `UACIExecuteFlowchartByName` para definir diagramas de flujo específicos para ejecutar. El método `getOffers` espera a que la resegmentación finalice antes de ejecutarse. Por lo tanto, si llama a un método `postEvent`, que desencadena una resegmentación inmediatamente antes de una llamada a `getOffers`, podría haber una retardo.

Valor de retorno

El servidor de ejecución responde a `postEvent` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion

- OfferList
- Profile
- SessionID
- StatusCode

obtenerOfertas

El método `getOffers` permite solicitar ofertas del servidor de ejecución.

```
function callGetOffers(commandsToExecute, callback) {

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

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

}
```

- **session ID**-una serie que identifica la sesión actual.
- **Interaction point**-una serie que identifica el nombre del punto de interacción al que hace referencia este método.

  Nota: Este nombre debe coincidir exactamente con el nombre del punto de interacción definido en el canal interactivo.
- **nofRequested**-un entero que identifica el número de ofertas solicitadas.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a `onSuccess`. Si el método ha fallado, la función de devolución de llamada llama a `onError`.

El método `getOffers` espera el número de milisegundos definido en la propiedad `segmentationMaxWaitTimeInMS` a que finalice toda la resegmentación antes de ejecutarse. Por lo tanto, si invoca un método `postEvent` que desencadena una resegmentación o un método `setAudience` inmediatamente antes de una llamada `getOffers`, puede producirse un retardo.

Valor de retorno

El servidor de ejecución responde a `getOffers` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getOffersForMultipleInteractionPoints

El método `getOffersForMultipleInteractionPoints` permite solicitar ofertas del servidor de ejecución para varios IP con desduplicación.

```
function callGetOffersForMultipleInteractionPoints(commandsToExecute, callback) {

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

        //recortar serie
```

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

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

        //crear solicitudes obtener oferta
        var getOffReqs = [];
        for(var i = 0; i < parts.length; i += 1) {
            var getofReqObj = parseGetOfferReq(parts[i]);
            if (getofReqObj) {
                getOffReqs.push(getofReqObj);

        InteractAPI.getOffersForMultipleInteractionPoints
        (ssId, getOffReqs, callback);

}
```

- **session ID**-una serie que identifica la sesión actual.
- **requestDetailsStr** - una serie que proporciona una matriz de objetos `GetOfferRequest`.

  Cada objeto `GetOfferRequest` especifica:

  – **nombreIP**: el nombre del punto de interacción (IP) para el que el objeto solicita ofertas

  – **númeroSolicitado**: el número de ofertas exclusivas necesarias para el IP especificado

  – **atributosOferta**: requisitos de los atributos de las ofertas entregadas utilizando una instancia de `OfferAttributeRequirements`

  – **políticaDuplicación**: ID de política de duplicación de las ofertas que se entregarán

    Las políticas de duplicación determinan si se devolverán ofertas duplicadas en los distintos puntos de interacción en una única llamada de método. (*Dentro* de un punto de interacción individual, nunca se devuelven ofertas duplicadas). Actualmente, se da soporte a dos políticas de duplicación.

    - NO_DUPLICATION (Valor de ID = 1). Ninguna de las ofertas que se han incluido en las instancias de `GetOfferRequest` anteriores se incluirán en esta instancia de `GetOfferRequest` (es decir, Interact aplicará la desduplicación).
    - ALLOW_DUPLICATION (Valor de ID = 2). Se incluirán todas las ofertas que cumplan los requisitos especificados en esta instancia de `GetOfferRequest`. Las ofertas que se han incluido en las instancias de `GetOfferRequest` anteriores no se reconciliarán.

  – **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a `onSuccess`. Si el método ha fallado, la función de devolución de llamada llama a `onError`.

  El orden de las solicitudes en el parámetro de matriz es también el orden de prioridad cuando se entregan las ofertas.

  Por ejemplo, supongamos que los IP en la solicitud son IP1 e IP2, que no se permiten ofertas duplicadas (un ID de política de duplicación = 1) y que cada uno solicita cada dos ofertas. Si Interact encuentra las ofertas A, B y C para IP1 y las ofertas A y D para IP2, la respuesta contendrá las ofertas A y B para IP1, y sólo la oferta D para IP2.

  Asimismo, tenga en cuenta que cuando el ID de política de duplicación es 1, las ofertas que se hayan entregado con un IP con una prioridad mayor no se entregará con este IP.

El método getOffersForMultipleInteractionPoints espera el número de milisegundos definido en la propiedad segmentationMaxWaitTimeInMS a que finalice toda la resegmentación antes de ejecutarse. Por lo tanto, si invoca un método postEvent que desencadena una resegmentación o un método setAudience inmediatamente antes de una llamada getOffers, puede producirse un retardo.

Valor de retorno

El servidor de ejecución responde a getOffersForMultipleInteractionPoints con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- Array of OfferList
- Profile
- SessionID
- StatusCode

setAudience

El método setAudience permite establecer el ID y el nivel de audiencia para un 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);

}
```

- **ID_sesión**: una cadena que identifica el ID de sesión.
- **IDAudiencia**: matriz de objetos NameValuePairImpl que define el ID de audiencia.
- **nivelAudiencia**: cadena que define el nivel de audiencia.
- **parámetros**: objetos NameValuePairImpl que identifican los parámetros que se deben pasar con setAudience. Estos valores se almacenan en los datos de sesión y pueden utilizarse para la segmentación.

  Debe tener un valor para cada columna de su perfil. Este es un superconjunto de todas las columnas de todas las tablas definidas para el canal interactivo y los datos en tiempo real. Si ya ha completado todos los datos de sesión con startSession o postEvent, no es necesario enviar nuevos parámetros.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a onSuccess. Si el método ha fallado, la función de devolución de llamada llama a onError.

El método setAudience desencadena una resegmentación. El método getOffers espera a que la resegmentación finalice antes de ejecutarse. Por lo tanto, si invoca un método setAudience inmediatamente antes de una llamada getOffers, puede producirse un retardo.

El método setAudience también carga los datos del perfil para el ID de audiencia. Puede utilizar el método setAudience para forzar que se vuelvan a cargar los mismos datos del perfil cargados por el método startSession.

El método setAudience vuelve a cargar la tabla de lista blanca y de lista negra en la sesión existente. Puede utilizar el método setAudience con los parámetros UACIPurgePriorWhiteListOnLoad y UACIPurgePriorBlackListOnLoad para volver a cargar la tabla de lista blanca y de lista negra en una sesión existente.

De forma predeterminada, cuando se llama al método setAudience, se elimina todo el contenido de la lista negra. Puede establecer los parámetros UACIPurgePriorWhiteListOnLoad y UACIPurgePriorBlackListOnLoad en la llamada setAudience tal como se indica a continuación:

- Si establece UACIPurgePriorBlackListOnLoad= 0, se conservará todo el contenido de la tabla de lista blanca.
- Si establece UACIPurgePriorWhiteListOnLoad= 1, se eliminará el contenido de la tabla, y se cargará desde la base de datos el contenido de la lista blanca o de la lista negra del ID de audiencia. Una vez completado, se iniciará la resegmentación.

Valor de retorno

El servidor de ejecución responde a setAudience con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getProfile

El método getProfile permite recuperar la información de perfil y temporal sobre el visitante que visita el punto de encuentro.

```
function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        InteractAPI.getProfile(ssId, callback);

}
```

- **session ID**-una serie que identifica el ID de sesión.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a onSuccess. Si el método ha fallado, la función de devolución de llamada llama a onError.

Valor de retorno

El servidor de ejecución responde a getProfile con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- OfferList
- ProfileRecord
- SessionID
- StatusCode

endSession

El método endSession marca el final de la sesión de ejecución. Cuando el servidor de ejecución recibe este método, el servidor de ejecución se registra en el historial, borra la memoria, etc.

```
function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        InteractAPI.endSession(ssId, callback);

}
```

- **session ID** - serie exclusiva que identifica la sesión.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a onSuccess. Si el método ha fallado, la función de devolución de llamada llama a onError.

Si no se invoca el método endSession, las sesiones de ejecución exceden el tiempo de espera. El período de tiempo de espera puede configurarse con la propiedad sessionTimeout.

Valor de retorno

El servidor de ejecución responde al método endSession con el objeto Respose con los siguientes atributos completados:

- SessionID
- ApiVersion
- OfferList
- Profile
- StatusCode
- AdvisoryMessages

setDebug

El método setDebug le permite establecer el nivel de detalle de registro para todas las rutas de código de la sesión.

```
function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        InteractAPI.setDebug(ssId, isDebug, callback);

}
```

- **ID_sesión**: cadena que identifica el ID de sesión.
- **debug**: booleano que habilita o inhabilita la información de depuración. Los valores válidos son true o false. Si es true, Interact registra información de depuración en el registro del servidor de ejecución.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a onSuccess. Si el método ha fallado, la función de devolución de llamada llama a onError.

Valor de retorno

El servidor de ejecución responde a setDebug con un objeto Response con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getVersion

El método `getVersion` devuelve la versión de la implementación actual del servidor de ejecución de Interact.

```
function callGetVersion(commandsToExecute, callback) {

        InteractAPI.getVersion(callback);


}
```

Se recomienda utilizar este método cuando inicialice el punto de encuentro con la API de Interact.

- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a `onSuccess`. Si el método ha fallado, la función de devolución de llamada llama a `onError`.

Valor de retorno

El servidor de ejecución responde a `getVersion` con un objeto de respuesta con los siguientes atributos completados:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

executeBatch

El método `executeBatch` permite ejecutar varios métodos con una sola solicitud en el servidor de ejecución.

```
function callExecuteBatch(commandsToExecute, callback) {

        if (!commandsToExecute)
            return ;

        InteractAPI.executeBatch(commandsToExecute.ssid,
        commandsToExecute.commands, callback);
}
```

- **session ID**-Una serie que identifica el ID de sesión. Este ID de sesión se utiliza para todos los comandos ejecutados por esta llamada de método.
- **commands**-Una matriz de objetos de mandato, una para cada mandato que desea realizar.
- **callback** - si el método se ha realizado correctamente, la función de devolución de llamada llama a `onSuccess`. Si el método ha fallado, la función de devolución de llamada llama a `onError`.

El resultado de invocar este método equivale a invocar explícitamente cada método en la matriz de comandos. Este método minimiza el número de solicitudes reales en el servidor de ejecución. El servidor de ejecución ejecuta cada método en serie; para cada llamada, se capturan los errores o avisos en el objeto de respuesta correspondiente a esa llamada de método. Si se encuentra un error, executeBatch continúa con el resto de las llamadas del lote. Si la ejecución de un método da como resultado un error, el estado de nivel superior del objeto BatchResponse refleja ese error. Si no se produce ningún error, el estado de nivel superior refleja todos los avisos que se han producido. Si no se produce ningún aviso, el estado de nivel superior refleja una ejecución satisfactoria del lote.

Valor de retorno

El servidor de ejecución responde a `executeBatch` con un objeto BatchResponse.

Ejemplo 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) {

    //leer sesión de inicio configurada
    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;

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

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

    //crear solicitudes obtener oferta
    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);
}

//recortar serie
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) {

    }
```

Ejemplo de objeto onSuccesss de JavaScript de respuesta

Este ejemplo muestra las tres variables para el objeto JavaScript de respuesta: offerLists, messages y profile.

`offerList` devuelve una lista no nula si llama a `getOffer` o `getOffersForMultipleInteractionPoints` como API o como parte de los mandatos de proceso por lotes. Siempre deberá comprobar si esto es nulo antes de realizar cualquier operación en esta variable.

Siembre deberá comprobar el estado de la respuesta JavaScript de `messages`.

`Profile` ha devuelto un no nulo si utiliza `getProfile` como API o como parte de los mandatos de proceso por lotes. Si no utiliza `getProfile`, puede ignorar esta variable. Siempre deberá comprobar si esto es nulo antes de realizar cualquier operación en esta variable.

```
function onSuccess(response)
InteractAPI.ResponseTransUtil._buildResponse = function(response) {
        'use strict';

        if (!response) return null;

        var offerList = null;
        //transformar offerLists a objetos JS
        if (response.offerLists) {
            offerList = [];
            for (var ofListCt=0; ofListCt<response.offerLists.length;ofListCt++) {
                var ofListObj = this._buildOfferList(response.offerLists[ofListCt]);
                if (ofListObj) offerList.push(ofListObj);
            }
        }

        var messages = null;
        //transformar mensajes a objetos JS
        if (response.messages) {
            messages = [];
```

```
            for (var msgCt=0; msgCt<response.messages.length;msgCt++) {
                var msgObj = this._buildAdvisoryMessage(response.messages[msgCt]);
                if (msgObj) messages.push(msgObj);
            }
        }

        var profile = null;
        //transformar nvps de perfil a objetos JS
        if (response.profile) {
            profile = [];
            for (var nvpCt=0; nvpCt<response.profile.length;nvpCt++) {
                var nvpObj = this._buildNameValuePair(response.profile[nvpCt]);
                if (nvpObj) profile.push(nvpObj);
            }
        }

        return InteractAPI.Response.create(response.sessionId,
                                           response.statusCode, offerList,
                                           profile, response.version,
                                           messages) ;
    };
```

Capítulo 9. Acerca de la API de ExternalCallout

Interact ofrece una macro ampliable, EXTERNALCALLOUT, que se utiliza con los diagramas de flujo interactivos. Esta macro permite ejecutar la lógica personalizada para comunicarse con sistemas externos durante las ejecuciones del diagrama de flujo. Por ejemplo, si desea calcular la puntuación de crédito de un cliente durante una ejecución de diagrama de flujo, puede crear una clase Java (una llamada) y, a continuación, utilizar la macro EXTERNALCALLOUT en un proceso Selección en el diagrama de flujo interactivo para obtener la puntuación de crédito de la llamada.

La configuración de EXTERNALCALLOUT consta de dos pasos principales. En primer lugar, debe crear una clase Java que implemente la API ExternalCallout. En segundo lugar, debe configurar las propiedades de configuración necesarias de la Marketing Platform en el servidor de ejecución en la categoría Interact | flowchart | ExternalCallouts.

Además de la información de esta sección, hay disponible un JavaDoc para la API ExternalCallout para cualquier servidor de ejecución de Interact en el directorio Interact/docs/externalCalloutJavaDoc.

Interfaz IAffiniumExternalCallout

La API de ExternalCallout está contenida en la interfaz IAffiniumExternalCallout. Debe implementar la interfaz IAffiniumExternalCallout para utilizar la macro EXTERNALCALLOUT.

La clase que implementa IAffiniumExternalCallout debe tener un constructor que pueda inicializar el servidor de ejecución.

- Si no hay constructores en la clase, el compilador Java crea un constructor predeterminado y ya es suficiente.
- Si hay constructores con argumentos, debe proporcionarse un constructor público sin argumento, que será utilizado por el servidor de ejecución.

Cuando desarrolle la llamada externa, recuerde lo siguiente:

- Cada evaluación de expresión con una llamada externa crea una nueva instancia de la clase. Debe gestionar los problemas de seguridad de subprocesos para los miembros estáticos en la clase.
- Si la llamada externa utiliza recursos del sistema como, por ejemplo, archivos o una conexión de base de datos, debe gestionar las conexiones. El servidor de ejecución no tiene un recurso para limpiar las conexiones automáticamente.

Debe compilar su implementación en interact_externalcallout.jar, que se encuentra en el directorio lib de la instalación del entorno de ejecución de IBM Interact.

IAffiniumExternalCallout habilita el servidor de ejecución para solicitar datos de la clase Java. La interfaz consta de cuatro métodos:

- getNumberOfArguments
- getValue
- initialize
- shutdown

Adición de un servicio web para su uso con la macro EXTERNALCALLOUT

Utilice este procedimiento para añadir un servicio web para su uso con la macro EXTERNALCALLOUT. La macro EXTERNALCALLOUT sólo reconoce las llamadas si ha definido la propiedades de configuración adecuadas.

Procedimiento

En la Marketing Platform del entorno de ejecución, añada o defina las siguientes propiedades de configuración en la categoría Interact > flowchart > externalCallouts.

Propiedad de configuración	Valor
Categoría externalCallouts	Crear una categoría para la llamada externa
class	Los nombres de clase para la llamada externa
classpath	La ruta de clase para los archivos de clase de llamada externa
Categoría Parameter Data	Si la llamada externa requiere parámetros, cree nuevas propiedades de configuración de parámetros para ellos y asígneles un valor a cada uno

getNumberOfArguments

El método getNumberOfArguments devuelve el número de argumentos esperados por la clase Java con la que se está integrando.

```
getNumberOfArguments()
```

Valor de retorno

El método getNumberOfArguments devuelve un entero.

Ejemplo

El ejemplo siguiente muestra la impresión del número de argumentos.

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

El método getValue ejecuta la funcionalidad principal de la llamada y devuelve los resultados.

```
getValue(audienceID, configData, arguments)
```

El método getValue requiere los siguientes parámetros:

- **audienceID**: un valor que identifica el ID de audiencia.
- **configData**: una correlación con pares clave-valor de datos de configuración necesarios para la llamada.
- **arguments**: los argumentos necesarios para la llamada. Cada argumento puede ser Cadena, Doble, Fecha o una Lista de alguno de estos. Una argumento de lista puede contener valores nulos. No obstante, una lista no puede contener, por ejemplo, una cadena y un doble.

Debe ejecutarse la comprobación de tipos de argumentos en su implementación.

Si el método `getValue` falla por alguna razón, devuelve `CalloutException`.

Valor de retorno

El método getValue devuelve una lista de cadenas.

Ejemplo

```
public List<String> getValue(AudienceId audienceId, Map<String,
  String> configurationData, Object... arguments) throws CalloutException
{
   Long customerId = (Long) audienceId.getComponentValue("Customer");
   // ahora consulte scoreQueryUtility para obtener la puntuación
   // de crédito de customerId
   Double score = scoreQueryUtility.query(customerId);
   String str = Double.toString(score);
   List<String> list = new LinkedList<String>();
   list.add(str);
   return list;
}
```

initialize

El método `initialize` se invoca una vez cuando se inicia el servidor de ejecución. Si hay operaciones que pueden obstaculizar el rendimiento durante el tiempo de ejecución como, por ejemplo, la carga de una tabla de base de datos, debe ejecutarlas este método.

```
initialize(configData)
```

El método `initialize` requiere el siguiente parámetro:

- **configData**: una correlación con pares clave-valor de datos de configuración necesarios para la llamada.

  Interact lee estos valores de los parámetros de llamada externa definidos en la categoría `Interact > Flowchart > External Callouts > [External Callout] > Parameter Data`.

Si el método `initialize` falla por alguna razón, devuelve `CalloutException`.

Valor de retorno

Ninguno.

Ejemplo

```
public void initialize(Map<String, String> configurationData) throws CalloutException
{
      // configurationData tiene los pares de clave-valor específicos del entorno
    // donde se ejecuta el servidor
      // inicializar scoreQueryUtility aquí
}
```

shutdown

El método `shutdown` se invoca una vez cuando se cierra el servidor de ejecución. Si hay tareas de limpieza necesarias para la llamada, deben ejecutarse en este momento.

```
shutdown(configData)
```

El método `shutdown` requiere el siguiente parámetro:

- **configData**: una correlación con pares clave-valor de datos de configuración necesarios para la llamada.

Si el método `shutdown` falla por alguna razón, devuelve `CalloutException`.

Valor de retorno

Ninguno.

Ejemplo

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
     // cerrar scoreQueryUtility aquí
}
```

Ejemplo de API de ExternalCallout

Este ejemplo crea una llamada externa que obtiene una puntuación de crédito.

Cree una llamada externa para obtener una puntuación de crédito:

1. Cree un archivo denominado `GetCreditScore.java` con el siguiente contenido. En este archivo se supone que hay una clase denominada `ScoreQueryUtility` que capta una puntuación de una aplicación de modelado.

```
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
{
 // la clase que tiene la lógica para consultar un sistema externo
 // para obtener una puntuación de crédito de un clientes
  private static ScoreQueryUtility scoreQueryUtility;
 public void initialize(Map<String, String> configurationData) throws CalloutException
 {
    // configurationData tiene los pares de clave-valor específicos
    // del entorno donde se ejecuta el servidor
    // inicializar scoreQueryUtility aquí
 }

 public void shutdown(Map<String, String> configurationData) throws CalloutException
 {
  // cerrar scoreQueryUtility aquí
 }

 public int getNumberOfArguments()
 {
    // no espere ningún argumento adicional aparte del ID de cliente
  return 0;
 }

  public List<String> getValue(AudienceId audienceId, Map<String, String> configurationData,
    Object... arguments) throws CalloutException
 {
    Long customerId = (Long) audienceId.getComponentValue("Customer");
    // ahora consulte scoreQueryUtility para obtener la puntuación de crédito de 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` en `GetCreditScore.class`.
3. Cree un archivo JAR llamado `creditscore.jar` que contenga `GetCreditScore.class` y los otros archivos de clase que utiliza.
4. Copie el archivo JAR en alguna ubicación en el servidor de ejecución, por ejemplo, `/data/interact/creditscore.jar`.
5. Cree una llamada externa con el nombre `GetCreditScore` y una ruta de clases `/data/interact/creditscore.jar` en la categoría `externalCallouts` en la página Gestionar configuraciones.
6. En un diagrama de flujo interactivo, la llamada se puede utilizar como `EXTERNALCALLOUT('GetCreditScore')`.

Interfaz IInteractProfileDataService

La API de Profile Data Services está contenida en la interfaz `iInteractProfileDataService`. Esta interfaz le permite importar datos jerárquicos a una sesión de Interact a través de uno o más orígenes de datos externos (tales como un archivo sin formato, servicio web, etc) en el momento en que se inicia la sesión de Interact o cambia el ID de audiencia de una sesión de Interact.

Para desarrollar datos jerárquicos utilizando la API de Profile Data Services, debe escribir una clase Java que extrae información de cualquier origen de datos y la correlaciona con un objeto `ISessionDataRootNode`; luego haga referencia a estos datos correlacionados utilizando la macro EXTERNALCALLOUT en un proceso Selección de un diagrama de flujo interactivo.

Debe compilar su implementación en `interact_externalcallout.jar`, que se encuentra en el directorio `lib` de la instalación del entorno de ejecución de IBM Interact.

Para obtener un conjunto completo de documentación Javadoc para utilizar esta interfaz, vea los archivos contenidos en *directorio_inicio_Interact*`/docs/externalCalloutJavaDoc` con cualquier navegador web.

Para ver una implementación de muestra de cómo utilizar Profile Data Services, así como descripciones de cómo se implementó el ejemplo, consulte *directorio_inicio_Interact*`/samples/externalcallout/XMLProfileDataService.java`.

Nota: La implementación de ejemplo está pensada para que se utilice solamente como ejemplo. No debería utilizar este ejemplo en su implementación.

Adición de un origen de datos para utilizarlo con Profile Data Services

Utilice este procedimiento para añadir un origen de datos para utilizarlo con Profile Data Services.

Acerca de esta tarea

La macro EXTERNALCALLOUT reconoce un origen de datos para una importación de datos jerárquicos de Profile Data Services sólo si se han definido las propiedades de configuración apropiadas.

Procedimiento

En la Marketing Platform correspondiente al entorno de ejecución, añada o defina las propiedades de configuración siguientes en la categoría Interact > Perfil > Niveles de audiencia > [AudienceLevelName] > Profile Data Services.

Propiedad de configuración	Valor
Categoría Nombre de categoría nuevo	Nombre del origen de datos que está definiendo. El nombre que especifique aquí debe ser exclusivo entre los orígenes de datos para un mismo nivel de audiencia.
enabled	Indica si el origen de datos está habilitado para el nivel de audiencia en el que está definido.
className	Nombre completo de la clase de origen de datos que implementa IInteractProfileDataService
classPath	La ruta de clases de los archivos de clase de Profile Data Services. Si no especifica un valor, se utiliza de forma predeterminada la ruta de clases del servidor de aplicaciones
Categoría prioridad	Prioridad del origen de datos dentro de este nivel de audiencia. Debe ser un valor exclusivo entre todos los orígenes de datos para cada nivel de audiencia. (Es decir, si la prioridad se establece en 100 para un origen de datos, ningún otro origen de datos dentro del nivel de audiencia puede tener una prioridad de 100).

Interfaz IParameterizableCallout

La API Parameterizable Callout está contenida en la interfaz IParameterizableCallout.

Esta interfaz es la interfaz base de las interfaces de la API expuesta que puede aceptar parámetros de la configuración mediante Marketing Platform. Dado que esta es una interfaz base, no se debe implementar directamente. El parámetro se recupera a partir de los nodos hijo del nodo Parameter Data bajo la categoría que hace referencia a esta implementación. En el ejemplo siguiente, ESB es una implementación personalizada del servicio de datos de perfil, que a su vez implementa la interfaz IParameterizableCallout. Los parámetros endPoint y login, junto con sus valores, se pasan a esta clase de implementación cuando el motor de Interact la intenta inicializar y finalizar.

```
Profile Data Services
...ESB
   ...Parameter Data
      ...endPoint
      ...login
```

La interfaz consta de dos métodos:

- initialize
- shutdown

initialize

El método initialize inicializa esta clase de implementación.

```
void initialize(java.util.Map<java.lang.String,java.lang.String> configurationData)
                throws CalloutException
```

El método initialize requiere el siguiente parámetro:

- **configurationData**: una correlación con pares de nombre y valor de los parámetros configurados por los usuarios

Throws

CalloutException

shutdown

El método shutdown concluye esta clase de implementación.

```
void shutdown(java.util.Map<java.lang.String,java.lang.String> configurationData)
              throws CalloutException
```

El método shutdown requiere el siguiente parámetro:

- **configurationData**: una correlación con pares de nombre y valor de los parámetros configurados por los usuarios

Throws

CalloutException

Interfaz ITriggeredMessageAction

La API de Triggered Message Action está contenida en la interfaz ITriggeredMessageAction. Esta interfaz le permite obtener y establecer el nombre de esta instancia.

La interfaz ITriggeredMessageAction sirve como interfaz base para otras interfaces y no se debe implementar nunca directamente.

La interfaz consta de dos métodos:

- getName
- setName

getName

El método getName devuelve el nombre de la instancia ITriggeredMessageAction.

```
java.lang.String getName()
```

setName

El método setName establece el nombre de la instancia ITriggeredMessageAction.

```
void setName(java.lang.String name)
```

Mientras inicializa la clase de implementación de esta interfaz, Interact establece el nombre de la interfaz con el nombre especificado en la interfaz de usuario de configuración.

En el ejemplo siguiente, el nombre de esta pasarela es InteractLog.

```
triggeredMessage
           ...gateways
              ...InteractLog
```

El método setName requiere el parámetro siguiente:

- name: El nombre que desea establecer para la instancia de ITriggeredMessageAction.

Interfaz IChannelSelector

La API de Channel Selector está contenida en la interfaz IChannelSelector. Esta interfaz le permite seleccionar canales de salida en función de la oferta que se ha de enviar y los atributos de la sesión.

Para ver una implementación de muestra de cómo utilizar Triggered Message Action, así como descripciones de cómo se ha implementado el ejemplo, consulte *directorio_inicio_Interact*/samples/triggeredmessage/SampleChannelSelector.java.

Nota: La implementación de ejemplo está pensada para que se utilice solamente como ejemplo. No debería utilizar este ejemplo en su implementación.

Debe intentar utilizar esta implementación, en lugar de escribir la suya propia.

La interfaz consta de un método:

- selectChannels

selectChannels

El método selectChannels selecciona los canales de salida a los que debe enviarse la oferta que se ha pasado con la interfaz IChannelSelector.

```
java.util.List<java.lang.String> selectChannels
                  (java.util.Map<java.lang.String,java.util.Map<java.lang.String,
                                 java.lang.Object>> availableChannels,
                                 com.unicacorp.interact.api.Offer offer,
                                 com.unicacorp.interact.treatment.
                                 optimization.IInteractSessionData sessionData)
```

Interact intenta enviar esta oferta a todos estos canales devueltos.

El método selectChannels requiere los parámetros siguientes:

- **availableChannels**: una correlación de canales de salida, que se configuran en la interfaz de usuario de Mensajes desencadenados en los valores de tiempo de diseño de Interact. En cada entrada de la correlación, la clave es el nombre del canal y el valor son los parámetros configurados para dicho canal en el tiempo de diseño de Interact. El orden de iteración de esta correlación coincide con el orden definido en dicha interfaz de usuario. Si se utiliza el canal preferido del perfil en la interfaz de usuario de mensajes desencadenados, se sustituye por el canal real antes de que se invoque este método. Además, si aparece varias veces el mismo canal en la interfaz de usuario, solo se conserva la aparición de mayor prioridad y se elimina todos los duplicados.
- **offer**: la oferta que se ha de entregar
- **sessionData**: los atributos almacenados actualmente en la sesión de Interact asociada

Interfaz IDispatcher

La API Dispatcher está contenida en la interfaz `IDispatcher`. Esta interfaz envía ofertas a las pasarelas de destino.

Dado que solo hay una instancia de esta clase para cada asignador configurado, la implementación de esta interfaz no debe tener estado desde la perspectiva de Interact.

Para ver una implementación de muestra de cómo utilizar Triggered Message Action, así como descripciones de cómo se ha implementado el ejemplo, consulte *directorio_inicio_Interact*`/samples/triggeredmessage/SampleDispatcher.java`.

Nota: La implementación de ejemplo está pensada para que se utilice solamente como ejemplo. No debería utilizar este ejemplo en su implementación.

Debe intentar utilizar esta implementación, en lugar de escribir la suya propia.

La interfaz consta de un método:

- `dispatch`

dispatch

El método `dispatch` envía ofertas a las pasarelas de destinos en la interfaz `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
```

Una vez seleccionados los canales para una oferta candidata, Interact intenta enviar las ofertas candidatas a los manejadores asociados al canal. Se intentan los manejadores en función de las prioridades definidas, de alta a baja. Para cada manejador, Interact invoca este método del asignador configurado. Depende de la implementación de esta instancia del asignador cómo se direcciona la oferta a la pasarela de destino, la cual se configura en el mismo manejador. Si como resultado de la evaluación del mismo mensaje desencadenado se envían varias ofertas al mismo manejador, Interact intenta enviar todas estas ofertas en un lote.

El método `dispatch` requiere los parámetros siguientes:

- **channel**: el canal de salida al que se envían estas ofertas
- **gatewayName**: el nombre de la pasarela de destino
- **offers**: las ofertas que se han de enviar a la pasarela en un lote
- **profileData**: los atributos de perfil que rellena `IGateway.validate` y se pasan a `IGateway.deliver`

Valor de retorno

Se devuelve el método `dispatch` si la asignación se ha realizado correctamente o ha fallado.

Throws

`com.unicacorp.interact.exceptions.InteractException`

Interfaz IGateway

La API Gateway está contenida en la interfaz `IGateway`. Esta interfaz recibe ofertas desde Interact y envía las ofertas a su destino.

Cada implementación de esta interfaz se comunica con un destino determinado. El destino debe transformar los datos, rellenar los atributos y realizar otras tareas necesarias relacionadas con los destinos.

Para ver una implementación de muestra de cómo utilizar Triggered Message Action, así como descripciones de cómo se ha implementado el ejemplo, consulte *directorio_inicio_Interact*`/samples/triggeredmessage/SampleOutboundGateway.java`.

Nota: La implementación de ejemplo está pensada para que se utilice solamente como ejemplo. No debería utilizar este ejemplo en su implementación.

La interfaz consta de dos métodos:

- `deliver`
- `validate`

deliver

El método `deliver` se invoca para enviar la o las ofertas a un destino en la interfaz `IGateway`.

```
void deliver(java.util.Collection<com.unicacorp.interact.api.Offer> offers,
                    com.unicacorp.interact.api.NameValuePair[] profileData,
                    java.lang.String channel)
```

El método `deliver` requiere los siguientes parámetros:

- **offers**: la oferta que se ha de enviar
- **profileData**: los atributos de perfil que rellena el método validate en `parameterMap`
- **channel**: el canal de salida al que se enviarán las ofertas

validate

El método `validate` valida las ofertas candidatas en la interfaz `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)
```

El motor de Interact invoca este método para validar las ofertas candidatas. La implementación de este método debe comprobar las ofertas, los atributos de la oferta y los atributos de sesión con los requisitos del destino para determinar qué oferta u ofertas se han de enviar a través de esta pasarela. Además, es posible que añada parámetros necesarios a la correlación que se ha pasado, la cual se devuelve de nuevo al método deliver.

El método `validate` requiere los siguientes parámetros:

- **sessionData**: los atributos almacenados actualmente en la sesión de Interact asociada

- **candidateOffers**: las ofertas que ha seleccionado Interact en función del método de selección de ofertas, sus parámetros y otros factores. Estas ofertas son elegibles para su entrega desde Interact, pero continúan estando sujetas a las pasarelas.
- **parameterMap**: una correlación que debe utilizar la implementación de este método para pasar los parámetros al método de entrega.
- **channel**: el canal de salida al que se enviarán las ofertas

Capítulo 10. Utilidades de IBM Interact

Esta sección describe los programas de utilidad de administración disponibles en Interact.

Ejecución del programa de utilidad de despliegue (runDeployment.sh/.bat)

La herramienta de línea de mandatos `runDeployment` le permite desplegar un canal interactivo para un grupo de servidores determinado desde la línea de mandatos, utilizando los valores proporcionados por un archivo `deployment.properties` que describe todos los parámetros posibles y reside en la misma ubicación que la propia herramienta `runDeployment`. La capacidad de ejecutar un despliegue de canal interactivo desde la línea de mandatos es especialmente útil cuando se utiliza la característica `OffersBySQL`. Por ejemplo, puede configurar un diagrama de flujo por lotes de Campaign para que se ejecute de forma periódica. Cuando finaliza la ejecución del diagrama de flujo, se puede activar un desencadenante para inicializar el despliegue de las ofertas en la tabla `OffersBySQL` mediante la herramienta de línea de mandatos.

Descripción

La herramienta de línea de mandatos `runDeployment` se instala automáticamente en el servidor de tiempo de tiempo de diseño de Interact, en la ubicación siguiente:

directorio_inicio_Interact`/interactDT/tools/deployment/runDeployment.sh` (o `runDeployment.bat` en un servidor Windows)

El único argumento que se pasa al mandato es la ubicación de un archivo denominado `deployment.properties` que describe todos los parámetros posibles necesarios para desplegar la combinación canal interactivo/grupo de servidores de ejecución. Se proporciona un archivo de muestra para referencia.

Nota: Antes de utilizar el programa de utilidad `runDeployment`, primero debe editarlo con un editor de texto cualquiera para proporcionar la ubicación del entorno de ejecución Java en el servidor. Por ejemplo, puede especificar *directorio_inicio_Interact*`/jre` o *directorio_inicio_Platform*`/jre` como vía de acceso si cualquiera de esos dos directorios contiene el entorno de ejecución Java que desea que sea utilizado por el programa de utilidad. Como alternativa, puede proporcionar la ruta de cualquier entorno de ejecución Java que se pueda utilizar con este release de los productos IBM .

Utilización del programa de utilidad runDeployment en un entorno seguro (SSL)

Para utilizar el programa de utilidad runDeployment cuando se ha habilitado la seguridad en el servidor de Interact (y por lo tanto la conexión se realiza a través de un puerto SSL), debe añadir la propiedad trustStore de Java como se muestra a continuación:

1. Cuando esté editando el archivo `deployment.properties` para el despliegue del canal interactivo, modifique la propiedad `deploymentURL` para utilizar el URL seguro de SSL, como en este ejemplo:

deploymentURL=https://*<HOST>*.*<DOMAIN>*:*<PORT>*/Campaign/interact/
InvokeDeploymentServlet

2. Edite el script runDeployment.sh o runDeployment.bat utilizando un editor de texto cualquiera para añadir el argumento siguiente a la línea que empieza por ${JAVA_HOME}:

   -Djavax.net.ssl.trustStore=*<TrustStorePath>*

   Por ejemplo, la línea podría tener este aspecto después de añadir el argumento trustStore:

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

   Sustituya *<TrustStorePath>* por la ruta del almacén de confianza de SSL.

Ejecución del programa de utilidad

Después de haber editado el programa de utilidad para proporcionar el entorno de ejecución Java, y haber personalizado una copia del archivo deployment.properties de acuerdo con el entorno utilizado, puede ejecutar el programa de utilidad con este mandato:

Interact_home/interactDT/tools/deployment/runDeployment.sh
deployment.properties

Sustituya *directorio_inicio_Interact* por el valor real de la instalación de tiempo de diseño de Interact, y sustituya *deployment.properties* por la ruta y el nombre del archivo de propiedades que ha personalizado para este despliegue.

Archivo deployment.properties de muestra

El archivo deployment.properties de muestra contiene una lista, en forma de comentario, de todos los parámetros que debe personalizar para que se adecuen al entorno utilizado. El archivo de muestra también contiene comentarios que explican lo que hace cada parámetro, y por qué puede ser necesario personalizar un valor determinado.

```
##############################################################################
#
# The following properties feed into the InvokeDeploymentClient program.
# The program will look for a deploymentURL setting.  The program will post a
# request against that url; all other settings are posted as parameters in
# that request. The program then checks the status of the deployment and
# returns back when the deployment is at a terminal state (or if the
# specified waitTime has been reached).
#
# the output of the program will be of this format:
# <STATE> : <Misc Detail>
#
# where state can be one of the following:
# ERROR
# RUNNING
# SUCCESS
#
# Misc Detail is data that would normally populate the status message area
#  in the deployment gui of the IC summary page. NOTE: HTML tags may exist
# in the Misc Detail
#
##############################################################################

##############################################################################
# deploymentURL: url to the InvokeDeployment servlet that resides in Interact
```

```
# Design time.  should be in the following format:
# http://dt_host:port/Campaign/interact/InvokeDeploymentServlet
################################################################################
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet

################################################################################
# dtLogin:  this is the login that you would use to login to the Design Time if
# you had wanted to deploy the IC via the deployment gui inside the IC summary
# page.
################################################################################
dtLogin=asm_admin

################################################################################
# dtPW:  this is the PW that goes along with the dtLogin
################################################################################
dtPW=

################################################################################
# icName:  this is the name of the Interactive Channel that you want to deploy
################################################################################
icName=ic1

################################################################################
# partition:  this is the name of the partition
################################################################################
partition=partition1

################################################################################
# request:  this is the type of request that you want this tool to execute
# currently, there two behaviors.  If the value is "deploy", then the deployment
# will be executed.  All other values would cause the tool to simply return the
# status of the last deployment of the specified IC.
################################################################################
request=deploy

################################################################################
# serverGroup:  this is the name of the server group that you would like to
# deploy the IC.
################################################################################
serverGroup=defaultServerGroup

################################################################################
# serverGroupType:  this will indicate whether or not this deployment is going
# against production server group or a test server group. 1 denotes production
# 2 denotes test.
################################################################################
serverGroupType=1

################################################################################
# rtLogin:  this is the account used to authenticate against the server group
# that you are deploying to.
################################################################################
rtLogin=asm_admin

################################################################################
# rtPW:  this is the password associated to the rtLogin
################################################################################
rtPW=

################################################################################
# waitTime:  Once the tool submits the deployment request, the tool will check
# the status of the deployment.  If the deployment has not completed (or
# failed), then the tool will continue to poll the system for the status until
# a completed state has been reached, OR until the specified waitTime (in
# seconds) has been reached.
################################################################################
waitTime=5
```

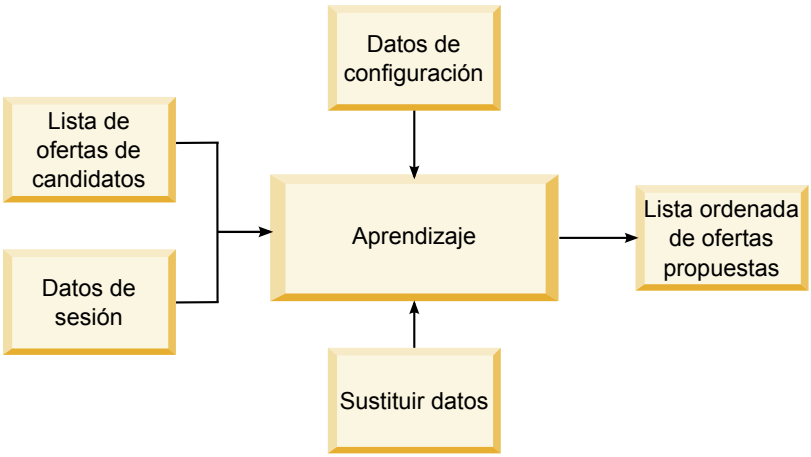
```
################################################################################
# pollTime: If the status of a deployment is still in running state, then the
# tool will continue to check the status.  It will sleep in between status
# checks a number of seconds based on the pollTime setting .
################################################################################
pollTime=3

################################################################################
# global: Setting to false will make the tool NOT deploy the global settings.
# Non-availability of the property will still deploy the global settings.
################################################################################
global=true
```

Capítulo 11. Acerca de la API de aprendizaje

Interact ofrece un módulo de aprendizaje que utiliza un algoritmo Naive Bayesian para supervisar las acciones de los visitantes y proponer las ofertas óptimas (en términos de aceptación). Puede implementar la misma interfaz Java con sus propios algoritmos utilizando la API de aprendizaje para crear su propio módulo de aprendizaje.

Nota: Si utiliza el aprendizaje externo, los informes de ejemplo sobre el aprendizaje (los informes Detalles de aprendizaje de ofertas interactivas y Análisis de elevación de segmento interactivo) no devuelven datos válidos.

En el nivel más simple, la API de aprendizaje proporciona métodos para recopilar datos del entorno de ejecución y devolver una lista ordenada de ofertas recomendadas.



Puede recopilar los siguientes datos de Interact

- Datos de contactos de ofertas
- Datos de aceptación de ofertas
- Todos los datos de sesión
- Datos de ofertas específicos de Campaign
- Propiedades de configuración definidas en la categoría `learning` para el entorno de diseño y la categoría `offerserving` para el entorno de ejecución

Puede utilizar estos datos en los algoritmos para crear una lista de ofertas propuestas. A continuación, devuelva una lista de ofertas recomendadas, en el orden de mayor a menor recomendación.

Aunque no se muestra en el diagrama, también puede utilizar la API de aprendizaje para recopilar datos para su implementación de aprendizaje. Puede conservar estos datos en la memoria o registrarlos en un archivo o una base de datos para continuar el análisis.

Después de crear las clases Java, puede convertirlas en archivos jar. Una vez creados los archivos jar, también debe configurar el entorno de ejecución para reconocer el módulo de aprendizaje externo editando las propiedades de

configuración. Debe copiar los archivos jar o las clases Java para cada servidor de ejecución utilizando el módulo de aprendizaje externo.

Además de la información de esta guía, hay disponible un JavaDoc para la API del optimizador de aprendizaje en cualquier servidor de ejecución en el directorio `Interact/docs/learningOptimizerJavaDoc`.

Debe compilar su implementación en `interact_learning.jar`, que se encuentra en el directorio `lib` de la instalación del entorno de ejecución de Interact.

Cuando escribe su implementación de aprendizaje personalizada, debe tener en cuenta las siguientes directrices.

- El rendimiento es crítico.
- Debe trabajar con varios subprocesos y garantizar la seguridad para los subprocesos.
- Debe gestionar todos los recursos externos teniendo en cuenta los modos de error y el rendimiento.
- Utilice las excepciones, el registro (log4j) y la memoria según corresponda.

Configuración del entorno de ejecución para reconocer módulos de aprendizaje externos

Puede utilizar la API Java de aprendizaje para escribir su propio módulo de aprendizaje. Debe configurar el entorno de ejecución para que reconozca su utilidad de aprendizaje en la Marketing Platform.

Acerca de esta tarea

Debe reiniciar el servidor de ejecución de Interact para que estos cambios entren en vigor.

Procedimiento

1. En la Marketing Platform del entorno de ejecución, edite las siguientes propiedades de configuración en la categoría `Interact > offerserving`. Las propiedades de configuración de la API del optimizador de aprendizaje se encuentran en la categoría `Interact > offerserving > External Learning Config`.

Propiedad de configuración	Valor
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Nombre de clase del aprendizaje externo
`externalLearningClassPath`	La ruta de la clase o los archivos JAR en el servidor de ejecución para el aprendizaje externo. Si utiliza un grupo de servidores y todos los servidores de ejecución hacen referencia a la misma instancia de Marketing Platform, cada servidor debe tener una copia de la clase o los archivos JAR en la misma ubicación.

2. Reinicie el servidor de ejecución de Interact para que estos cambios entren en vigor.

Interfaz ILearning

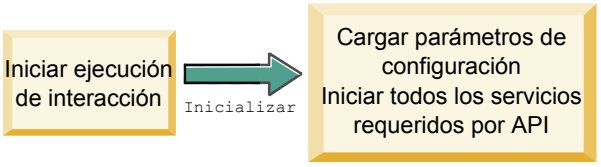
La API de aprendizaje se construye a partir de la interfaz `ILearning`. Debe implementar la interfaz `ILearning` para dar soporte a la lógica personalizada del módulo de aprendizaje.

Entre otras cosas, la interfaz `ILearning` permite recopilar datos del entorno de ejecución para la clase Java, y para enviar una lista de ofertas recomendadas al servidor de ejecución.

initialize

El método `initialize` se invoca una vez cuando se inicia el servidor de ejecución. Si hay operaciones que no se deben repetir, pero pueden obstaculizar el rendimiento durante el tiempo de ejecución como, por ejemplo, la carga de datos estáticos de una tabla de base de datos, debe ejecutarlas este método.

```
initialize(ILearningConfig config, boolean debug)
```



- **config**: un objeto `ILearningConfig` define todas las propiedades de configuración relevantes para el aprendizaje.
- **debug**: un valor booleano. Si es `true`, indica el nivel de detalle del nivel de registro en que se ha establecido el sistema del entorno de ejecución para la depuración. Para obtener los mejores resultados, seleccione este valor antes de escribir en un registro.

Si el método `initialize` falla por alguna razón, genera `LearningException`.

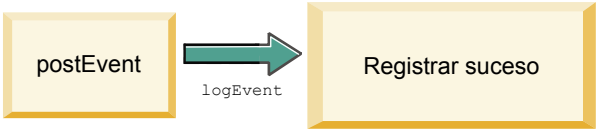
Valor de retorno

Ninguno.

logEvent

El servidor de ejecución invoca el método `logEvent` siempre que la API de Interact publica un evento que está configurado para registrarse como un contacto o una respuesta. Utilice este método para registrar datos de contacto y de respuesta en una base de datos o un archivo a efectos de informes y aprendizaje. Por ejemplo, si desea determinar mediante algoritmos la probabilidad de que un cliente acepte una oferta basándose en los criterios, utilice este método para registrar los datos.

```
logEvent(ILearningContext context,
             IOffer offer,
             IClientArgs clientArgs,
             IInteractSession session,
             boolean debug)
```



- **context**: un objeto `ILearningContext` que define el contexto de aprendizaje del evento, por ejemplo, contacto, aceptación o rechazo.
- **offer**: un objeto `IOffer` que define la oferta sobre la que se está registrando este evento.
- **clientArgs**: un objeto `IClientArgs` que define los parámetros. Actualmente, `logEvent` no necesita `clientArgs`, por lo que este parámetro puede estar vacío.
- **session**: un objeto `IInteractSession` que define todos los datos de sesión.
- **debug**: un valor booleano. Si es `true`, indica el nivel de detalle del nivel de registro en que se ha establecido el sistema del entorno de ejecución para la depuración. Para obtener los mejores resultados, seleccione este valor antes de escribir en un registro.

Si el método `logEvent` falla, genera una `LearningException`.

Valor de retorno

Ninguno.

optimizeRecommendList

El método `optimizeRecommendList` debe utilizar la lista de ofertas recomendadas y los datos de sesión, y devolver una lista que contenga el número de ofertas solicitadas. El método `optimizeRecommendList` debe ordenar las ofertas de alguna manera, con su propio algoritmo de aprendizaje. La lista de ofertas debe ordenarse de modo que las ofertas que desee presentar primero estén al principio de la lista. Por ejemplo, si el algoritmo de aprendizaje proporciona una baja puntuación a las mejores ofertas, las ofertas deben ordenarse como 1, 2, 3. Si el algoritmo de aprendizaje proporciona una alta puntuación a las mejores ofertas, las ofertas deben ordenarse como 100, 99, 98.

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

El método `optimizeRecommendList` requiere los siguientes parámetros:

- **recList**: una lista de los objetos de tratamiento (ofertas) recomendados por el entorno de ejecución.
- **clientArg**: un objeto IClientArgs que contiene como mínimo el número de ofertas solicitadas por el entorno de ejecución.
- **session**: un objeto IInteractSession que contiene todos los datos de sesión.
- **debug**: un valor booleano. Si es `true`, indica el nivel de detalle del nivel de registro en que se ha establecido el sistema del entorno de ejecución para la depuración. Para obtener los mejores resultados, seleccione este valor antes de escribir en un registro.

Si el método `optimizeRecommendList` falla, genera una `LearningException`.

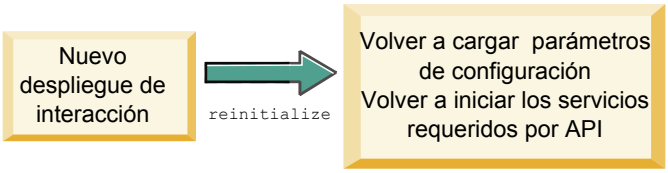
Valor de retorno

El método `optimizeRecommendList` devuelve una lista de objetos `ITreatment`.

reinitialize

El entorno de ejecución invoca el método `reinitialize` cada vez que hay un nuevo despliegue. Este método pasa todos los datos de configuración de aprendizaje. Si tiene servicios necesarios para la API de aprendizaje que leen propiedades de configuración, esta interfaz debe reiniciarlos.

```
reinitialize(ILearningConfig config,
                  boolean debug)
```



- **config**: un objeto `ILearningConfig` que contiene todas las propiedades de configuración.
- **debug**: un valor booleano. Si es `true`, indica el nivel de detalle del nivel de registro en que se ha establecido el sistema del entorno de ejecución para la depuración. Para obtener los mejores resultados, seleccione este valor antes de escribir en un registro.

Si el método `logEvent` falla, genera una `LearningException`.

Valor de retorno

Ninguno.

shutdown

El entorno de ejecución invoca el método `shutdown` cuando se cierra el servidor de ejecución. Si hay tareas de limpieza necesarias para el módulo de aprendizaje, deben ejecutarse en este momento.

```
shutdown(ILearningConfig config, boolean debug)
```

El método `shutdown` requiere los siguientes parámetros.

- **config**: un objeto `ILearningConfig` que define todas las propiedades de configuración.
- **debug**: un valor booleano. Si es `true`, indica el nivel de detalle del nivel de registro en que se ha establecido el sistema del entorno de ejecución para la depuración. Para obtener los mejores resultados, seleccione este valor antes de escribir en un registro.

Si el método `shutdown` falla por alguna razón, genera `LearningException`.

Valor de retorno

Ninguno.

Interfaz IAudienceID

La interfaz `IAudienceID` da soporte a la interfaz `IInteractSession`. Es una interfaz para el ID de audiencia. Como el ID de audiencia puede estar formado por varias partes, esta interfaz permite acceder a todos los elementos del ID de audiencia, así como al nombre del nivel de audiencia.

getAudienceLevel

El método `getAudienceLevel` devuelve el nivel de audiencia.

```
getAudienceLevel()
```

Valor de retorno

El método `getAudienceLevel` devuelve una cadena que define el nivel de audiencia.

getComponentNames

El método `getComponentNames` obtiene un conjunto de los nombres de los componentes que forman el ID de audiencia. Por ejemplo, si el ID de audiencia está formado por los valores `customerName` y `accountID`, `getComponentNames` devolverá un conjunto que contiene las cadenas customerName y accountID.

```
getComponentNames()
```

Valor de retorno

Un conjunto de cadenas que contienen los nombres de los componentes del ID de audiencia.

getComponentValue

El método `getComponentValue` devuelve el valor del componente denominado.

```
getComponentValue(String componentName)
```

- **nombreComponente**: una cadena que define el nombre del componente para el que desea recuperar el valor. Esta cadena no es sensible a las mayúsculas y minúsculas.

Valor de retorno

El método `getComponentValue` devuelve un objeto que define el valor del componente.

IClientArgs

La interfaz `IClientArgs` da soporte a la interfaz `ILearning`. Esta interfaz es una abstracción para cubrir los datos que se pasan al servidor desde el punto de encuentro que no están cubiertos por los datos de sesión. Por ejemplo, el número de ofertas solicitadas por el método `getOffers` de la API de Interact. Estos datos se almacenan en una correlación.

getValue

El método `getValue` devuelve el valor del elemento de correlación solicitado.

`getValue(int clientArgKey)`

Los siguientes elementos son necesarios en la correlación.

- **1**: NUMBER_OF_OFFERS_REQUESTED. El número de ofertas solicitadas por el método `getOffers` de la API de Interact. Esta constante devuelve un entero.

Valor de retorno

El método `getValue` devuelve un objeto que define el valor de la constante de correlación solicitada.

IInteractSession

La interfaz `IInteractSession` da soporte a la interfaz `ILearning`. Es una interfaz para la sesión actual en el entorno de ejecución.

getAudienceId

El método `getAudienceId` devuelve un objeto `AudienceID`. Utilice la interfaz `IAudienceID` para extraer los valores.

`getAudienceId()`

Valor de retorno

El método `getAudienceId` devuelve un objeto `AudienceID`.

getSessionData

El método `getSessionData` devuelve una correlación no modificable de datos de sesión, donde el nombre de la variable de sesión es la clave. El nombre de la variable de sesión está siempre en mayúsculas. Utilice la interfaz `IInteractSessionData` para extraer los valores.

`getSessionData()`

Valor de retorno

El método `getSessionData` devuelve un objeto `IInteractSessionData`.

Interfaz IInteractSessionData

La interfaz `IInteractSessionData` da soporte a la interfaz `ILearning`. Es una interfaz para los datos de sesión de ejecución del visitante actual. Los datos de sesión se almacenan como una lista de pares nombre-valor. También puede utilizar esta interfaz para cambiar el valor de los datos en la sesión de ejecución.

getDataType

El método `getDataType` devuelve el tipo de datos para el nombre de parámetro especificado.

`getDataType(string parameterName)`

Valor de retorno

El método `getDataType` devuelve un objeto InteractDataType. IntearctDataType es una enumeración Java representada por Desconocido, Cadena, Doble, Fecha o Lista.

getParameterNames

El método `getParameterNames` devuelve un conjunto de todos los nombres de los datos en la sesión actual.

```
getParameterNames()
```

Valor de retorno

El método `getParameterNames` devuelve un conjunto de nombres para los que se han establecido valores. Cada nombre del conjunto puede pasarse a getValue(String) para devolver un valor.

getValue

El método `getValue` devuelve el valor de objeto correspondiente al `parameterName` especificado. El objeto puede ser una Cadena, un Doble o una Fecha.

```
getValue(parameterName)
```

El método `getValue` requiere el siguiente parámetro:

- **nombreParámetro**: una cadena que define el nombre del par nombre-valor de los datos de sesión.

Valor de retorno

El método `getValue` devuelve un objeto que contiene el valor de parámetro especificado.

setValue

El método `setValue` permite establecer un valor para el `parameterName` especificado. El valor puede ser una Cadena, un Doble o una Fecha.

```
setValue(string parameterName, object value)
```

El método `setValue` requiere los siguientes parámetros:

- **nombreParámetro**: una cadena que define el nombre del par nombre-valor de los datos de sesión.
- **valor**: un objeto que define el valor del parámetro designado.

Valor de retorno

Ninguno.

ILearningAttribute

La interfaz `ILearningAttribute` da soporte a la interfaz `ILearningConfig`. Esta es una interfaz para los atributos de aprendizaje definidos en las propiedades de configuración en la categoría `learningAttributes`.

getName

El método `getName` devuelve el nombre del atributo de aprendizaje.

```
getName()
```

Valor de retorno

El método `getName` devuelve una cadena que define el nombre del atributo de aprendizaje.

ILearningConfig

La interfaz `ILearningConfig` da soporte a la interfaz `ILearning`. Es una interfaz para las propiedades de configuración de aprendizaje. Todos estos métodos devuelven el valor de la propiedad.

La interfaz consta de 15 métodos:

- **getAdditionalParameters**: devuelve una correlación de las propiedades adicionales definidas en la categoría `External Learning Config`
- **getAggregateStatsIntervalInMinutes**: devuelve un entero
- **getConfidenceLevel**: devuelve un entero
- **getDataSourceName**: devuelve una cadena
- **getDataSourceType**: devuelve una cadena
- **getInsertRawStatsIntervalInMinutes**: devuelve un entero
- **getLearningAttributes**: devuelve una lista de objetos `ILearningAttribute`
- **getMaxAttributeNames**: devuelve un entero
- **getMaxAttributeValues**: devuelve un entero
- **getMinPresentCountThreshold**: devuelve un entero
- **getOtherAttributeValue**: devuelve una cadena
- **getPercentRandomSelection**: devuelve un entero
- **getRecencyWeightingFactor**: devuelve un flotante
- **getRecencyWeightingPeriod**: devuelve un entero
- **isPruningEnabled**: devuelve un booleano

ILearningContext

La interfaz `ILearningContext` da soporte a la interfaz `ILearning`.

getLearningContext

El método `getLearningContext` devuelve una constante que indica si es un escenario de contactos, aceptaciones o rechazos.

```
getLearningContext()
```

- **1**-LOG_AS_CONTACT
- **2**-LOG_AS_ACCEPT
- **3**-LOG_AS_REJECT

4 y 5 están reservados para uso en el futuro.

Valor de retorno

El método `getLearningContext` devuelve un entero.

getResponseCode

El método getResponseCode devuelve el código de respuesta asignado a esta oferta. Este valor debe existir en la tabla UA_UsrResponseType en las tablas del sistema de Campaign.

```
getResponseCode()
```

Valor de retorno

El método getResponseCode devuelve una cadena que define el código de respuesta.

IOffer

La interfaz IOffer da soporte a la interfaz ITreatment. Es una interfaz para el objeto de oferta definido en el entorno de diseño. Utilice la interfaz IOffer para recopilar los detalles de oferta del entorno de ejecución.

getCreateDate

El método getCreateDate devuelve la fecha de creación de la oferta.

```
getCreateDate()
```

Valor de retorno

El método getCreateDate devuelve una fecha que define la fecha de creación de la oferta.

getEffectiveDateFlag

El método getEffectiveDateFlag devuelve un número que define la fecha efectiva de la oferta.

```
getEffectiveDateFlag()
```

- **0**: la fecha efectiva es una fecha absoluta, por ejemplo, 15 de marzo de 2010.
- **1**: la fecha efectiva es la fecha de recomendación.

Valor de retorno

El método getEffectiveDateFlag devuelve un entero que define la fecha efectiva de la oferta.

getExpirationDateFlag

El método getExpirationDateFlag devuelve un valor entero que describe la fecha de caducidad de la oferta.

```
getExpirationDateFlag()
```

- **0**: una fecha absoluta, por ejemplo, 15 de marzo de 2010.
- **1**: cierto número de días después de la recomendación, por ejemplo, 14.
- **2**: fin de mes después de la recomendación. Si una oferta se presenta el 31 de marzo, la oferta caducará ese día.

Valor de retorno

El método getExpirationDateFlag devuelve un entero que describe la fecha de caducidad de la oferta.

getOfferAttributes

El método `getOfferAttributes` devuelve atributos de oferta definidos para la oferta como un objeto `IOfferAttributes`.

```
getOfferAttributes()
```

Valor de retorno

El método `getOfferAttributes` devuelve un objeto `IOfferAttributes`.

getOfferCode

El método `getOfferCode` devuelve el código de oferta de la oferta tal como está definido en Campaign.

```
getOfferCode()
```

Valor de retorno

El método `getOfferCode` devuelve un objeto `IOfferCode`.

getOfferDescription

El método `getOfferDescription` devuelve la descripción de la oferta definida en Campaign.

```
getOfferDescription()
```

Valor de retorno

El método `getOfferDescription` devuelve una serie.

getOfferID

El método `getOfferID` devuelve el ID de oferta tal como está definido en Campaign.

```
getOfferID()
```

Valor de retorno

El método `getOfferID` devuelve un valor largo que define el ID de oferta.

getOfferName

El método `getOfferName` devuelve el nombre de la oferta tal como está definido en Campaign.

```
getOfferName()
```

Valor de retorno

El método `getOfferName` devuelve una serie.

getUpdateDate

El método `getUpdateDate` devuelve la fecha de la última actualización de la oferta.

```
getUpdateDate()
```

Valor de retorno

El método `getUpdateDate` devuelve una fecha que define cuándo se ha actualizado una oferta por última vez.

IOfferAttributes

La interfaz `IOfferAttributes` da soporte a la interfaz `IOffer`. Es una interfaz para los atributos de oferta definidos para una oferta en el entorno de diseño. Utilice la interfaz `IOfferAttributes` para recopilar los atributos de oferta del entorno de ejecución.

getParameterNames

El método `getParameterNames` devuelve una lista de los nombres de parámetros de oferta.

```
getParameterNames()
```

Valor de retorno

El método `getParameterNames` devuelve un conjunto que define la lista de nombres de parámetros de oferta.

getValue

El método `getValue` devuelve un objeto que define el valor del atributo de oferta.

```
getValue(String nombreParámetro)
```

El método `getValue` devuelve el valor del atributo de oferta especificado.

Valor de retorno

Interfaz IOfferCode

La interfaz `IOfferCode` da soporte a la interfaz `ILearning`. Es una interfaz para el código de oferta definido para una oferta en el entorno de diseño. Un código de oferta puede estar formado por una o varias cadenas. Utilice la interfaz `IOfferCode` para recopilar el código de oferta del entorno de ejecución.

getPartCount

El método `getPartCount` devuelve el número de partes que componen un código de oferta.

```
getPartCount()
```

Valor de retorno

El método `getPartCount` devuelve un valor entero que define el número de partes del código de oferta.

getParts

El método `getParts` obtiene una lista no modificable de las partes del código de oferta.

```
getParts()
```

Valor de retorno

El método `getParts` devuelve una lista no modificable de las partes del código de oferta.

LearningException

La clase `LearningException` da soporte a la interfaz `ILearning`. Algunos métodos en la interfaz requerirán que las implementaciones generen una `LearningException`, que es una subclase simple de `java.lang.Exception`. Se recomienda especialmente a efectos de depuración que la `LearningException` se construya con la excepción raíz, si existe alguna.

IScoreOverride

La interfaz `IScoreOverride` da soporte a la interfaz `ITreatment`. Esta interfaz permite leer los datos definidos en la tabla de anulación de puntuaciones o la tabla de ofertas predeterminadas.

getOfferCode

El método `getOfferCode` devuelve el valor de las columnas de código de oferta en la tabla de sustituciones de puntuación para este miembro de audiencia.

```
getOfferCode()
```

Valor de retorno

El método `getOfferCode` devuelve un objeto `IOfferCode` que define el valor de las columnas de código de oferta en la tabla de sustituciones de puntuación.

getParameterNames

El método `getParameterNames` devuelve la lista de parámetros.

```
getParameterNames()
```

Valor de retorno

El método `getParameterNames` devuelve un conjunto que define la lista de parámetros.

El método `IScoreOverride` contiene los parámetros siguientes. A menos que se indique lo contrario, estos parámetros son los mismos que la tabla de sustituciones de puntuación.

- ADJ_EXPLORE_SCORE_COLUMN
- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT: para sustituir el recuento actual estimado (durante el cálculo del peso de la oferta)
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN: para crear una expresión booleana para determinar la elegibilidad de la oferta

- PREDICATE_SCORE: para crear una expresión que genere una puntuación numérica
- SCORE_COLUMN
- ZONE_COLUMN

También puede hacer referencia a cualquier columna que añada a la tabla de ofertas predeterminadas o de sustituciones de puntuación utilizando el mismo nombre que la columna.

getValue

El método `getValue` devuelve el valor de la columna de zona de la tabla de sustituciones de puntuación para este miembro de audiencia.

```
getValue(String nombreParámetro)
```

- **nombreParámetro**: cadena que define el nombre del parámetro para el que desea el valor.

Valor de retorno

El método `getValue` devuelve un objeto que define el valor del parámetro solicitado.

ISelectionMethod

La interfaz `ISelection` indica el método utilizado para encontrar la lista recomendada. El valor predeterminado para el objeto de tratamiento es EXTERNAL_LEARNING, por lo que no tiene que establecer este valor. El valor se almacena en última instancia en el historial de contactos detallado para la creación de informes.

Puede ampliar esta interfaz más allá de las constantes existentes si desea almacenar los datos para su análisis posterior. Por ejemplo, puede crear dos módulos de aprendizaje diferentes e implementarlos en grupo de servidores diferentes. Puede ampliar la interfaz `ISelection` para incluir SERVER_GROUP_1 y SERVER_GROUP_2. A continuación, puede comparar los resultados de los dos módulos de aprendizaje.

Interfaz ITreatment

La interfaz `ITreatment` da soporte a la interfaz `ILearning` como una interfaz para la información de tratamiento. Un tratamiento representa la oferta asignada a una determinada celda tal como está definida en el entorno de diseño. A partir de esta interfaz, puede obtener información de celda y oferta, así como la puntuación de marketing asignada.

getCellCode

El método `getCellCode` devuelve el código de celda tal como está definido en Campaign. La celda es la celda asignada al segmento inteligente asociado a esta oferta.

```
getCellCode()
```

Valor de retorno

El método `getCellCode` devuelve una serie que define el código de celda.

getCellId

El método `getCellId` devuelve el ID interno de la celda tal como está definido en Campaign. La celda es la celda asignada al segmento inteligente asociado a esta oferta.

```
getOfferName()
```

Valor de retorno

El método `getCellId` devuelve un valor largo que define el ID de celda.

getCellName

El método `getCellName` devuelve el nombre de la celda tal como está definida en Campaign. La celda es la celda asignada al segmento inteligente asociado a esta oferta.

```
getCellName()
```

Valor de retorno

El método `getCellName` devuelve una serie que define el nombre de celda.

getLearningScore

El método `getLearningScore` devuelve la puntuación de este tratamiento.

```
getLearningScore()
```

La prioridad es la siguiente.

1. Devolver el valor de sustitución, si está presente en la correlación Sustituir valores especificada en `IScoreoveride.PREDICATE_SCORE_COLUMN`
2. Devolver puntuación de predicado si el valor no es nulo
3. Devolver la puntuación de los usuarios de marketing, si está presente en la correlación Sustituir valores especificada en `IScoreoveride.SCORE`
4. Devolver la puntuación de los usuarios de marketing

Valor de retorno

El método `getLearningScore` devuelve un entero que define la puntuación determinada por el algoritmo de aprendizaje.

getMarketerScore

El método `getMarketerScore` devuelve la puntuación del usuario de marketing definida mediante el graduador en la pestaña de estrategia de interacción para la oferta.

```
getMarketerScore()
```

Para recuperar la puntuación de un usuario de marketing definida mediante las opciones avanzadas de la pestaña de estrategia de interacción, utilice `getPredicateScore`.

Para recuperar la puntuación del usuario de marketing utilizada realmente por el tratamiento, utilice `getLearningScore`.

Valor de retorno

El método `getMarketerScore` devuelve un entero que define la puntuación del usuario de marketing.

getOffer

El método `getOffer` devuelve la oferta para el tratamiento.

```
getOffer()
```

Valor de retorno

El método `getOffer` devuelve un objeto `IOffer` que define la oferta para este tratamiento.

getOverrideValues

El método `getOverrideValues` devuelve las sustituciones definidas en las ofertas predeterminadas o tabla de sustituciones de puntuación.

```
getOverrideValues()
```

Valor de retorno

El método `getOverrideValues` devuelve un objeto `IScoreOverride`.

getPredicate

El método `getPredicate` devuelve el predicado definido por la columna de predicado de la tabla de ofertas predeterminadas, la tabla de sustituciones de puntuación o las opciones avanzadas de las reglas de tratamiento.

```
getPredicate()
```

Valor de retorno

El método `getPredicate` devuelve una serie que define el predicado definido por la columna de predicado de la tabla de ofertas predeterminadas, la tabla de sustituciones de puntuación o las opciones avanzadas de las reglas de tratamiento.

getPredicateScore

El método `getPredicateScore` devuelve la puntuación establecida por la columna de predicado de la tabla de ofertas predeterminadas, la tabla de sustituciones de puntuación o las opciones avanzadas de las reglas de tratamiento.

```
getPredicateScore()
```

Valor de retorno

El método `getPredicateScore` devuelve un valor doble que define la puntuación establecida por la columna de predicado de la tabla de ofertas predeterminadas, la tabla de sustituciones de puntuación o las opciones avanzadas de las reglas de tratamiento.

getScore

El método `getScore` devuelve la puntuación de marketing que está definida ya sea mediante la estrategia de interacción en Campaign o mediante la tabla de anulación de puntuaciones.

```
getScore()
```

El método getScore devuelve uno de los elementos siguientes:

- La puntuación de marketing de la oferta tal como está definida en la pestaña de estrategia de interacción en Campaign si la propiedad enableScoreOverrideLookup está establecida en false.
- La puntuación de la oferta tal como está definida en scoreOverrideTable si la propiedad enableScoreOverrideLookup se establece en true.

Valor de retorno

El método getScore devuelve un entero que representa la puntuación de la oferta.

getTreatmentCode

El método getTreatmentCode devuelve el código de tratamiento.

```
getTreatmentCode()
```

Valor de retorno

El método getTreatmentCode devuelve una serie que define el código de tratamiento.

setActualValueUsed

Utilice el método setActualValueUsed para definir qué valores se utilizan en las distintas etapas de la ejecución del algoritmo de aprendizaje.

```
setActualValueUsed(string nombreParm, object valor)
```

Por ejemplo, si utiliza este método para grabar en las tablas de historial de respuestas y de contactos, y modificar los informes de muestra existentes, puede incluir datos del algoritmo de aprendizaje en la creación de informes.

- **nombreParm**: cadena que define el nombre del parámetro que está estableciendo.
- **valor**: objeto que define el valor del parámetro que está estableciendo.

Valor de retorno

Ninguno.

Ejemplo de API de aprendizaje

Esta sección contiene una implementación de muestra de ILearningInterface. Tenga en cuenta que esta implementación es sólo una muestra y no está diseñada para utilizarse en un entorno de producción.

Este ejemplo realiza un seguimiento de los recuentos de aceptaciones y contactos, y utiliza la proporción de aceptaciones y contactos para una determinada oferta como la tasa de probabilidad de aceptación de la oferta. Las ofertas que no se presentan reciben una mayor prioridad de recomendación. Las ofertas con al menos un contacto se ordenan según la tasa de probabilidad de aceptación descendente.

En este ejemplo, todos los recuentos se mantienen en memoria. No es un escenario realista, ya que el servidor de ejecución se quedará sin memoria. En un escenario de producción real, los recuentos deben ser persistentes en una base de datos.

```
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;

/**
 * Es una implementación de muestra del optimizador de aprendizaje.
 * La interfaz ILearning puede encontrarse en la biblioteca interact.jar.
 *
 * Para utilizar realmente esta implementación, seleccione ExternalLearning como optimizationType en el nodo offerServing
 * de la aplicación de Interact en la configuración de plataforma. En el nodo offerserving también hay
 * una categoría External Learning Config, donde debe establecer el nombre de la clase en:
 * com.unicacorp.interact.samples.learning.v2.SampleLearning. Tenga en cuenta que esta implementación es sólo una muestra
 * y no está diseñada para utilizarse en un entorno de producción.
 *
 *
 * Este ejemplo realiza un seguimiento de los recuentos de aceptaciones y contactos, y utiliza la proporción de aceptaciones y contactos
 * para una determinada oferta como la tasa de probabilidad de aceptación de la oferta.  *
 *
 * Las ofertas que no se presenten recibirán una mayor prioridad de recomendación.
 * Las ofertas con al menos un contacto se ordenarán según la tasa de probabilidad de aceptación descendente.
 *
 * Nota: todos los recuentos se mantienen en memoria. No es un escenario realista, ya que se quedará sin memoria antes o
 * después. En un escenario de producción real, los recuentos deben ser persistentes en una base de datos.
 *
 */

public class SampleLearning implements ILearning
{

    // Una correlación de ID de oferta con recuentos de contactos para el ID de oferta
    private Map<Long,Integer> _offerToContactCount = new HashMap<Long, Integer>();

    // Una correlación de ID de oferta con recuentos de contactos para el ID de oferta
    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
    {
        // Si se necesitan conexiones remotas, es un buen lugar para inicializarlas ya que este
        // método se invoca una vez al inicio de la aplicación web de tiempo de ejecución de Interact.
        // Este ejemplo no tiene conexiones remotas e imprime a efectos de depuración que este método
        // se invocará
        System.out.println("Calling initialize for SampleLearning");
    }

    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#reinitialize
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void reinitialize(ILearningConfig config, boolean debug) throws LearningException
    {
        // Si se despliega un IC, se invoca este método de reinicialización para que la implementación
        // renueve los valores de configuración actualizados
        System.out.println("Calling reinitialize for SampleLearning");
    }


    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#logEvent
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningContext,
```

```
 *   com.unicacorp.interact.treatment.optimization.v2.IOffer,
 *   com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
 *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
 */
public void logEvent(ILearningContext context, IOffer offer, IClientArgs clientArgs,
  IInteractSession session, boolean debug) throws LearningException
{
    System.out.println("Calling logEvent for SampleLearning");


    if(context.getLearningContext()==ILearningContext.LOG_AS_CONTACT)
    {
        System.out.println("adding contact");

        // Realizar un seguimiento de todos los contactos en la memoria
        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");
        // Realizar un seguimiento de todos los recuentos de aceptación en la memoria mediante la adición a la correlación
        synchronized(_offerToAcceptCount)
        {
           Integer count = _offerToAcceptCount.get(offer.getOfferId());
           if(count == null)
               count = new Integer(1);
           else
               count++;
           _offerToAcceptCount.put(offer.getOfferId(), ++count);
        }
        }

}

/* (non-Javadoc)
 * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#optimizeRecommendList
 *   (java.util.List, com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
 *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
 */
public List<ITreatment> optimizeRecommendList(List<ITreatment> recList,
    IClientArgs clientArgs, IInteractSession session, boolean debug)
    throws LearningException
{
    System.out.println("Calling optimizeRecommendList for SampleLearning");

    // Ordenar los tratamientos candidatos invocando el clasificador definido en esta clase y devolver la lista ordenada
    Collections.sort(recList,new MyOfferSorter());

    // ahora devolver sólo lo que se ha solicitado mediante la variable "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;
}

/* (non-Javadoc)
 * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#shutdown
 *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
 */
public void shutdown(ILearningConfig config, boolean debug) throws LearningException
{
    // Si existen conexiones remotas, es un buen lugar para desconectarse ordenadamente
    // de ellas ya que este método se invoca en la conclusión de la aplicación web de tiempo de ejecución de
    // Interact. Para este ejemplo, no hay nada que hacer
    // excepto imprimir una sentencia de depuración.
    System.out.println("Calling shutdown for SampleLearning");
```

```
    }
    // Ordenar por:
    // 1. ofertas con cero contactos - para los vínculos, el orden se basa en la entrada original
    // 2. tasa de probabilidad de aceptación descendente - para los vínculos, el orden se basa en la entrada original

    public class MyOfferSorter implements Comparator<ITreatment>
    {
        private static final long serialVersionUID = 1L;

        /* (non-Javadoc)
         * @see java.lang.Comparable#compareTo(java.lang.Object)
         */
        public int compare(ITreatment treatment1, ITreatment treatment2)
        {

            // obtener el recuento de contactos para ambos tratamientos
            Integer contactCount1 = _offerToContactCount.get(treatment1.getOffer().getOfferId());
            Integer contactCount2 = _offerToContactCount.get(treatment2.getOffer().getOfferId());

            // si no se ha puesto en contacto con el tratamiento, será el ganador
            if(contactCount1 == null || contactCount1 == 0)
                return -1;

            if(contactCount2 == null || contactCount2 == 0)
                return 1;

            // obtener los recuentos de aceptación
            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;

            // orden descendente
            return (int) (acceptProbability2 - acceptProbability1);

        }
    }

}
```

Apéndice A. IBM Interact WSDL

La instalación de Interact incluye dos archivos XML de WDSL (Web Services Description Language) que describen los servicios web disponibles y cómo acceder a ellos. Puede visualizar estos archivos en el directorio de inicio de Interact y en un ejemplo que se muestra aquí.

Cuando haya instalado Interact, podrá encontrar los archivos WSDL de Interact en la siguiente ubicación:

- <*Inicio_Interact*>/conf/InteractService.wsdl
- <*Inicio_Interact*>/conf/InteractAdminService.wsdl

Con cada release o fixpack de software, podría haber cambios en el WSDL de Interact. Para obtener más información, consulte las *Notas del release de Interact* o los archivos readme del release.

Aquí se muestra una copia de `InteractService.wsdl` para su consulta. Para asegurarse de que está utilizando la información más reciente, consulte los archivos WDSL instalados con 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>
```

Apéndice B. Propiedades de configuración del entorno de ejecución de Interact

Esta sección describe todas las propiedades de configuración del entorno de ejecución de Interact.

Interact | general

Estas propiedades de configuración definen los valores generales del entorno de ejecución, incluidos el nivel de registro predeterminado y el valor de entorno local.

log4jConfig

Descripción

La ubicación del archivo que contiene las propiedades log4j. Esta ruta debe ser relativa a la variable de entorno `INTERACT_HOME`. `INTERACT_HOME` es la ubicación del directorio de instalación de Interact.

Valor predeterminado

`./conf/interact_log4j.properties`

asmUserForDefaultLocale

Descripción

La propiedad `asmUserForDefaultLocale` define el usuario de IBM EMM del que Interact deriva sus valores de entorno local.

Los valores de entorno local definen qué idioma aparece en el tiempo de diseño y en qué idioma están los mensajes de aviso de la API de Interact. Si el valor de entorno local no coincide con los valores del sistema operativo de las máquinas, Interact todavía funciona, aunque la pantalla del tiempo de diseño y los mensajes de aviso pueden aparecer en otro idioma.

Valor predeterminado

`asm_admin`

Interact | general | learningTablesDataSource

Estas propiedades de configuración definen la configuración del origen de datos para las tablas de aprendizaje incorporadas. Debe definir este origen de datos si utiliza el aprendizaje incorporado de Interact.

Si crea su propia implementación de aprendizaje utilizando la API de aprendizaje, puede configurar su implementación de aprendizaje personalizada para leer estos valores utilizando la interfaz `ILearningConfig`.

jndiName

Descripción

Utilice esta propiedad `jndiName` para identificar el origen de datos JNDI (Java Naming and Directory Interface) que está definido en el servidor de

aplicaciones (WebSphere o WebLogic) para las tablas de aprendizaje a las que acceden los servidores de ejecución de Interact.

Las tablas de aprendizaje se crean utilizando el archivo ddl `aci_lrntab` y contienen las siguientes tablas (entre otras): `UACI_AttributeValue` y `UACI_OfferStats`.

Valor predeterminado

No se ha definido ningún valor predeterminado.

type

Descripción

El tipo de base de datos para el origen de datos utilizado por las tablas de aprendizaje a las que acceden los servidores de ejecución de Interact.

Las tablas de aprendizaje se crean utilizando el archivo ddl `aci_lrntab` y contienen las siguientes tablas (entre otras): `UACI_AttributeValue` y `UACI_OfferStats`.

Valor predeterminado

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descripción

La propiedad `ConnectionRetryPeriod` especifica la cantidad de tiempo en segundos que Interact reintenta automáticamente la solicitud de conexión de base de datos anómala para las tablas de aprendizaje. Interact intenta reconectarse automáticamente a la base de datos durante este periodo de tiempo antes de informar de un error o una anomalía de base de datos. Si el valor se establece en 0, Interact intentará reconectarse indefinidamente; si el valor se establece en -1, no se realizará ningún intento.

Las tablas de aprendizaje se crean utilizando el archivo ddl `aci_lrntab` y contienen las siguientes tablas (entre otras): `UACI_AttributeValue` y `UACI_OfferStats`.

Valor predeterminado

-1

connectionRetryDelay

Descripción

La propiedad `ConnectionRetryDelay` especifica la cantidad de tiempo en segundos que Interact espera antes de intentar reconectarse a la base de datos después de una anomalía para las tablas de aprendizaje. Si el valor se establece en -1, no realiza ningún intento.

Las tablas de aprendizaje se crean utilizando el archivo ddl `aci_lrntab` y contienen las siguientes tablas (entre otras): `UACI_AttributeValue` y `UACI_OfferStats`.

Valor predeterminado

-1

esquema

Descripción

El nombre del esquema que contiene las tablas para el módulo de aprendizaje incorporado. Interact inserta el valor de esta propiedad antes de todos los nombres de tabla, por ejemplo, `UACI_IntChannel` pasa a ser `schema.UACI_IntChannel`.

No tiene que definir un esquema. Si no define un esquema, Interact supone que el propietario de las tablas es el mismo que el del esquema. Debe establecer este valor para eliminar ambigüedades.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Interact | general | prodUserDataSource

Estas propiedades de configuración definen la configuración del origen de datos para las tablas de perfiles de producción. Debe definir este origen de datos. Es el origen de datos al que hace referencia el entorno de ejecución cuando ejecuta diagramas de flujo interactivos después del despliegue.

jndiName

Descripción

Utilice esta propiedad `jndiName` para identificar el origen de datos JNDI (Java Naming and Directory Interface) que está definido en el servidor de aplicaciones (WebSphere o WebLogic) para las tablas de clientes a las que acceden los servidores de ejecución de Interact.

Valor predeterminado

No se ha definido ningún valor predeterminado.

type

Descripción

El tipo de base de datos para las tablas de clientes a las que acceden los servidores de ejecución de Interact.

Valor predeterminado

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

aliasPrefix

Descripción

La propiedad `AliasPrefix` especifica la manera en que Interact forma el nombre de alias que Interact crea automáticamente cuando utiliza una tabla de dimensiones y escribe en una nueva tabla en las tablas de clientes a las que acceden los servidores de ejecución de Interact.

Tenga en cuenta que cada base de datos tiene una longitud de identificador máxima; revise la documentación para la base de datos que está utilizando para asegurarse de que el valor que ha establecido no excede la longitud de identificador máxima para su base de datos.

Valor predeterminado

A

connectionRetryPeriod

Descripción

La propiedad `ConnectionRetryPeriod` especifica la cantidad de tiempo en segundos que Interact reintenta automáticamente la solicitud de conexión de base de datos anómala para las tablas de clientes de tiempo de ejecución. Interact intenta reconectarse automáticamente a la base de datos durante este periodo de tiempo antes de informar de un error o una anomalía de base de datos. Si el valor se establece en 0, Interact intentará reconectarse indefinidamente; si el valor se establece en -1, no se realizará ningún intento.

Valor predeterminado

-1

connectionRetryDelay

Descripción

La propiedad `ConnectionRetryDelay` especifica la cantidad de tiempo en segundos que Interact espera antes de intentar reconectarse a la base de datos después de una anomalía para las tablas de clientes de tiempo de ejecución de Interact. Si el valor se establece en -1, no realiza ningún intento.

Valor predeterminado

-1

esquema

Descripción

El nombre del esquema que contiene las tablas de datos de perfil. Interact inserta el valor de esta propiedad antes de todos los nombres de tabla, por ejemplo, `UACI_IntChannel` pasa a ser `schema.UACI_IntChannel`.

No tiene que definir un esquema. Si no define un esquema, Interact supone que el propietario de las tablas es el mismo que el del esquema. Debe establecer este valor para eliminar ambigüedades.

Cuando utiliza una base de datos DB2, el nombre de esquema debe estar en mayúsculas.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Interact | general | systemTablesDataSource

Estas propiedades de configuración definen la configuración del origen de datos para las tablas del sistema del entorno de ejecución. Debe definir este origen de datos.

jndiName

Descripción

Utilice esta propiedad jndiName para identificar el origen de datos JNDI (Java Naming and Directory Interface) que está definido en el servidor de aplicaciones (WebSphere o WebLogic) para las tablas del entorno de ejecución.

La base de datos del entorno de ejecución es la base de datos que se completa con los scripts dll aci_runtime y aci_populate_runtime y, por ejemplo, contiene las siguientes tablas (entre otras): UACI_CHOfferAttrib y UACI_DefaultedStat.

Valor predeterminado

No se ha definido ningún valor predeterminado.

type

Descripción

El tipo de base de datos para las tablas del sistema del entorno de ejecución.

La base de datos del entorno de ejecución es la base de datos que se completa con los scripts dll aci_runtime y aci_populate_runtime y, por ejemplo, contiene las siguientes tablas (entre otras): UACI_CHOfferAttrib y UACI_DefaultedStat.

Valor predeterminado

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descripción

La propiedad ConnectionRetryPeriod especifica la cantidad de tiempo en segundos que Interact reintenta automáticamente la solicitud de conexión de base de datos anómala para las tablas del sistema de tiempo de ejecución. Interact intenta reconectarse automáticamente a la base de datos durante este periodo de tiempo antes de informar de un error o una anomalía de base de datos. Si el valor se establece en 0, Interact intentará reconectarse indefinidamente; si el valor se establece en -1, no se realizará ningún intento.

La base de datos del entorno de ejecución es la base de datos que se completa con los scripts dll aci_runtime y aci_populate_runtime y, por ejemplo, contiene las siguientes tablas (entre otras): UACI_CHOfferAttrib y UACI_DefaultedStat.

Valor predeterminado

-1

connectionRetryDelay

Descripción

La propiedad ConnectionRetryDelay especifica la cantidad de tiempo en segundos que Interact espera antes de intentar reconectarse a la base de

datos después de una anomalía para las tablas del sistema de tiempo de ejecución de Interact. Si el valor se establece en -1, no realiza ningún intento.

La base de datos del entorno de ejecución es la base de datos que se completa con los scripts dll `aci_runtime` y `aci_populate_runtime` y, por ejemplo, contiene las siguientes tablas (entre otras): `UACI_CHOfferAttrib` y `UACI_DefaultedStat`.

Valor predeterminado

-1

esquema

Descripción

El nombre del esquema que contiene las tablas para el entorno de ejecución. Interact inserta el valor de esta propiedad antes de todos los nombres de tabla, por ejemplo, `UACI_IntChannel` pasa a ser `schema.UACI_IntChannel`.

No tiene que definir un esquema. Si no define un esquema, Interact supone que el propietario de las tablas es el mismo que el del esquema. Debe establecer este valor para eliminar ambigüedades.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Interact | general | systemTablesDataSource | loaderProperties

Estas propiedades de configuración definen la configuración de una utilidad de carga de base de datos para las tablas del sistema del entorno de ejecución. Sólo debe definir estas propiedades si utiliza una utilidad de carga de base de datos.

databaseName

Descripción

El nombre de la base de datos a la que se conecta el cargador de base de datos.

Valor predeterminado

No se ha definido ningún valor predeterminado.

LoaderCommandForAppend

Descripción

El parámetro `LoaderCommandForAppend` especifica el comando emitido para invocar la utilidad de carga de base de datos para añadir registros a las tablas de base de datos de preparación del historial de contactos y respuestas en Interact. Debe establecer este parámetro para habilitar la utilidad de carga de base de datos para los datos del historial de contactos y respuestas.

Este parámetro se especifica como un nombre de ruta completo al ejecutable de la utilidad de carga de base de datos o a un script que inicia la utilidad de carga de base de datos. La utilización de un script permite realizar configuraciones adicionales antes de invocar la utilidad de carga.

La mayoría de las utilidades de carga de base de datos requieren varios argumentos para poder iniciarse satisfactoriamente. Por ejemplo, la

especificación del archivo de datos y el archivo de control desde los que se carga, y la base de datos y la tabla donde se carga. Los tokens se sustituyen por los elementos especificados cuando se ejecuta el comando.

Consulte la documentación de la utilidad de carga de base de datos para conocer la sintaxis correcta que se debe utilizar cuando se invoca la utilidad de carga de base de datos.

Este parámetro no está definido de forma predeterminada.

Los tokens disponibles para `LoaderCommandForAppend` se describen en la siguiente tabla.

Token	**Descripción**
<CONTROLFILE>	Este token se sustituye por la ruta completa y el nombre de archivo del archivo de control temporal que Interact genera de acuerdo con la plantilla que se especifica en el parámetro `LoaderControlFileTemplate`.
<DATABASE>	Este token se sustituye por el nombre del origen de datos en el que Interact está cargando datos. Es el mismo nombre de origen de datos que se utiliza en el nombre de categoría para este origen de datos.
<DATAFILE>	Este token se sustituye por la ruta completa y el nombre de archivo del archivo de datos temporal creado por Interact durante el proceso de carga. Este archivo se encuentra en el directorio temporal de Interact, `UNICA_ACTMPDIR`.
<DBCOLUMNNUMBER>	Este token se sustituye por el ordinal de columna en la base de datos.
<FIELDLENGTH>	Este token se sustituye por la longitud del campo que se está cargando en la base de datos.
<FIELDNAME>	Este token se sustituye por el nombre del campo que se está cargando en la base de datos.
<FIELDNUMBER>	Este token se sustituye por el número del campo que se está cargando en la base de datos.
<FIELDTYPE>	Este token se sustituye por el literal "CHAR( )". La longitud de este campo se especifica entre paréntesis (). Si la base de datos no entiende el tipo de campo, CHAR, puede especificar manualmente el texto correspondiente para el tipo de campo y utilizar el token <FIELDLENGTH>. Por ejemplo, para SQLSVR y SQL2000, puede utilizar "SQLCHAR(<FIELDLENGTH>)".

Token	Descripción
<NATIVETYPE>	Este token se sustituye por el tipo de base de datos en el que se carga este campo.
<NUMFIELDS>	Este token se sustituye por el número de campos en la tabla.
<PASSWORD>	Este token se sustituye por la contraseña de base de datos de la conexión de diagrama de flujo actual con el origen de datos.
<TABLENAME>	Este token se sustituye por el nombre de tabla de base de datos en el que Interact está cargando datos.
<USER>	Este token se sustituye por el usuario de base de datos de la conexión de diagrama de flujo actual con el origen de datos.

Valor predeterminado

No se ha definido ningún valor predeterminado.

LoaderControlFileTemplateForAppend

Descripción

La propiedad `LoaderControlFileTemplateForAppend` especifica la ruta completa y el nombre de archivo de la plantilla de archivo de control que se ha configurado previamente en Interact. Cuando se establece este parámetro, Interact crea dinámicamente un archivo de control temporal basado en la plantilla que se especifica aquí. La ruta y el nombre de este archivo de control temporal están disponibles en el token `<CONTROLFILE>` que está disponible en la propiedad `LoaderCommandForAppend`.

Antes de utilizar Interact en modo de utilidad de carga de la base de datos, debe configurar la plantilla de archivo de control que ha especificado este parámetro. La plantilla de archivo de control da soporte a los siguientes tokens, que se sustituyen dinámicamente cuando Interact crea el archivo de control temporal.

Consulte la documentación de la utilidad de carga de base de datos para conocer la sintaxis correcta que requiere el archivo de control. Los tokens disponibles para su plantilla de archivo de control son los mismos que los de la propiedad `LoaderControlFileTemplate`.

Este parámetro no está definido de forma predeterminada.

Valor predeterminado

No se ha definido ningún valor predeterminado.

LoaderDelimiterForAppend

Descripción

La propiedad `LoaderDelimiterForAppend` especifica si el archivo de datos temporal de Interact es un archivo sin formato con delimitadores o un

archivo de ancho fijo y, si está delimitado, el carácter o el juego de caracteres utilizado como delimitadores.

Si el valor no está definido, Interact crea el archivo de datos temporal como un archivo sin formato de ancho fijo.

Si especifica un valor, se utilizará cuando se invoque al cargador para completar una tabla que no se sabe que está vacía. Interact crea el archivo de datos temporal como un archivo sin formato con delimitadores, utilizando el valor de esta propiedad como delimitador.

Esta propiedad no está definida de forma predeterminada.

Valor predeterminado

Valores válidos

Caracteres, que puede escribir entre comillas dobles, si lo desea.

LoaderDelimiterAtEndForAppend

Descripción

Algunas utilidades de carga externas requieren que el archivo de datos esté delimitado y que cada línea finalice con el delimitador. Para satisfacer este requisito, establezca el valor de `LoaderDelimiterAtEndForAppend` en `TRUE`, para que cuando se invoque el cargador para completar una tabla que no se sabe que está vacía, Interact utilice delimitadores al final de cada línea.

Valor predeterminado

`FALSE`

Valores válidos

`TRUE | FALSE`

LoaderUseLocaleDP

Descripción

La propiedad `LoaderUseLocaleDP` especifica, cuando Interact escribe valores numéricos en archivos que va a cargar una utilidad de carga de base de datos, si se utiliza el símbolo específico del entorno local para el separador decimal.

Establezca este valor en `FALSE` para especificar que se utilice el punto (.) como separador decimal.

Establezca este valor en `TRUE` para especificar que se utilice el símbolo de separador decimal correspondiente al entorno local.

Valor predeterminado

`FALSE`

Valores válidos

`TRUE | FALSE`

Interact | general | testRunDataSource

Estas propiedades de configuración definen la configuración del origen de datos para las tablas de ejecución de prueba del entorno de diseño de Interact. Debe

definir este origen de datos para al menos uno de los entornos de ejecución. Estas son las tablas que se utilizan cuando realiza una ejecución de prueba del diagrama de flujo interactivo.

jndiName

Descripción

Utilice esta propiedad `jndiName` para identificar el origen de datos JNDI (Java Naming and Directory Interface) que está definido en el servidor de aplicaciones (WebSphere o WebLogic) para las tablas de clientes a las que accede el entorno de diseño cuando ejecuta las ejecuciones de prueba de diagramas de flujo interactivos.

Valor predeterminado

No se ha definido ningún valor predeterminado.

type

Descripción

El tipo de base de datos para las tablas de clientes a las que accede el entorno de diseño cuando ejecuta las ejecuciones de prueba de diagramas de flujo interactivos.

Valor predeterminado

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

aliasPrefix

Descripción

La propiedad `AliasPrefix` especifica la manera en que Interact forma el nombre de alias que Interact crea automáticamente cuando utiliza una tabla de dimensiones y escribe en una nueva tabla para las tablas de clientes a las que accede el entorno de diseño cuando ejecuta las ejecuciones de prueba de diagramas de flujo interactivos.

Tenga en cuenta que cada base de datos tiene una longitud de identificador máxima; revise la documentación para la base de datos que está utilizando para asegurarse de que el valor que ha establecido no excede la longitud de identificador máxima para su base de datos.

Valor predeterminado

A

connectionRetryPeriod

Descripción

La propiedad `ConnectionRetryPeriod` especifica la cantidad de tiempo en segundos que Interact reintenta automáticamente la solicitud de conexión de base de datos anómala para las tablas de ejecución de prueba. Interact intenta reconectarse automáticamente a la base de datos durante este periodo de tiempo antes de informar de un error o una anomalía de base

de datos. Si el valor se establece en 0, Interact intentará reconectarse indefinidamente; si el valor se establece en -1, no se realizará ningún intento.

Valor predeterminado

-1

connectionRetryDelay

Descripción

La propiedad `ConnectionRetryDelay` especifica la cantidad de tiempo en segundos que Interact espera antes de intentar reconectarse a la base de datos después de una anomalía para las tablas de ejecución de prueba. Si el valor se establece en -1, no realiza ningún intento.

Valor predeterminado

-1

esquema

Descripción

El nombre del esquema que contiene las tablas para las ejecuciones de prueba del diagrama de flujo interactivo. Interact inserta el valor de esta propiedad antes de todos los nombres de tabla, por ejemplo, `UACI_IntChannel` pasa a ser `schema.UACI_IntChannel`.

No tiene que definir un esquema. Si no define un esquema, Interact supone que el propietario de las tablas es el mismo que el del esquema. Debe establecer este valor para eliminar ambigüedades.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Interact | general | contactAndResponseHistoryDataSource

Estas propiedades de configuración definen los valores de conexión para el origen de datos del historial de respuestas y contactos necesario para el seguimiento de respuestas de sesiones cruzadas de Interact. Estos valores no están relacionados con el módulo de historial de contactos y respuestas.

jndiName

Descripción

Utilice esta propiedad `jndiName` para identificar el origen de datos JNDI (Java Naming and Directory Interface) que está definido en el servidor de aplicaciones (WebSphere o WebLogic) para el origen de datos del historial de respuestas y contactos necesario para el seguimiento de respuestas de sesiones cruzadas de Interact.

Valor predeterminado

type

Descripción

El tipo de base de datos para el origen de datos utilizado por el origen de datos del historial de respuestas y contactos necesario para el seguimiento de respuestas de sesiones cruzadas de Interact.

Valor predeterminado

SQLServer

Valores válidos

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descripción

La propiedad `ConnectionRetryPeriod` especifica la cantidad de tiempo en segundos que Interact reintenta automáticamente la solicitud de conexión de base de datos anómala para el seguimiento de respuestas de sesiones cruzadas de Interact.Interact intenta reconectarse automáticamente a la base de datos durante este periodo de tiempo antes de informar de un error o una anomalía de base de datos. Si el valor se establece en 0, Interact intentará reconectarse indefinidamente; si el valor se establece en -1, no se realizará ningún intento.

Valor predeterminado

-1

connectionRetryDelay

Descripción

La propiedad `ConnectionRetryDelay` especifica la cantidad de tiempo en segundos que Interact espera antes de intentar reconectarse a la base de datos después de una anomalía para el seguimiento de respuestas de sesiones cruzadas Interact. Si el valor se establece en -1, no realiza ningún intento.

Valor predeterminado

-1

schema

Descripción

El nombre del esquema que contiene las tablas para el seguimiento de respuestas de sesiones cruzadas de Interact. Interact inserta el valor de esta propiedad antes de todos los nombres de tabla, por ejemplo, `UACI_IntChannel` pasa a ser `schema.UACI_IntChannel`.

No tiene que definir un esquema. Si no define un esquema, Interact supone que el propietario de las tablas es el mismo que el del esquema. Debe establecer este valor para eliminar ambigüedades.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Interact | general | idsByType

Estas propiedades de configuración definen los valores de los números de ID utilizados por el módulo de historial de contactos y respuestas.

initialValue

Descripción

El valor de ID inicial utilizado cuando se generan ID utilizando la tabla UACI_IDsByType.

Valor predeterminado

1

Valores válidos

Un valor mayor que 0.

retries

Descripción

El número de reintentos antes de producir una excepción cuando se generan ID utilizando la tabla UACI_IDsByType.

Valor predeterminado

20

Valores válidos

Un entero mayor que 0.

Interact | flowchart

En esta sección se definen los valores de configuración de los diagramas de flujo interactivos.

defaultDateFormat

Descripción

El formato de fecha predeterminado utilizado por Interact para convertir la fecha en una cadena de caracteres y la cadena de caracteres en una fecha.

Valor predeterminado

MM/dd/yy

idleFlowchartThreadTimeoutInMinutes

Descripción

El número de minutos que Interact permite que un subproceso dedicado a un diagrama de flujo interactivo esté desocupado antes de liberar el subproceso.

Valor predeterminado

5

idleProcessBoxThreadTimeoutInMinutes

Descripción

El número de minutos que Interact permite que un subproceso dedicado a un proceso de diagrama de flujo interactivo esté desocupado antes de liberar el subproceso.

Valor predeterminado

5

maxSizeOfFlowchartEngineInboundQueue

Descripción

El número máximo de solicitudes de ejecución de diagrama de flujo que Interact mantiene en cola. Si se alcanza este número de solicitudes, Interact dejará de aceptar solicitudes.

Valor predeterminado

1000

maxNumberOfFlowchartThreads

Descripción

El número máximo de subprocesos dedicados a solicitudes de diagramas de flujo interactivos.

Valor predeterminado

25

maxNumberOfProcessBoxThreads

Descripción

El número máximo de subprocesos dedicados a procesos de diagramas de flujo interactivos.

Valor predeterminado

50

maxNumberOfProcessBoxThreadsPerFlowchart

Descripción

El número máximo de subprocesos dedicados a procesos de diagramas de flujo interactivos por instancia de diagrama de flujo.

Valor predeterminado

3

minNumberOfFlowchartThreads

Descripción

El número mínimo de subprocesos dedicados a solicitudes de diagramas de flujo interactivos.

Valor predeterminado

10

minNumberOfProcessBoxThreads

Descripción

El número mínimo de subprocesos dedicados a procesos de diagramas de flujo interactivos.

Valor predeterminado

20

sessionVarPrefix

Descripción

El prefijo de las variables de sesión.

Valor predeterminado

SessionVar

Interact | flowchart | ExternalCallouts | [ExternalCalloutName]

En esta sección se definen los valores de clase para las llamadas externas personalizadas que ha escrito con la API de llamadas externas.

class

Descripción

El nombre de la clase Java representada por esta llamada externa.

Es la clase Java a la que puede acceder con la macro de IBM `EXTERNALCALLOUT`.

Valor predeterminado

No se ha definido ningún valor predeterminado.

classpath

Descripción

La ruta de clases de la clase Java representada por esta llamada externa. La ruta de clases debe hacer referencia a los archivos jar en el servidor del entorno de ejecución. Si utiliza un grupo de servidores y todos los servidores de ejecución utilizan la misma Marketing Platform, cada servidor debe tener una copia del archivo jar en la misma ubicación. La ruta de clases debe estar formada por las ubicaciones absolutas de los archivos jar, separadas por el delimitador de ruta del sistema operativo del servidor del entorno de ejecución, por ejemplo, un punto y coma (;) en Windows y dos puntos (:) en los sistemas UNIX. Los directorios que contienen archivos de clase no se aceptan. Por ejemplo, en un sistema UNIX: `/path1/file1.jar:/path2/file2.jar`.

Esta ruta de clases debe tener menos de 1024 caracteres. Puede utilizar el archivo de manifiesto en un archivo .jar para especificar otros archivos .jar para que sólo tenga que aparecer un archivo .jar en su ruta de clases.

Es la clase Java a la que puede acceder con la macro de IBM `EXTERNALCALLOUT`.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Interact | flowchart | ExternalCallouts | [ExternalCalloutName] | Parameter Data | [parameterName]

En esta sección se definen los valores de parámetro para una llamadas externa personalizada que ha escrito con la API de llamadas externas.

value

Descripción

El valor de un parámetro necesario para la clase para la llamada externa.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Ejemplo

Si la llamada externa requiere el nombre de host de un servidor externo, debe crear una categoría de parámetro denominada `host` y definir la propiedad `value` como el nombre del servidor.

Interact | monitoring

Este conjunto de propiedades de configuración permite definir los valores de supervisión JMX. Sólo debe configurar estas propiedades si está utilizando la supervisión JMX. Existen otras propiedades de supervisión JMX que se deben definir para el módulo del historial de contactos y respuestas en las propiedades de configuración del entorno de diseño de Interact.

protocol

Descripción

Define el protocolo del servicio de mensajería de Interact.

Si elige JMXMP, debe incluir los siguientes archivos JAR en su ruta de clases en orden:

`Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar`

Valor predeterminado

JMXMP

Valores válidos

JMXMP | RMI

port

Descripción

El número de puerto del servicio de mensajería.

Valor predeterminado

9998

enableSecurity

Descripción

Un booleano que habilita o inhabilita la seguridad del servicio de mensajería JMXMP para el servidor de ejecución de Interact. Si se establece en `true`, debe proporcionar un nombre de usuario y una contraseña para acceder al servicio JMX de tiempo de ejecución de Interact. Esta credencial de usuario se autentica mediante la Marketing Platform del servidor de ejecución. Jconsole no permite el inicio de sesión con contraseña vacía.

Esta propiedad no tiene ningún efecto si el protocolo es RMI. Esta propiedad no tiene ningún efecto en JMX para Campaign (el tiempo de diseño de Interact).

Valor predeterminado

True

Valores válidos

True | False

Interact | profile

Este conjunto de propiedades de configuración controla varias de las características de presentación de ofertas opcionales, incluida la supresión de ofertas y la anulación de puntuaciones.

enableScoreOverrideLookup

Descripción

Si se establece en `True`, Interact carga los datos de anulación de puntuaciones desde `scoreOverrideTable` al crear una sesión. Si se establece en `False`, Interact no carga los datos de anulación de puntuaciones de marketing cuando se crea una sesión.

Si es true, también debe configurar la propiedad `IBM EMM > Interact > profile > Audience Levels > (Audience Level) > scoreOverrideTable`. Sólo debe definir la propiedad `scoreOverrideTable` para los niveles de audiencia que necesite. Si deja `scoreOverrideTable` en blanco para un nivel de audiencia, se inhabilita la tabla de anulación de puntuaciones para el nivel de audiencia.

Valor predeterminado

`False`

Valores válidos

True | False

enableOfferSuppressionLookup

Descripción

Si se establece en `True`, Interact carga los datos de supresión de ofertas desde `offerSuppressionTable` al crear una sesión. Si se establece en `False`, Interact no carga los datos de supresión de ofertas cuando se crea una sesión.

Si es true, también debe configurar la propiedad `IBM EMM > Interact > profile > Audience Levels > (Audience Level) > offerSuppressionTable`. Sólo debe definir la propiedad `enableOfferSuppressionLookup` para los niveles de audiencia que necesite.

Valor predeterminado

`False`

Valores válidos

True | False

enableProfileLookup

Descripción

En una nueva instalación de Interact, esta propiedad está en desuso. En una instalación actualizada de Interact, esta propiedad es válida hasta el primer despliegue.

El comportamiento de carga para una tabla utilizada en un diagrama de flujo interactivo pero no correlacionada en el canal interactivo. Si se establece en `True`, Interact carga los datos del perfil desde `profileTable` al crear una sesión.

Si es true, también debe configurar la propiedad `IBM EMM > Interact > profile > Audience Levels > (Audience Level) > profileTable`.

El valor **Cargar estos datos en la memoria cuando comience una sesión de visita** en el asistente de correlación de tablas de canal interactivo altera temporalmente esta propiedad de configuración.

Valor predeterminado

`False`

Valores válidos

True | False

defaultOfferUpdatePollPeriod

Descripción

El número de segundos que espera el sistema antes de actualizar las ofertas predeterminadas en la memoria caché desde la tabla de ofertas predeterminadas. Si se establece en -1, el sistema no actualiza las ofertas predeterminadas en la memoria caché después de cargar la lista de valores de inicio en la memoria caché cuando se inicia el servidor de ejecución.

Valor predeterminado

-1

Interact | profile | Audience Levels | [AudienceLevelName]

Este conjunto de propiedades de configuración permite definir los nombres de tabla necesarios para las características adicionales de Interact. Sólo tiene que definir el nombre de tabla si está utilizando la característica asociada.

scoreOverrideTable

Descripción

El nombre de la tabla que contiene la información de anulación de puntuaciones para este nivel de audiencia. Esta propiedad sólo es aplicable si establece `enableScoreOverrideLookup` en true. Debe definir esta propiedad para los niveles de audiencia para los que desea habilitar una tabla de anulación de puntuaciones. Si no tiene una tabla de anulación de puntuaciones para este nivel de audiencia, puede dejar esta propiedad sin definir, aunque `enableScoreOverrideLookup` se haya establecido en true.

Interact busca esta tabla en las tablas del cliente a las que acceden los servidores de ejecución de Interact, definidos por las propiedades `prodUserDataSource`.

Si ha definido la propiedad `schema` para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo,

schema.UACI_ScoreOverride. Si especifica un nombre completo, por ejemplo, mySchema.UACI_ScoreOverride, Interact no añade el nombre de esquema.

Valor predeterminado

UACI_ScoreOverride

offerSuppressionTable

Descripción

El nombre de la tabla que contiene la información de supresión de ofertas para este nivel de audiencia. Debe definir esta propiedad para los niveles de audiencia para los que desea habilitar una tabla de supresión de ofertas. Si no tiene una tabla de supresión de ofertas para este nivel de audiencia, puede dejar esta propiedad sin definir. Si enableOfferSuppressionLookup se establece en true, esta propiedad debe establecerse en una tabla válida.

Interact busca esta tabla en las tablas del cliente a las que acceden los servidores de ejecución, definidos por las propiedades prodUserDataSource.

Valor predeterminado

UACI_BlackList

profileTable

Descripción

En una nueva instalación de Interact, esta propiedad está en desuso. En una instalación actualizada de Interact, esta propiedad es válida hasta el primer despliegue.

El nombre de la tabla que contiene los datos del perfil para este nivel de audiencia.

Interact busca esta tabla en las tablas del cliente a las que acceden los servidores de ejecución, definidos por las propiedades prodUserDataSource.

Si ha definido la propiedad schema para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo, schema.UACI_usrProd. Si especifica un nombre completo, por ejemplo, mySchema.UACI_usrProd, Interact no añade el nombre de esquema.

Valor predeterminado

No se ha definido ningún valor predeterminado.

contactHistoryTable

Descripción

El nombre de la tabla de preparación de los datos del historial de contactos para este nivel de audiencia.

Esta tabla se almacena en las tablas del entorno de ejecución (systemTablesDataSource).

Si ha definido la propiedad schema para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo, schema.UACI_CHStaging. Si especifica un nombre completo, por ejemplo, mySchema.UACI_CHStaging, Interact no añade el nombre de esquema.

Si el registro del historial de contactos está inhabilitado, no es necesario establecer esta propiedad.

Valor predeterminado

UACI_CHStaging

chOfferAttribTable

Descripción

El nombre de la tabla de atributos de oferta del historial de contactos para este nivel de audiencia.

Esta tabla se almacena en las tablas del entorno de ejecución (`systemTablesDataSource`).

Si ha definido la propiedad `schema` para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo, `schema.UACI_CHOfferAttrib`. Si especifica un nombre completo, por ejemplo, `mySchema.UACI_CHOfferAttrib`, Interact no añade el nombre de esquema.

Si el registro del historial de contactos está inhabilitado, no es necesario establecer esta propiedad.

Valor predeterminado

UACI_CHOfferAttrib

responseHistoryTable

Descripción

El nombre de la tabla de preparación del historial de respuestas para este nivel de audiencia.

Esta tabla se almacena en las tablas del entorno de ejecución (`systemTablesDataSource`).

Si ha definido la propiedad `schema` para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo, `schema.UACI_RHStaging`. Si especifica un nombre completo, por ejemplo, `mySchema.UACI_RHStaging`, Interact no añade el nombre de esquema.

Si el registro del historial de respuestas está inhabilitado, no es necesario establecer esta propiedad.

Valor predeterminado

UACI_RHStaging

crossSessionResponseTable

Descripción

El nombre de la tabla para este nivel de audiencia que se necesita para el seguimiento de respuestas de sesiones cruzadas en las tablas del historial de contactos y respuestas a las que puede acceder la característica de seguimiento de respuestas.

Si ha definido la propiedad `schema` para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo,

schema.UACI_XSessResponse. Si especifica un nombre completo, por ejemplo, mySchema.UACI_XSessResponse, Interact no añade el nombre de esquema.

Si el registro de respuestas de sesiones cruzadas está inhabilitado, no es necesario establecer esta propiedad.

Valor predeterminado

UACI_XSessResponse

userEventLoggingTable

Descripción

Es el nombre de la tabla de base de datos que se utiliza para registrar las actividades de eventos definidos por el usuario. Los eventos definidos por el usuario aparecen en la pestaña Eventos de las páginas del canal interactivo en la interfaz de Interact. La tabla de base de datos que especifique aquí almacenará información tal como el ID de evento, el nombre, las veces que este evento se ha producido para este nivel de audiencia desde la última vez que se ha vaciado la memoria caché de actividad de eventos, etc.

Si ha definido la propiedad schema para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo, schema.UACI_UserEventActivity. Si especifica un nombre completo, por ejemplo, mySchema.UACI_UserEventActivity, Interact no añade el nombre de esquema.

Valor predeterminado

UACI_UserEventActivity

patternStateTable

Descripción

>Es el nombre de la tabla de base de datos que se utiliza para registrar los estados de patrones de eventos, como por ejemplo si se ha cumplido o no la condición del patrón, si el patrón está caducado o inhabilitado, etc.

Si ha definido la propiedad schema para este origen de datos, Interact añade el esquema delante de este nombre de tabla, por ejemplo, schema.UACI_EventPatternState. Si especifica un nombre completo, por ejemplo, mySchema.UACI_EventPatternState, Interact no añade el nombre de esquema.

Se necesita una patternStateTable para cada nivel de audiencia, aunque no utilice patrones de evento. La patternStateTable se basa en el ddl del UACI_EventPatternState incluido. A continuación, se muestra un ejemplo donde el ID de audiencia tiene dos componentes: ComponentNum y 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 predeterminado

UACI_EventPatternState

Interact | profile | Audience Levels | [AudienceLevelName] | Ofertas por SQL sin formato

Este conjunto de propiedades de configuración permite definir los nombres de tabla necesarios para las características adicionales de Interact. Sólo tiene que definir el nombre de tabla si está utilizando la característica asociada.

enableOffersByRawSQL

Descripción

Si se establece en True, Interact habilita la característica offersBySQL para este nivel de audiencia, que permite configurar el código SQL que se debe ejecutar para crear el conjunto deseado de ofertas candidatas en el tiempo de ejecución. Si es False, Interact no utiliza la característica offersBySQL.

Si establece esta propiedad en true, también puede configurar la propiedad Interact | profile | Audience Levels | (Audience Level) | Offers by Raw SQL | SQL Template para definir una o varias plantillas SQL.

Valor predeterminado

False

Valores válidos

True | False

cacheSize

Descripción

Tamaño de la memoria caché utilizada para almacenar los resultados de las consultas de OfferBySQL. Tenga en cuenta que la utilización de una memoria caché puede tener un impacto negativo si los resultados de consulta son exclusivos para la mayoría de las sesiones.

Valor predeterminado

-1 (off)

Valores válidos

-1 | Value

cacheLifeInMinutes

Descripción

Si la memoria caché está habilitada, indica el número de minutos antes de que el sistema borre la memoria caché para evitar la obsolescencia.

Valor predeterminado

-1 (off)

Valores válidos

-1 | Value

defaultSQLTemplate

Descripción

El nombre de la plantilla SQL que se utiliza si no se ha especificado una mediante las llamadas a la API.

Valor predeterminado

`Ninguno`

Valores válidos

Nombre de plantilla SQL

Interact | profile | Audience Levels | [AudienceLevelName] | SQL Template

Estas propiedades de configuración permiten definir una o varias plantillas de consulta SQL utilizadas por la característica offersBySQL de Interact.

name

Descripción

El nombre que desea asignar a esta plantilla de consulta SQL. Escriba un nombre descriptivo que sea significativo cuando utilice esta plantilla SQL en las llamadas a la API. Tenga en cuenta que si utiliza un nombre aquí que es *idéntico* a un nombre definido en el cuadro de proceso Lista de interacción para un tratamiento offerBySQL, se utilizará el SQL del cuadro de proceso en lugar del SQL que especifique aquí.

Valor predeterminado

Ninguno

SQL

Descripción

Contiene la consulta SQL que invocará esta plantilla. La consulta SQL puede contener referencias a nombres de variable que forman parte de los datos de sesión del visitante (perfil). Por ejemplo, `select * from MyOffers where category = ${preferredCategory}` se basará en la sesión que contiene una variable denominada `preferredCategory`.

Debe configurar el SQL para consultar las tablas de oferta específicas que ha creado durante el tiempo de diseño para utilizarlas con esta característica. Tenga en cuenta que los procedimientos almacenados no están soportados aquí.

Valor predeterminado

Ninguno

Interact | profile | Audience Levels | [AudienceLevelName | Profile Data Services | [DataSource]

Este conjunto de propiedades de configuración permite definir los nombres de tabla necesarios para las características adicionales de Interact. Sólo tiene que definir el nombre de tabla si está utilizando la característica asociada. La categoría Profile Data Services proporciona información acerca de un origen de datos incorporado (denominado Database) que se crea para todos los niveles de audiencia, y que está configurado previamente con una prioridad de 100. Pero puede modificar o inhabilitar este valor. Esta categoría también contiene una plantilla para orígenes de datos externos adicionales. Cuando pulsa la plantilla denominada **External Data Services**, puede completar los valores de configuración que se describen aquí.

Nuevo nombre de categoría

Descripción

(No está disponible para la entrada Database predeterminada). Nombre del origen de datos que está definiendo. El nombre que especifique aquí debe ser exclusivo entre los orígenes de datos para un mismo nivel de audiencia.

Valor predeterminado

Ninguno

Valores válidos

Se puede utilizar una cadena de texto cualquiera.

enabled

Descripción

Si el valor es `True`, este origen de datos está habilitado para el nivel de audiencia al que está asignado. Si el valor es `False`, Interact no utiliza este origen de datos para este nivel de audiencia.

Valor predeterminado

`True`

Valores válidos

True | False

className

Descripción

(No está disponible para la entrada Database predeterminada). Nombre completo de la clase de origen de datos que implementa `IInteractProfileDataService`.

Valor predeterminado

Ninguno.

Valores válidos

Nombre de clase completo en forma de cadena de caracteres.

classPath

Descripción

(No está disponible para la entrada Database predeterminada). Valor de configuración opcional que proporciona la ruta para cargar esta clase de implementación del origen de datos. Si no especifica un valor, se utiliza por omisión la ruta de clases del servidor de aplicaciones

Valor predeterminado

No se muestra, pero de forma predeterminada se utiliza la ruta de clases del servidor de aplicaciones si no se proporciona ningún valor aquí.

Valores válidos

Ruta de clases en forma de cadena de caracteres.

priority

Descripción

Prioridad del origen de datos dentro de este nivel de audiencia. Debe ser un valor exclusivo entre todos los orígenes de datos para cada nivel de audiencia. (Es decir, si la prioridad se establece en 100 para un origen de datos, ningún otro origen de datos dentro del nivel de audiencia puede tener una prioridad de 100.)

Valor predeterminado

`100` para el origen de datos predeterminado Database, `200` para un origen de datos definido por el usuario

Valores válidos

Se puede utilizar cualquier número entero no negativo.

Interact | offerserving

Estas propiedades de configuración definen las propiedades de configuración del aprendizaje genérico. Si utiliza el aprendizaje incorporado, para ajustar la implementación de aprendizaje, utilice las propiedades de configuración del entorno de diseño.

offerTieBreakMethod

Descripción

La propiedad `offerTieBreakMethod` define el comportamiento de la presentación de ofertas cuando dos ofertas tienen puntuaciones equivalentes (relacionadas). Si establece esta propiedad en su valor predeterminado, que es Random (Aleatorio), Interact presenta una selección aleatoria entre las ofertas que tienen puntuaciones equivalentes. Si establece esta configuración en Newer Offer (Oferta más reciente),Interact presenta la oferta más reciente)la que tenga el ID de oferta más alto) antes que la oferta más antigua (la del ID de oferta más bajo) en caso de que las dos tengan la misma puntuación.

Nota:

Interact tiene una característica opcional que permite al administrador configurar el sistema de forma que devuelva las ofertas en orden aleatorio independientemente de la puntuación, estableciendo la opción `percentRandomSelection (Campaign | partitions | [partition_number] | Interact | learning | percentRandomSelection)`. La propiedad `offerTieBreakMethod` que se describe aquí se utiliza únicamente cuando `percentRandomSelection` está establecido en cero (inhabilitado).

Valor predeterminado

Aleatorio

Valores válidos

Random | Newer Offer

optimizationType

Descripción

La propiedad optimizationType define si Interact utiliza un motor de aprendizaje como ayuda en las asignaciones de ofertas. Si se establece en NoLearning, Interact no utiliza el aprendizaje. Si se establece en BuiltInLearning, Interact utiliza el motor de aprendizaje Bayesiano que se incorpora con Interact. Si se establece en ExternalLearning, Interact utiliza el motor de aprendizaje que proporcione. Si selecciona ExternalLearning, debe definir las propiedades externalLearningClass y externalLearningClassPath.

Valor predeterminado

NoLearning

Valores válidos

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

Descripción

El número máximo de milisegundos que el servidor de ejecución espera a que se complete un diagrama de flujo interactivo antes de recibir ofertas.

Valor predeterminado

5000

treatmentCodePrefix

Descripción

El prefijo que se añade a los códigos de tratamiento.

Valor predeterminado

No se ha definido ningún valor predeterminado.

effectiveDateBehavior

Descripción

Determina si Interact debe utilizar la fecha efectiva de una oferta en el filtrado de ofertas que se presentan a un visitante. Los valores incluyen los siguientes:

- -1 indica a Interact que ignore la fecha efectiva de la oferta.

  0 indica a Interact que utilice la fecha efectiva para filtrar la oferta, de forma que si la fecha efectiva de la oferta es anterior o igual a la fecha actual, la fecha efectiva de la oferta, la oferta se proporciona a los visitantes.

  Si hay establecido un valor **effectiveDateGracePeriod**, también se aplica el periodo de gracia para determinar si se debe proporcionar la oferta.
- Cualquier entero positivo indica a Interact que utilice la fecha actual más el valor de esta propiedad para determinar si se debe presentar la oferta a los visitantes, de forma que si la fecha efectiva de la oferta es anterior a la fecha actual más el valor de esta propiedad, la oferta se proporciona a los visitantes.

  Si hay establecido un valor **effectiveDateGracePeriod**, también se aplica el periodo de gracia para determinar si se debe proporcionar la oferta.

Valor predeterminado

-1

effectiveDateGracePeriodOfferAttr

Descripción

Especifica el nombre del atributo personalizado en una definición de oferta que indica el período de gracia de fecha efectiva. Por ejemplo, podría configurar esta propiedad con un valor de `AltGracePeriod`. A continuación, definiría las ofertas con un atributo personalizado denominado `AltGracePeriod` que se utiliza para especificar el número de días que se debe utilizar como periodo de gracia con la propiedad **effectiveDateBehavior**.

Supongamos que crea una plantilla de oferta nueva con una fecha efectiva de 10 días desde la fecha actual, e incluye un atributo personalizado denominado `AltGracePeriod`. Cuando se crea una oferta utilizando la plantilla, si establece el valor de `AltGracePeriod` en 14 días, la oferta se proporcionaría a los visitantes, ya que la fecha actual está dentro del periodo de gracia de la fecha efectiva de la oferta.

Valor predeterminado

En blanco

alwaysLogLearningAttributes

Descripción

Indica si Interact debe escribir información en los archivos de registro sobre los atributos de visitante utilizados por los módulos de aprendizaje. Tenga en cuenta que si se establece este valor en true el rendimiento del aprendizaje y el tamaño de los archivos de registro pueden resultar afectados.

Valor predeterminado

False

Interact | offerserving | Built-in Learning Config

Estas propiedades de configuración definen la configuración de escritura de base de datos para el aprendizaje incorporado. Para ajustar la implementación de aprendizaje, utilice las propiedades de configuración del entorno de diseño.

version

Descripción

Puede seleccionar 1 o 2. La Versión 1 es la versión de configuración básica que no utiliza parámetros para establecer los límites de subproceso y registro. La versión 2 es la versión de configuración ampliada que permite establecer el parámetro de subproceso y registro para mejorar el rendimiento. Estos parámetros realizan la agregación y la supresión cuando se alcanzan estos límites de parámetro.

Valor predeterminado

1

insertRawStatsIntervalInMinutes

Descripción

El número de minutos que el módulo de aprendizaje de Interact espera antes de insertar más filas en las tablas de preparación de aprendizaje.

Deberá modificar este tiempo según la cantidad de datos que el módulo de aprendizaje esté procesando en el entorno.

Valor predeterminado

5

Valores válidos

Un entero positivo

aggregateStatsIntervalInMinutes

Descripción

El número de minutos que el módulo de aprendizaje de Interact espera entre la agregación de datos en las tablas de estadísticas de aprendizaje. Deberá modificar este tiempo según la cantidad de datos que el módulo de aprendizaje esté procesando en el entorno.

Valor predeterminado

15

Valores válidos

Un entero mayor que cero.

autoAdjustPercentage

Descripción

El valor que determina el porcentaje de datos que la ejecución de la agregación intenta procesar basándose en las métricas de la ejecución anterior. De forma predeterminada, este valor se establece en cero, lo que significa que el agregador procesa todos los registros de preparación y esta función de ajuste automático está inhabilitada.

Valor predeterminado

0

Valores válidos

Un número entre 0 y 100.

enableObservationModeOnly

Descripción

Si se establece en `True`, habilita una modalidad de aprendizaje donde Interact recopila datos para aprendizaje sin utilizar esos datos para recomendaciones o arbitraje de ofertas. Esto le permite utilizar el autoaprendizaje en modo de inicio hasta que determine que se han recopilado datos suficientes para las recomendaciones.

Valor predeterminado

False

Valores válidos

True | False

excludeAbnormalAttribute

Descripción

El valor que determina si se deben marcar esos atributos como no válidos. Si se establece en `IncludeAttribute`, se incluyen atributos anormales que no están marcados como no válidos. Si se establece en `ExcludeAttribute`, los atributos anormales se excluyen y se marcan como no válidos.

Valor predeterminado

IncludeAttribute

Valores válidos

IncludeAttribute | ExcludeAttribute

Interact | offerserving | Built-in Learning Config | Parameter Data | [parameterName]

Estas propiedades de configuración definen los parámetros del módulo de aprendizaje externo.

numberOfThreads

Descripción

El número máximo de subprocesos que el agregador de aprendizaje utiliza para procesar los datos. Un valor válido es un entero positivo que no sea mayor que el número máximo de conexión configuradas en el origen de datos de aprendizaje. Este parámetro sólo se utiliza en el agregador versión 2.

Valor predeterminado

10

maxLogTimeSpanInMin

Descripción

Si se selecciona un agregador versión 1, puede procesar los registros de preparación en las iteraciones para evitar los procesos por lotes de base de datos demasiado grandes. En este caso, esos registros de preparación se procesan por fragmentos, iteración a iteración, en un único ciclo de agregación. El valor de este parámetro especifica el intervalo de tiempo máximo de registros de preparación que el agregador intenta procesar en cada iteración. Este intervalo de tiempo se basa en el campo LogTime que está asociado con cada registro de preparación, y sólo se procesan los registros cuyo LogTime pertenece a la ventana de tiempo más temprana. Un valor válido es un entero que no sea negativo. Si el valor es 0, no hay límite, lo que significa que todos los registros de preparación se procesan en una única iteración.

Valor predeterminado

0

maxRecords

Descripción

Si se selecciona un agregador versión 2, puede procesar los registros de preparación en las iteraciones para evitar los procesos por lotes de base de datos demasiado grandes. En este caso, esos registros de preparación se procesan en fragmentos, iteración a iteración, en un único ciclo de agregación. El valor de este parámetro especifica el número máximo de

registros de preparación que el agregador intenta procesar en cada iteración. Un valor válido es un entero que no sea negativo. Si el valor es 0, no hay límite, lo que significa que todos los registros de preparación se procesan en una única iteración.

Valor predeterminado

0

value

Descripción

El valor de un parámetro necesario para la clase para un módulo de aprendizaje incorporado.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Interact | offerserving | External Learning Config

Estas propiedades de configuración definen los valores de clase para un módulo de aprendizaje externo que ha escrito utilizando la API de aprendizaje.

class

Descripción

Si `optimizationType` se establece en `ExternalLearning`, establezca `externalLearningClass` en el nombre de clase del motor de aprendizaje externo.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Disponibilidad

Esta propiedad sólo es aplicable si `optimizationType` se ha establecido en `ExternalLearning`.

classPath

Descripción

Si `optimizationType` se establece en `ExternalLearning`, establezca `externalLearningClass` en la ruta de clases del motor de aprendizaje externo.

La ruta de clases debe hacer referencia a los archivos jar en el servidor del entorno de ejecución. Si utiliza un grupo de servidores y todos los servidores de ejecución utilizan la misma Marketing Platform, cada servidor debe tener una copia del archivo jar en la misma ubicación. La ruta de clases debe estar formada por las ubicaciones absolutas de los archivos jar, separadas por el delimitador de ruta del sistema operativo del servidor del entorno de ejecución, por ejemplo, un punto y coma (;) en Windows y dos puntos (:) en los sistemas UNIX. Los directorios que contienen archivos de clase no se aceptan. Por ejemplo, en un sistema UNIX: `/path1/file1.jar:/path2/file2.jar`.

Esta ruta de clases debe tener menos de 1024 caracteres. Puede utilizar el archivo de manifiesto en un archivo .jar para especificar otros archivos .jar para que sólo tenga que aparecer un archivo .jar en su ruta de clases.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Disponibilidad

Esta propiedad sólo es aplicable si `optimizationType` se ha establecido en `ExternalLearning`.

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

Estas propiedades de configuración definen los parámetros del módulo de aprendizaje externo.

value

Descripción

El valor de un parámetro necesario para la clase para un módulo de aprendizaje externo.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Ejemplo

Si el módulo de aprendizaje externo requiere una ruta a una aplicación de resolución de algoritmos, debe crear una categoría de parámetro denominada `solverPath` y definir la propiedad `value` como la ruta a la aplicación.

Interact | services

Las propiedades de configuración de esta categoría definen los valores de todos los servicios que gestionan la recopilación de datos y estadísticas del historial de contactos y respuestas para informar y escribir en las tablas del sistema del entorno de ejecución.

externalLoaderStagingDirectory

Descripción

Esta propiedad define la ubicación del directorio de preparación para una utilidad de carga de base de datos.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Valores válidos

Una ruta relativa al directorio de instalación de Interact o una ruta absoluta a un directorio de preparación.

Si habilita una utilidad de carga de base de datos, debe establecer la propiedad `cacheType` de las categorías `contactHist` y `responstHist` en `External Loader File`.

Interact | services | contactHist

Las propiedades de configuración de esta categoría definen los valores del servicio que recopila datos para las tablas de preparación del historial de contactos.

enableLog

Descripción

Si es `true`, habilita el servicio que recopila datos para registrar los datos del historial de contactos. Si es `false`, no se recopilan datos.

Valor predeterminado

True

Valores válidos

True | False

cacheType

Descripción

Define si los datos recopilados para el historial de contactos se mantienen en la memoria (`Memory Cache`) o en un archivo (`External Loader File`). Sólo puede utilizar `External Loader File` si ha configurado Interact para utilizar una utilidad de carga de base de datos.

Si selecciona `Memory Cache`, utilice la configuración de categoría `cache`. Si selecciona `External Loader File`, utilice la configuración de categoría `fileCache`.

Valor predeterminado

Memory Cache

Valores válidos

Memory Cache | External Loader File

Interact | services | contactHist | cache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila datos para la tabla de preparación del historial de contactos.

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba los datos recopilados del historial de contactos en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | contactHist | fileCache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila los datos del historial de contactos si utiliza una utilidad de carga de base de datos.

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba los datos recopilados del historial de contactos en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | defaultedStats

Las propiedades de configuración de esta categoría definen los valores del servicio que recopila las estadísticas sobre el número de veces que se ha utilizado la cadena predeterminada para el punto de interacción.

enableLog

Descripción

Si es `true`, habilita el servicio que recopila las estadísticas sobre el número de veces que se ha utilizado la cadena predeterminada para el punto de interacción en la tabla `UACI_DefaultedStat`. Si es `false`, no se recopilan estadísticas de cadenas predeterminadas.

Si no utiliza los informes de IBM, puede establecer esta propiedad en `false`, ya que la recopilación de datos no es necesaria.

Valor predeterminado

True

Valores válidos

True | False

Interact | services | defaultedStats | cache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila las estadísticas sobre el número de veces que se ha utilizado la cadena predeterminada para el punto de interacción.

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba las estadísticas recopiladas de cadenas predeterminadas en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | eligOpsStats

Las propiedades de configuración de esta categoría definen los valores del servicio que escribe las estadísticas de las ofertas elegibles.

enableLog

Descripción

Si es `true`, habilita el servicio que recopila las estadísticas de las ofertas elegibles. Si es `false`, no se recopilan estadísticas de ofertas elegibles.

Si no utiliza los informes de IBM, puede establecer esta propiedad en `false`, ya que la recopilación de datos no es necesaria.

Valor predeterminado

True

Valores válidos

True | False

Interact | services | eligOpsStats | cache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila las estadísticas de ofertas elegibles.

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba las estadísticas recopiladas de ofertas elegibles en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | eventActivity

Las propiedades de configuración de esta categoría definen los valores del servicio que recopila las estadísticas de actividades de eventos.

enableLog

Descripción

Si es `true`, habilita el servicio que recopila las estadísticas de actividades de eventos. Si es `false`, no se recopilan estadísticas de eventos.

Si no utiliza los informes de IBM, puede establecer esta propiedad en `false`, ya que la recopilación de datos no es necesaria.

Valor predeterminado

True

Valores válidos

True | False

Interact | services | eventActivity | cache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila las estadísticas de actividades de eventos.

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba las estadísticas recopiladas de actividades de eventos en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | eventPattern

Las propiedades de configuración de la categoría `eventPattern` definen los valores del servicio que recopila las estadísticas de actividad de patrones de eventos.

persistUnknownUserStates

Descripción

Determina si los estados de patrón de evento de un ID de audiencia desconocido (visitante) se retienen en la base de datos. De forma predeterminada, cuando finaliza una sesión, los estados de todos los patrones de eventos asociados al ID de audiencia de ese visitante se almacenan en la base de datos, siempre y cuando se conozca el ID de audiencia (es decir, que el perfil del visitante se encuentre en el origen de datos de perfiles).

La propiedad persistUnknownUserStates determina qué ocurre si no se conoce el ID de audiencia. De forma predeterminada, esta propiedad está establecida en False, y en caso de ID de audiencia desconocidos, los estados de patrón de evento se descartan al finalizar la sesión.

Si establece esta propiedad en True, los estados de patrón de evento de los usuarios desconocidos (cuyo perfil no se encuentre en el servicio de datos de perfil configurado) se conservarán.

Valor predeterminado

False

Valores válidos

True | False

mergeUnknowUserInSessionStates

Descripción

Determina cómo se conservan los estados de patrón de modelo de los ID de audiencia (visitantes) desconocidos. Si el ID de audiencia cambia en medio de una sesión, Interact intenta cargar los estados de patrón de evento guardados del nuevo ID de audiencia de la tabla de la base de datos. Si el ID de audiencia era desconocido anteriormente y la propiedad mergeUnknowUserInSessionStates se establece en True, las actividades de eventos de usuario pertenecientes al ID de audiencia anterior de la misma sesión se fusionarán con las del nuevo ID de audiencia.

Valor predeterminado

False

Valores válidos

True | False

enableUserEventLog

Descripción

Determina si las actividades de eventos de usuario se registran en la base de datos.

Valor predeterminado

False

Valores válidos

True | False

Interact | services | eventPattern | userEventCache

Las propiedades de configuración de la categoría userEventCache definen los valores que determinan cuándo la actividad de eventos pasa de la memoria caché a persistir en la base de datos.

threshold

Descripción

Determina el número máximo de estados de patrones de eventos que se pueden almacenar en la memoria caché de estados de patrones de eventos. Cuando se alcanza el límite, los estados utilizados menos recientements se eliminan de la memoria caché.

Valor predeterminado

100

Valores válidos

El número deseado de estados de patrones de eventos a retener en caché.

insertPeriodInSecs

Descripción

Determina el máximo de tiempo en segundos que las actividades de usuario se ponen en cola en la memoria. Cuando se alcanza el límite de tiempo especificado por esta propiedad, estas actividades se hacen persistir en la base de datos.

Valor predeterminado

3600 (60 minutos)

Valores válidos

El número de segundos deseado.

Interact | services | eventPattern | advancedPatterns

Las propiedades de configuración de esta categoría controlan si se habilita la integración con Interact Advanced Patterns, y definen los intervalos de tiempo de espera para las conexiones con Interact Advanced Patterns.

enableAdvancedPatterns

Descripción

Si es `true`, habilita la integración con Interact Advanced Patterns. Si es `false`, la integración no está habilitada. Si la integración se ha habilitado anteriormente, Interact utiliza los estados de patrón más recientes recibidos de Interact Advanced Patterns.

Valor predeterminado

True

Valores válidos

True | False

connectionTimeoutInMilliseconds

Descripción

Tiempo máximo que puede tardar el establecimiento de una conexión HTTP del entorno en tiempo real de Interact a Interact Advanced Patterns. Si la solicitud excede el tiempo de espera, Interact utiliza los datos guardados por última vez de los patrones.

Valor predeterminado

30

readTimeoutInMilliseconds

Descripción

Después de que se establezca una conexión HTTP entre el entorno en tiempo real de Interact y Interact Advanced Patterns, y se envíe una solicitud a Interact Advanced Patterns para obtener el estado de un patrón de evento, el tiempo máximo que puede transcurrir antes de recibir datos. Si la solicitud excede el tiempo de espera, Interact utiliza los datos guardados por última vez de los patrones.

Valor predeterminado

100

connectionPoolSize

Descripción

Tamaño de la agrupación de conexiones HTTP para la comunicación entre el entorno en tiempo real de Interact y Interact Advanced Patterns.

Valor predeterminado

10

Interact | services | eventPattern | advancedPatterns | autoReconnect

Las propiedades de configuración de esta categoría especifican los parámetros para la función de reconexión automática en la integración con Interact Advanced Patterns.

enable

Descripción

Determina si el sistema se debe reconectar automáticamente si se producen problemas de conexión entre el entorno en tiempo real de Interact y Interact Advanced Patterns. El valor predeterminado **True** habilita esta característica.

Valor predeterminado

True

Valores válidos

True | False

durationInMinutes

Descripción

Esta propiedad especifica el intervalo de tiempo, en minutos, durante el que el sistema evalúa problemas repetitivos de conexión entre el entorno de tiempo real de Interact y Interact Advanced Patterns.

Valor predeterminado

10

numberOfFailuresBeforeDisconnect

Descripción

Esta propiedad especifica el número de anomalías de conexión permitidas durante el período de tiempo especificado antes de que el sistema se desconecte automáticamente de Interact Advanced Patterns.

Valor predeterminado

3

consecutiveFailuresBeforeDisconnect

Descripción

Determina si la característica de reconexión automática sólo evalúa los errores consecutivos de la conexión entre el entorno de tiempo real de Interact con Interact Advanced Patterns. Si establece este valor en **False**, todos los errores dentro del intervalo de tiempo especificado se evalúan.

Valor predeterminado

True

sleepBeforeReconnectDurationInMinutes

Descripción

El sistema espera el número de minutos especificado en esta propiedad antes de volver a conectar después de que el sistema se desconecte debido a errores repetidos tal como se especifica en las otras propiedades en esta categoría.

Valor predeterminado

5

sendNotificationAfterDisconnect

Descripción

Esta propiedad determina si el sistema envía una notificación por correo electrónico cuando se produce un error de conexión. El mensaje de notificación incluye el nombre de instancia en tiempo real de Interact para el que se ha producido el error y el período de tiempo antes de que se produzca la reconexión, tal como se especifica en la propiedad **sleepBeforeReconnectDurationInMinutes**. El valor predeterminado **True** indica que se envían notificaciones.

Valor predeterminado

True

Interact | services | customLogger

Las propiedades de configuración de esta categoría definen los valores del servicio que recopila datos personalizados para escribir en una tabla (un evento que utiliza el parámetro de evento `UACICustomLoggerTableName`).

enableLog

Descripción

Si es `true`, habilita la característica de registro personalizado en la tabla. Si es `false`, el parámetro de evento `UACICustomLoggerTableName` no tiene ningún efecto.

Valor predeterminado

True

Valores válidos

True | False

Interact | services | customLogger | cache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila datos personalizados en una tabla (un evento que utiliza el parámetro de evento `UACICustomLoggerTableName`).

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba los datos personalizados recopilados en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | responseHist

Las propiedades de configuración de esta categoría definen los valores del servicio que escribe en las tablas de preparación del historial de respuestas.

enableLog

Descripción

Si es `true`, habilita el servicio que escribe en las tablas de preparación del historial de respuestas. Si es `false`, no se escriben datos en las tablas de preparación del historial de respuestas.

La tabla de preparación del historial de respuestas está definida por la propiedad `responseHistoryTable` del nivel de audiencia. El valor predeterminado es `UACI_RHStaging`.

Valor predeterminado

True

Valores válidos

True | False

cacheType

Descripción

Define si la memoria caché se mantiene en la memoria o en un archivo. Sólo puede utilizar `External Loader File` si ha configurado Interact para utilizar una utilidad de carga de base de datos.

Si selecciona `Memory Cache`, utilice la configuración de categoría `cache`. Si selecciona `External Loader File`, utilice la configuración de categoría `fileCache`.

Valor predeterminado

Memory Cache

Valores válidos

Memory Cache | External Loader File

actionOnOrphan

Descripción

Este valor determina qué hacer con los eventos de respuesta que no tienen eventos contacto correspondientes. Si se establece en `NoAction`, el evento de respuesta se procesa como si el evento de contacto correspondiente se ha publicado. Si se establece en `Warning`, el evento de respuesta se procesa como si el evento de contacto correspondiente se ha publicado, pero se graba un mensaje de aviso en `interact.log`. Si se establece en `Skip`, el evento respuesta no se procesa y se graba un mensaje de error en `interact.log`. El valor que elija aquí se aplicará independientemente de que el registro de historial de respuestas se haya habilitado.

Valor predeterminado

NoAction

Valores válidos

NoAction | Warning | Skip

Interact | services | responseHist | cache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila los datos del historial de respuestas.

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba los datos recopilados del historial de respuestas en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | responseHist | fileCache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila los datos del historial de respuestas si utiliza una utilidad de carga de base de datos.

threshold

Descripción

El número de registros acumulados antes de que Interact los escriba en la base de datos.

`responseHist` - La tabla definida por la propiedad `responseHistoryTable` del nivel de audiencia. El valor predeterminado es `UACI_RHStaging`.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la base de datos.

Valor predeterminado

3600

Interact | services | crossSessionResponse

Las propiedades de configuración de esta categoría definen los valores generales para el servicio crossSessionResponse y el proceso xsession. Sólo necesita configurar estos valores si está utilizando el seguimiento de respuestas de sesiones cruzadas de Interact.

enableLog

Descripción

Si es `true`, habilita el servicio crossSessionResponse y Interact escribe datos en las tablas de preparación de seguimiento de respuestas de sesiones cruzadas. Si se establece en `false`, inhabilita el servicio crossSessionResponse.

Valor predeterminado

False

xsessionProcessIntervalInSecs

Descripción

El número de segundos entre las ejecuciones del proceso xsession. Este proceso mueve los datos de las tablas de preparación de seguimiento de respuestas de sesiones cruzadas a la tabla de preparación del historial de respuestas y el módulo de aprendizaje incorporado.

Valor predeterminado

180

Valores válidos

Un entero mayor que cero

purgeOrphanResponseThresholdInMinutes

Descripción

El número de minutos que espera el servicio crossSessionResponse antes de marcar las respuestas que no coinciden con los contactos en las tablas del historial de contactos y respuestas.

Si una respuesta no tiene ninguna coincidencia en las tablas de historial de contactos y respuestas después de `purgeOrphanResponseThresholdInMinutes` minutos, Interact marca la respuesta con un valor -1 en la columna `Mark` de la tabla de preparación xSessResponse. A continuación, puede hacer coincidir o suprimir manualmente estas respuestas.

Valor predeterminado

180

Interact | services | crossSessionResponse | cache

Las propiedades de configuración de esta categoría definen la configuración de la memoria caché del servicio que recopila los datos de respuestas de sesiones cruzadas.

threshold

Descripción

El número de registros acumulados antes de que el servicio flushCacheToDB escriba los datos de respuestas de sesiones cruzadas recopilados en la base de datos.

Valor predeterminado

100

insertPeriodInSecs

Descripción

El número de segundos entre las escrituras forzadas en la tabla `XSessResponse`.

Valor predeterminado

3600

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

Las propiedades de esta sección definen cómo el seguimiento de respuestas de sesiones cruzadas hace coincidir los códigos de tratamiento con el historial de contactos y respuestas.

SQL

Descripción

Esta propiedad define si Interact utiliza el SQL generado por el sistema o el SQL personalizado definido en la propiedad `OverrideSQL`.

Valor predeterminado

Utilizar SQL generado por el sistema

Valores válidos

Utilizar SQL generado por el sistema | Alterar temporalmente SQL

OverrideSQL

Descripción

Si no utiliza el comando SQL predeterminado para hacer coincidir el código de tratamiento con el historial de contactos y respuestas, especifique aquí el SQL o el procedimiento almacenado.

Este valor se ignora si `SQL` se establece en `Utilizar SQL generado por el sistema`.

Valor predeterminado

useStoredProcedure

Descripción

Si se establece en true, `OverrideSQL` debe incluir una referencia a un procedimiento almacenado que haga coincidir el código de tratamiento con el historial de contactos y respuestas.

Si se establece en false, `OverrideSQL`, cuando se utiliza, debe ser una consulta SQL.

Valor predeterminado

false

Valores válidos

true | false

Tipo

Descripción

El TrackingCodeType asociado definido en la tabla `UACI_TrackingType` en las tablas del entorno de ejecución. A menos que revise la tabla `UACI_TrackingType`, `Type` debe ser 1.

Valor predeterminado

1

Valores válidos

Un entero definido en la tabla `UACI_TrackingType`.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

Las propiedades de esta sección definen cómo el seguimiento de respuestas de sesiones cruzadas hace coincidir los códigos de oferta con el historial de contactos y respuestas.

SQL

Descripción

Esta propiedad define si Interact utiliza el SQL generado por el sistema o el SQL personalizado definido en la propiedad `OverrideSQL`.

Valor predeterminado

Utilizar SQL generado por el sistema

Valores válidos

Utilizar SQL generado por el sistema | Alterar temporalmente SQL

OverrideSQL

Descripción

Si no utiliza el comando SQL predeterminado para hacer coincidir el código de oferta con el historial de contactos y respuestas, especifique aquí el SQL o el procedimiento almacenado.

Este valor se ignora si `SQL` se establece en `Utilizar SQL generado por el sistema`.

Valor predeterminado

useStoredProcedure

Descripción

Si se establece en true, `OverrideSQL` debe incluir una referencia a un procedimiento almacenado que haga coincidir el código de oferta con el historial de contactos y respuestas.

Si se establece en false, `OverrideSQL`, cuando se utiliza, debe ser una consulta SQL.

Valor predeterminado

false

Valores válidos

true | false

Tipo

Descripción

El TrackingCodeType asociado definido en la tabla `UACI_TrackingType` en las tablas del entorno de ejecución. A menos que revise la tabla `UACI_TrackingType`, `Type` debe ser 2.

Valor predeterminado

2

Valores válidos

Un entero definido en la tabla `UACI_TrackingType`.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

Las propiedades de esta sección definen cómo el seguimiento de respuestas de sesiones cruzadas hace coincidir un código alternativo definido por el usuario con el historial de contactos y respuestas.

Nombre

Descripción

Esta propiedad define el nombre del código alternativo. Debe coincidir con el valor de Name en la tabla UACI_TrackingType en las tablas del entorno de ejecución.

Valor predeterminado

OverrideSQL

Descripción

El comando SQL o el procedimiento almacenado que hace coincidir el código alternativo con el historial de contactos y respuestas por código de oferta o código de tratamiento.

Valor predeterminado

useStoredProcedure

Descripción

Si se establece en true, OverrideSQL debe incluir una referencia a un procedimiento almacenado que haga coincidir el código alternativo con el historial de contactos y respuestas.

Si se establece en false, OverrideSQL, cuando se utiliza, debe ser una consulta SQL.

Valor predeterminado

false

Valores válidos

true | false

Tipo

Descripción

El TrackingCodeType asociado definido en la tabla UACI_TrackingType en las tablas del entorno de ejecución.

Valor predeterminado

3

Valores válidos

Un entero definido en la tabla UACI_TrackingType.

Interact | services | threadManagement | contactAndResponseHist

Las propiedades de configuración de esta categoría definen los valores de gestión de subprocesos para los servicios que recopilan datos para las tablas de preparación del historial de contactos y respuestas.

corePoolSize

Descripción

El número de subprocesos que se mantendrán en la agrupación, aunque estén desocupados, para recopilar los datos del historial de contactos y respuestas.

Valor predeterminado

5

maxPoolSize

Descripción

El número máximo de subprocesos que se mantendrán en la agrupación para recopilar los datos del historial de contactos y respuestas.

Valor predeterminado

5

keepAliveTimeSecs

Descripción

Cuando el número de subprocesos es mayor que el principal, es el tiempo máximo que los subprocesos desocupados excedentes esperan nuevas tareas antes de finalizar para recopilar los datos del historial de contactos y respuestas.

Valor predeterminado

5

queueCapacity

Descripción

El tamaño de la cola utilizada por la agrupación de subprocesos para recopilar los datos del historial de contactos y respuestas.

Valor predeterminado

1000

termWaitSecs

Descripción

Al cerrar el servidor de ejecución, es el número de segundos que se espera a que los subprocesos de servicio terminen de recopilar los datos del historial de contactos y respuestas.

Valor predeterminado

5

Interact | services | threadManagement | allOtherServices

Las propiedades de configuración en esta categoría definen los valores de gestión de subprocesos para los servicios que recopilan las estadísticas de elegibilidad de ofertas, las estadísticas de actividades de eventos, las estadísticas de uso de cadenas predeterminadas y el registro personalizado en los datos de la tabla.

corePoolSize

Descripción

El número de subprocesos que se mantendrán en la agrupación, aunque estén desocupados, para los servicios que recopilan las estadísticas de elegibilidad de ofertas, las estadísticas de actividades de eventos, las estadísticas de uso de cadenas predeterminadas y el registro personalizado en los datos de la tabla.

Valor predeterminado

5

maxPoolSize

Descripción

El número máximo de subprocesos que se mantendrán en la agrupación para los servicios que recopilan las estadísticas de elegibilidad de ofertas, las estadísticas de actividades de eventos, las estadísticas de uso de cadenas predeterminadas y el registro personalizado en los datos de la tabla.

Valor predeterminado

5

keepAliveTimeSecs

Descripción

Cuando el número de subprocesos es mayor que el principal, es el tiempo máximo que los subprocesos desocupados excedentes esperan nuevas tareas antes de finalizar para los servicios que recopilan las estadísticas de elegibilidad de ofertas, las estadísticas de actividades de eventos, las estadísticas de uso de cadenas predeterminadas y el registro personalizado en los datos de la tabla.

Valor predeterminado

5

queueCapacity

Descripción

El tamaño de la cola utilizada por la agrupación de subprocesos para los servicios que recopilan las estadísticas de elegibilidad de ofertas, las estadísticas de actividades de eventos, las estadísticas de uso de cadenas predeterminadas y el registro personalizado en los datos de la tabla.

Valor predeterminado

1000

termWaitSecs

Descripción

Al cerrar el servidor de ejecución, es el número de segundos que se espera a que finalicen los subprocesos de servicio para los servicios que recopilan las estadísticas de elegibilidad de ofertas, las estadísticas de actividades de eventos, las estadísticas de uso de cadenas predeterminadas y el registro personalizado en los datos de la tabla.

Valor predeterminado

5

Interact | services | threadManagement | flushCacheToDB

Las propiedades de configuración de esta categoría definen los valores de gestión de subprocesos para los subprocesos que escriben datos recopilados de la memoria caché en las tablas de base de datos del entorno de ejecución.

corePoolSize

Descripción

El número de subprocesos que se mantienen en la agrupación para los subprocesos planificados que escriben datos guardados en caché en el almacén de datos.

Valor predeterminado

5

maxPoolSize

Descripción

El número máximo de subprocesos que se mantienen en la agrupación para los subprocesos planificados que escriben datos guardados en caché en el almacén de datos.

Valor predeterminado

5

keepAliveTimeSecs

Descripción

Cuando el número de subprocesos es mayor que el principal, es el tiempo máximo que los subprocesos desocupados excedentes esperan nuevas tareas antes de finalizar para los subprocesos planificados que escriben datos guardados en caché en el almacén de datos.

Valor predeterminado

5

queueCapacity

Descripción

El tamaño de la cola utilizada por la agrupación de subprocesos para los subprocesos planificados que escriben datos guardados en caché en el almacén de datos.

Valor predeterminado

1000

termWaitSecs

Descripción

Al cerrar el servidor de ejecución, es el número de segundos que se espera a que finalicen los subprocesos de servicio para los subprocesos planificados que escriben datos guardados en caché en el almacén de datos.

Valor predeterminado

5

Interact | services | configurationMonitor

Las propiedades de configuración en esta categoría permiten habilitar o inhabilitar la integración con Interact Advanced Patterns sin necesidad de reiniciar Interact en tiempo real, y definen el intervalo para sondear el valor de propiedad que habilita la integración.

enable

Descripción

Si es `true`, habilita el servicio que renueva el valor de la propiedad **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**. Si es `false`, debe reiniciar Interact en tiempo real al cambiar el valor de la propiedad **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**.

Valor predeterminado

False

Valores válidos

True | False

refreshIntervalInMinutes

Descripción

Define el intervalo de tiempo para el sondeo del valor de la propiedad **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**.

Valor predeterminado

5

Interact | cacheManagement

Este conjunto de propiedades de configuración define los valores para seleccionar y configurar cada uno de los gestores de memoria caché soportados que puede utilizar para mejorar el rendimiento de Interact, como EHCache, que está incorporado en la instalación de Interact, o el almacenamiento en memoria caché de WebSphere eXtreme Scale, que es un complemento opcional, u otro sistema de almacenamiento en memoria caché externo.

Utilice las propiedades de configuración **Interact | cacheManagement | Cache Managers** para configurar el gestor de memoria caché que desee utilizar. Utilice las propiedades de configuración **Interact | cacheManagement | caches** para especificar qué gestor de memoria caché debe utilizar Interact para mejorar el rendimiento.

Interact | cacheManagement | Cache Managers

La categoría Cache Managers especifica los parámetros de las soluciones de gestión de memoria caché que tiene previsto utilizar con Interact.

Interact | cacheManagement | Cache Managers | EHCache

La categoría EHCache especifica los parámetros de la solución de gestión de memoria caché de EHCache, para que pueda personalizarla para mejorar el rendimiento de Interact.

Interact | Cache Managers | EHCache | Parameter Data

Las propiedades de configuración de esta categoría controlan cómo funciona el sistema de gestión de memoria caché de EHCache para mejorar el rendimiento de Interact.

cacheType

Descripción

Puede configurar los servidores de ejecución de Interact de un grupo de servidores para que utilicen una dirección de multidifusión para la compartición de datos de memoria caché. Esto se conoce como *memoria caché distribuida*. El parámetro cacheType especifica si está utilizando el mecanismo de almacenamiento en memoria caché de EHCache incorporado en modo **local** (autónomo) o **distribuido** (como con un grupo de servidores de ejecución).

Nota:

Si selecciona **Distributed** como cacheType, todos los servidores que comparten la memoria caché deben formar parte del mismo único grupo de servidores. También debe habilitar la multidifusión para trabajar entre todos los miembros de un grupo de servidores.

Valor predeterminado

Local

Valores válidos

Local | Distributed

multicastIPAddress

Descripción

Si especifica que el parámetro **cacheType** es "distributed", está configurando la memoria caché para que funcione mediante multidifusión entre todos los miembros de un grupo de servidores de ejecución de Interact. El valor de multicastIPAddress es la dirección IP que utilizan para la escucha todos los servidores de Interact para el grupo de servidores.

La dirección IP debe ser exclusiva entre los grupos de servidores.

Valor predeterminado

230.0.0.1

multicastPort

Descripción

Si especifica que el parámetro **cacheType** es "distributed", el parámetro **multicastPort** indica el puerto que utilizan para la escucha todos los servidores de Interact para el grupo de servidores.

Valor predeterminado

6363

overflowToDisk

Descripción

El gestor de memoria caché de EHCache gestiona la información de sesión utilizando la memoria disponible. Para los entornos donde el tamaño de sesión es grande debido a un perfil grande, el número de sesiones soportado en la memoria puede que no sea lo suficientemente grande para dar soporte al escenario del cliente. Si este es el caso, EHCache tiene una característica opcional para permitir que la información de memoria caché con un tamaño mayor al que se puede mantener en la memoria se grabe temporalmente en el disco duro en su lugar.

Si establece la propiedad **overflowToDisk** en "yes", cada máquina virtual Java (JVM) puede manejar más sesiones simultáneas de las que permitiría solo la memoria.

Valor predeterminado

No

Valores válidos

No | Yes

diskStore

Descripción

Cuando la propiedad de configuración **overflowToDisk** se establece en `Yes`, esta propiedad de configuración especifica el directorio de disco que contendrá las entradas de memoria caché que se han desbordado de la memoria. Si esta propiedad de configuración no existe o su valor no es válido, el directorio de disco se crea automáticamente en el directorio temporal predeterminado del sistema operativo.

Valor predeterminado

Ninguno

Valores válidos

Un directorio en el que la aplicación web que aloja el tiempo de ejecución de Interact tiene privilegios de escritura.

(Parámetro)

Descripción

Una plantilla que puede utilizar para crear un parámetro personalizado que se utilizará con el gestor de memoria caché. Puede configurar el nombre de parámetro que desee y el valor que debe tener.

Para crear un parámetro personalizado, pulse ***(Parámetro)***, y complete el nombre y el valor que desee asignar a este parámetro. Cuando pulsa **Guardar cambios**, el parámetro que ha creado se añade a la lista en la categoría de datos de parámetro.

Valor predeterminado

Ninguno

Interact | cacheManagement | Cache Managers | Extreme Scale

La categoría Extreme Scale especifica los parámetros del adaptador para utilizar la solución de gestión de memoria caché de WebSphere eXtreme Scale, para que pueda personalizarla para mejorar el rendimiento de Interact.

ClassName

Descripción

Nombre completo de la clase que conecta Interact al servidor de WebSphere eXtreme Scale. Debe ser `com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`.

Valor predeterminado

`com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`

ClassPath

Descripción

El URI de la ubicación del archivo `interact_wxs_adapter.jar` como, por ejemplo, `file:///IBM/EMM/Interact/lib/interact_wxs_adapter.jar` o `file:///C:/IBM/EMM/Interact/lib/interact_wxs_adapter.jar`. Sin embargo, si este archivo jar ya está incluido en la ruta de clases del servidor de aplicaciones que lo aloja, este campo se debe dejar en blanco.

Valor predeterminado

En blanco

Interact | Cache Managers | Extreme Scale | Parameter Data

Las propiedades de configuración de esta categoría controlan el adaptador de WebSphere eXtreme Scale que se incluye opcionalmente con su instalación de Interact. Estos valores se deben configurar para cada servidor de ejecución de Interact que funcione como cliente en la cuadrícula de servidores de eXtreme Scale.

catalogPropertyFile

Descripción

El URI de la ubicación del archivo de propiedades utilizado para iniciar el servidor de catálogo de WebSphere eXtreme Scale. Si el adaptador de Extreme Scale se utiliza para iniciar el servidor de catálogo, esta propiedad debe establecerse. De lo contrario, no se utilizará.

Valor predeterminado

`file:///C:/depot/Interact/dev/main/extremescale/config/catalogServer.props`

containerPropertyFile

Descripción

El URI de la ubicación del archivo de propiedades utilizado para iniciar las instancias de contenedor de WebSphere eXtreme Scale. Si el componente de servidor incluido se utiliza para iniciar los servidores de contenedor de WebSphere eXtreme Scale, esta propiedad se debe establecer. De lo contrario, no se utiliza.

Valor predeterminado

`file:///C:/depot/Interact/dev/main/extremescale/config/containerServer.props`

deploymentPolicyFile

Descripción

El URI de la ubicación del archivo de política de despliegue utilizado para iniciar el servidor de catálogo de WebSphere eXtreme Scale. Si el componente de servidor incluido se utiliza para iniciar el servidor de catálogo de WebSphere eXtreme Scale, esta propiedad se debe establecer. De lo contrario, no se utiliza.

Valor predeterminado

```
file:///C:/depot/Interact/dev/main/extremescale/config/
deployment.xml
```

objectGridConfigFile

Descripción

El URI de la ubicación del archivo de configuración de la cuadrícula de objetos utilizado para iniciar el servidor de catálogo de WebSphere eXtreme Scale y también el componente casi de memoria caché que se ejecuta conjuntamente con el servidor de ejecución de Interact en la misma máquina virtual Java (JVM).

Valor predeterminado

```
file:///C:/depot/Interact/dev/main/extremescale/config/
objectgrid.xml
```

gridName

Descripción

El nombre de la cuadrícula de WebSphere eXtreme Scale que contiene todas las memorias caché de Interact.

Valor predeterminado

InteractGrid

catalogURLs

Descripción

Un URL que contiene el nombre de host o dirección IP y el puerto en que el servidor de catálogo de WebSphere eXtreme Scale está a la escucha de nuevas conexiones.

Valor predeterminado

Ninguno

(Parámetro)

Descripción

Una plantilla que puede utilizar para crear un parámetro personalizado que se utilizará con el gestor de memoria caché. Puede configurar el nombre de parámetro que desee y el valor que debe tener.

Para crear un parámetro personalizado, pulse ***(Parámetro)***, y complete el nombre y el valor que desee asignar a este parámetro. Cuando pulsa **Guardar cambios**, el parámetro que ha creado se añade a la lista en la categoría de datos de parámetro.

Valor predeterminado

Ninguno

Interact | caches

Utilice este conjunto de propiedades de configuración para especificar qué gestor de memoria caché soportado desea utilizar para mejorar el rendimiento de Interact como, por ejemplo, Ehcache o el almacenamiento en memoria caché de WebSphere eXtreme Scale, y para configurar las propiedades de memoria caché específicas del servidor de ejecución que está configurando.

Esto incluye las memorias caché para almacenar datos de sesión, estados de patrón de evento y resultados de segmentación. Ajuste estos valores para especificar qué solución de memoria caché se utilizará para cada tipo de almacenamiento en memoria caché, y también para especificar valores individuales para controlar cómo funcionará la memoria caché.

Interact | cacheManagement | caches | InteractCache

La categoría InteractCache configura el almacenamiento en memoria caché de todos los objetos de sesión, incluidos los datos de perfil, los resultados de segmentos, los tratamientos entregados más recientemente, los parámetros pasados mediante los métodos de la API y otros objetos utilizados por el tiempo de ejecución de Interact.

La categoría InteractCache es necesaria para que Interact funcione correctamente.

La categoría InteractCache también puede configurarse mediante una configuración de EHCache externa para los valores que no están soportados en **Interact | cacheManagement | Caches**. Si utiliza EHCache, debe asegurarse de que InteractCache se haya configurado correctamente.

CacheManagerName

Descripción

El nombre del gestor de memoria caché que maneja la memoria caché de Interact. El valor que especifique aquí debe ser uno de los gestores de memoria caché definidos en las propiedades de configuración **Interact | cacheManagement | Cache Managers** como, por ejemplo, `EHCache` o `Extreme Scale`.

Valor predeterminado

EHCache

Valores válidos

Cualquier gestor de memoria caché definido en la propiedad de configuración **Interact | cacheManagement | Cache Managers**.

maxEntriesInCache

Descripción

El número máximo de objetos de datos de sesión que se almacenarán en esta memoria caché. Cuando se haya alcanzado el número máximo de objetos de datos de sesión y sea necesario almacenar datos para una sesión adicional, se suprimirá el objeto utilizado menos recientemente.

Valor predeterminado

100000

Valores válidos

Un entero mayor que 0.

timeoutInSecs

Descripción

El tiempo, en segundos, transcurrido desde la utilización o actualización de un objeto de datos de sesión que se utiliza para determinar cuándo se elimina el objeto de la memoria caché.

Valor predeterminado

300

Valores válidos

Un entero mayor que 0.

Interact | Caches | Interact Cache | Parameter Data

Las propiedades de configuración de esta categoría controlan la memoria caché de Interact que utiliza automáticamente la instalación de Interact. Estos valores se deben configurar individualmente para cada servidor de ejecución de Interact.

asyncIntervalMillis

Descripción

El tiempo en milisegundos que el gestor de memoria caché EHCache debe esperar antes de replicar los cambios en otras instancias de ejecución de Interact. Si el valor no es positivo, estos cambios se replicarán de forma síncrona.

Esta propiedad de configuración no se crea de forma predeterminada. Si crea esta propiedad, se utiliza solo cuando EHCache es el gestor de memoria caché, y cuando la propiedad **cacheType** de ehCache está establecida en `distributed`.

Valor predeterminado

Ninguno.

(Parámetro)

Descripción

Una plantilla que puede utilizar para crear un parámetro personalizado que se utilizará con la memoria caché de Interact. Puede configurar el nombre de parámetro que desee y el valor que debe tener.

Para crear un parámetro personalizado, pulse ***(Parámetro)***, y complete el nombre y el valor que desee asignar a este parámetro. Cuando pulsa **Guardar cambios**, el parámetro que ha creado se añade a la lista en la categoría de datos de parámetro.

Valor predeterminado

Ninguno

Interact | cacheManagement | caches | PatternStateCache

La categoría PatternStateCache se utiliza para alojar los estados de patrones de eventos y reglas de supresión de ofertas en tiempo real. De forma predeterminada, esta memoria caché está configurada como una memoria caché de lectura y escritura, de forma que Interact intenta utilizar el primer patrón de evento de memoria caché y los datos de supresión de oferta. Si la entrada solicitada no existe

en la memoria caché, la implementación de memoria caché la carga desde el origen de datos, mediante la configuración JNDI o directamente mediante una conexión JDBC.

Para utilizar una conexión JNDI, Interact se conecta a un proveedor de origen de datos existente que se ha definido mediante el servidor especificado utilizando el nombre JNDI, URL, etc. Para una conexión JDBC, debe proporcionar un conjunto de valores JDBC que incluyan el nombre de clase de controlador JDBC, el URL de base de datos y la información de autenticación.

Tenga en cuenta que si define varios orígenes JNDI y JDBC, se utilizará el primer origen JNDI habilitado y, si no hay orígenes JNDI habilitados, se utilizará el primer origen JDBC habilitado.

La categoría PatternStateCache es necesaria para que Interact funcione correctamente.

La categoría PatternStateCache también puede configurarse mediante una configuración de EHCache externa para los valores que no están soportados en **Interact | cacheManagement | Caches**. Si utiliza EHCache, debe asegurarse de que PatternStateCache se haya configurado correctamente.

CacheManagerName

Descripción

El nombre del gestor de memoria caché que maneja la memoria caché de estados de patrón de Interact. El valor que especifique aquí debe ser uno de los gestores de memoria caché definidos en las propiedades de configuración **Interact | cacheManagement | Cache Managers** como, por ejemplo, `EHCache` o `Extreme Scale`.

Valor predeterminado

EHCache

Valores válidos

Cualquier gestor de memoria caché definido en la propiedad de configuración **Interact | cacheManagement | Cache Managers**.

maxEntriesInCache

Descripción

El número máximo de estados de patrón de evento que se almacenarán en la memoria caché. Cuando se haya alcanzado el número máximo de estados de patrón de evento y sea necesario almacenar datos para un estado de patrón de evento adicional, se suprimirá el objeto utilizado menos recientemente.

Valor predeterminado

100000

Valores válidos

Un entero mayor que 0.

timeoutInSecs

Descripción

Especifica el tiempo de espera, en segundos, de un objeto de estado de patrón de evento en la memoria caché de estados de patrones de eventos. Si un objeto de estado de este tipo ha estado desocupado en la memoria caché durante `timeoutInSecs` número de segundos, se puede expulsar de la memoria caché según la regla de utilizado menos recientemente. Observe que el valor de esta propiedad debe ser mayor que el definido en la propiedad `sessionTimeoutInSecs`.

Valor predeterminado

300

Valores válidos

Un entero mayor que 0.

Interact | Caches | PatternStateCache | Parameter Data:

Las propiedades de configuración de esta categoría controlan la memoria caché de estados de patrón utilizada para alojar los estados de patrones de eventos y las reglas de supresión de ofertas en tiempo real.

(Parámetro)

Descripción

Una plantilla que puede utilizar para crear un parámetro personalizado que se utilizará con la memoria caché de estados de patrón. Puede configurar el nombre de parámetro que desee y el valor que debe tener.

Para crear un parámetro personalizado, pulse ***(Parámetro)***, y complete el nombre y el valor que desee asignar a este parámetro. Cuando pulsa **Guardar cambios**, el parámetro que ha creado se añade a la lista en la categoría de datos de parámetro.

Valor predeterminado

Ninguno

Interact | cacheManagement | caches | PatternStateCache | loaderWriter:

La categoría **loaderWriter** contiene la configuración del cargador que interactúa con repositorios externos para garantizar la recuperación y la persistencia de patrones de eventos.

className

Descripción

El nombre de clase completo de este cargador. Esta clase debe cumplir el requisito del gestor de memoria caché elegido.

Valor predeterminado

```
com.unicacorp.interact.cache.ehcache.loaderwriter.
PatternStateEHCacheLoaderWriter
```

Valores válidos

Un nombre de clase completo.

classPath

Descripción

La ruta del archivo de clase del cargador. Si deja este valor en blanco o la entrada no es válida, se utiliza la ruta de clases para ejecutar Interact.

Valor predeterminado

Ninguno

Valores válidos

Una ruta de clase válida.

writeMode

Descripción

Especifica el modo del grabador para persistir los estados de patrones de eventos nuevos o actualizados en la memoria caché. Las opciones válidas son:

- WRITE_THROUGH. Cada vez que hay una nueva entrada o se actualiza una entrada existente, dicha entrada se graba inmediatamente en los repositorios.
- WRITE_BEHIND. El gestor de memoria caché espera un tiempo para recopilar un determinado número de cambios y, a continuación, los persiste en los repositorios en lote.

Valor predeterminado

WRITE_THROUGH

Valores válidos

WRITE_THROUGH o WRITE_BEHIND.

batchSize

Descripción

El número máximo de objetos de estado de patrón de evento que el grabador persistirá en un lote. Esta propiedad sólo se utiliza cuando **writeMode** está establecido en `WRITE_BEHIND`.

Valor predeterminado

100

Valores válidos

Valor entero.

maxDelayInSecs

Descripción

El tiempo máximo, en segundos, que el gestor de memoria caché espera antes de que un objeto de estado de patrón de evento sea persistente. Esta propiedad sólo se utiliza cuando **writeMode** está establecido en `WRITE_BEHIND`.

Valor predeterminado

5

Valores válidos

Valor entero.

Interact | Caches | PatternStateCache | loaderWriter | Parameter Data:

Las propiedades de configuración de esta categoría controlan el cargador de memoria caché de estados de patrón.

(Parámetro)

Descripción

Una plantilla que puede utilizar para crear un parámetro personalizado que se utilizará con el cargador de memoria caché de estados de patrón. Puede configurar el nombre de parámetro que desee y el valor que debe tener.

Para crear un parámetro personalizado, pulse ***(Parámetro)***, y complete el nombre y el valor que desee asignar a este parámetro. Cuando pulsa **Guardar cambios**, el parámetro que ha creado se añade a la lista en la categoría de datos de parámetro.

Valor predeterminado

Ninguno

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jndiSettings:

La categoría **jndiSettings** contiene la configuración del origen de datos JNDI que el cargador utilizará para la comunicación con la base de datos de respaldo. Para crear un nuevo conjunto de valores JNDI, expanda la categoría **jdniSettings** y pulse la propiedad ***(jndiSetting)***.

(jndiSettings)

Nota: Cuando se utiliza WebSphere Application Server, loaderWriter no se conecta con **jndiSettings**.

Descripción

Al pulsar esta categoría, aparece un formulario. Para definir un origen de datos JNDI, complete los valores siguientes:

- **Nuevo nombre de categoría** es el nombre que desea utilizar para identificar esta conexión JNDI.
- **enabled** permite indicar si desea que esta conexión JNDI esté o no disponible para su uso. Establézcala en `True` para las conexiones nuevas.
- **jdniName** es el nombre JNDI que ya se ha definido en el origen de datos al configurarlo.
- **providerUrl** es el URL que se utiliza para buscar este origen de datos JNDI. Si deja este campo en blanco, se utiliza el URL de la aplicación web que aloja el tiempo de ejecución de Interact.
- **Fábrica de contexto inicial** es el nombre de clase completo de la clase de fábrica de contexto inicial para conectarse al proveedor JNDI. Si se utiliza la aplicación web que aloja el tiempo de ejecución de Interact para **providerUrl**, deje este campo en blanco.

Valor predeterminado

Ninguno.

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jdbcSettings:

La categoría **jdbcSettings** contiene la configuración de las conexiones JDBC que utilizará el cargador para la comunicación con la base de datos de respaldo. Para crear un nuevo conjunto de valores JDBC, expanda la categoría **jdbcSettings** y pulse la propiedad ***(jdbcSetting)***.

(jdbcSettings)

Descripción

Al pulsar esta categoría, aparece un formulario. Para definir un origen de datos JDBC, complete los valores siguientes:

- **Nuevo nombre de categoría** es el nombre que desea utilizar para identificar esta conexión JDBC.
- **enabled** permite indicar si desea que esta conexión JDBC esté o no disponible para su uso. Establézcala en `True` para las conexiones nuevas.
- **driverClassName** es el nombre de clase completo del controlador JDBC. Esta clase debe existir en la ruta de clase configurada para iniciar el servidor de memoria caché que lo aloja.
- **databaseUrl** es el URL que se utiliza para buscar este origen de datos JDBC.
- **asmUser** es el nombre del usuario de IBM EMM que se ha configurado con las credenciales para conectar a la base de datos en esta conexión JDBC.
- **asmDataSource** es el nombre del origen de datos de IBM EMM que se ha configurado con las credenciales para conectarse a la base de datos en esta conexión JDBC.
- **maxConnection** es el número máximo de conexiones simultáneas que se permite realizar a la base de datos en esta conexión JDBC.

Valor predeterminado

Ninguno.

Interact | triggeredMessage

Las propiedades de configuración de esta categoría definen los valores para todos los mensajes desencadenados y la entrega del canal de oferta.

backendProcessIntervalMin

Descripción

Esta propiedad define el periodo de tiempo, en minutos, durante el cual la hebra de proceso de fondo carga y procesa las entregas de ofertas con retardo. Este valor debe ser un entero. Si el valor es cero o es negativo, se inhabilita el proceso de fondo.

Valores válidos

Un entero positivo

autoLogContactAfterDelivery

Descripción

Si esta propiedad se establece en true, se publica automáticamente un evento de contacto en cuanto se asigna esta oferta, o la oferta se pone en cola para su entrega con retardo. Si esta propiedad se establece en false, no se publica automáticamente ningún evento de contacto para las ofertas de salida. Ese es el comportamiento predeterminado.

Valores válidos

True | False

waitForFlowchart

Descripción

Esta propiedad determina si el diagrama de flujo debe esperar a que finalice la ejecución actual de la segmentación, y el comportamiento si se supera el tiempo de espera.

DoNotWait: El proceso de un mensaje desencadenado se inicia independientemente de si la segmentación se está ejecutando o no actualmente. Sin embargo, si se utilizan segmentos en la regla de elegibilidad y/o se selecciona NextBestOffer como el método de selección de ofertas, la ejecución de TM continúa esperando.

OptionalWait: El proceso de un mensaje desencadenado espera hasta que finalice o transcurra el tiempo de espera de la segmentación que se está ejecutando actualmente. Si finaliza el tiempo de espera, se registra un aviso y el proceso de este mensaje desencadenado continúa. Es el valor predeterminado.

MandatoryWait: El proceso de un mensaje desencadenado espera hasta que finalice o transcurra el tiempo de espera de la segmentación que se está ejecutando actualmente. Si finaliza el tiempo de espera, se registra un error y se cancela el proceso de este mensaje desencadenado.

Valores válidos

DoNotWait | OptionalWait | MandatoryWait

Interact | triggeredMessage | offerSelection

Las propiedades de configuración de esta categoría definen los valores para la selección de ofertas en los mensajes desencadenados.

maxCandidateOffers

Descripción

Esta propiedad define el número máximo de ofertas elegibles que devuelve el motor para obtener la mejor oferta para su entrega. Es posible que ninguna de las ofertas elegibles devueltas se pueda enviar, en función del canal seleccionado. Cuantas más ofertas candidatas haya, menor será la posibilidad de que esto suceda. Sin embargo, un número mayor de ofertas candidatas aumenta el tiempo de proceso.

Valores válidos

Un entero positivo

defaultCellCode

Descripción

Si la oferta entregada es el resultado de la evaluación de una regla estratégica o de un registro controlado por tablas, se asocia a la misma una celda de destino y se utiliza la información de esta celda en todos los registros relacionados. Sin embargo, si se utiliza una lista de ofertas específicas como la entrada para la selección de ofertas, no hay ninguna celda disponible. En este caso, se utiliza el valor de esta configuración. Debe asegurarse de que esta celda de destino y su campaña estén incluidas en el despliegue. El método más fácil de obtenerlo es añadir la celda en una estrategia desplegada.

Interact | triggeredMessage | dispatchers

Las propiedades de configuración de esta categoría definen los valores para todos los asignadores de los mensajes desencadenados.

dispatchingThreads

Descripción

Esta propiedad define el número de hebras que utiliza el motor en las llamadas asíncronas a los asignadores. Si el valor es 0 o un valor negativo, los asignadores se invocan de forma sincronizada. El valor predeterminado es 0.

Valores válidos

Un entero

Interact | triggeredMessage | dispatchers | <nombreAsignador>

Las propiedades de configuración de esta categoría definen los valores para un asignador específico en los mensajes desencadenados.

category name

Descripción

Esta propiedad define el nombre de este asignador. El nombre debe ser exclusivo entre todos los asignadores.

type

Descripción

Esta propiedad define el tipo del asignador.

Valores válidos

InMemoryQueue | JMSQueue | Custom

Nota: Si utiliza JMSQueue o Custom, para integrar Interact con IBM MQ, el entorno de ejecución de Interact debe estar en el servidor de aplicaciones con JDK 1.7. En el caso de WebSphere y WebLogic, se le recomienda que utilice la versión del fixpack de JDK más reciente.

JMSQueue solo da soporte a WebLogic. No puede utilizar JMSQueue si utiliza WebSphere Application Server.

className

Descripción

Esta propiedad define el nombre de clase completo de esta implementación del asignador.Si el tipo es InMemoryQueue, el valor debe estar vacío. Si el

tipo es Custom, este valor debe tener el valor `com.unicacorp.interact.eventhandler.triggeredmessage.dispatchers.IBMMQDispatcher`.

classPath

Descripción

Esta propiedad define el URL del archivo JAR que incluye la implementación de este asignador.

Si el tipo es Custom, este valor debe tener el 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 | <nombreAsignador> | Parameter Data

Las propiedades de configuración de esta categoría definen los parámetros para un asignador específico en los mensajes desencadenados.

Puede elegir entre tres tipos de asignadores. InMemoryQueue es el asignador interno para Interact. Custom se utiliza para IBM MQ. Se utiliza JMSQueue para conectar con un proveedor de JMS mediante JNDI.

category name

Descripción

Esta propiedad define el nombre de este parámetro. El nombre debe ser exclusivo entre todos los parámetros de dicho asignador.

value

Descripción

Esta propiedad define los parámetros que necesita este asignador, con el formato de pares de nombre y valor.

Nota: Todos los parámetros para los mensajes desencadenados distinguen entre mayúsculas y minúsculas y se deben especificar como se muestran aquí.

Si el tipo es InMemoryQueue, se da soporte al parámetro siguiente.

- queueCapacity: Opcional. El número máximo de ofertas que pueden esperar en la cola a ser asignadas. Si se especifica, esta propiedad debe ser un entero positivo. Si no se especifica o si no es válida, se utiliza el valor predeterminado (1000).

Si el tipo es Custom, se da soporte a los parámetros siguientes.

- providerUrl: <nombre_host>:puerto (sensible a mayúsculas y minúsculas)
- queueManager: El nombre del gestor de colas que se ha creado en IBM MQ Server.
- messageQueueName: El nombre de la cola de mensajes que se ha creado en IBM MQ Server.
- enableConsumer: Esta propiedad se debe establecer en true.

- asmUserforMQAuth: El nombre de usuario para iniciar sesión en el servidor. Es necesaria cuando el servidor aplica la autenticación. De lo contrario, no se debe especificar.
- authDS: La contraseña asociada al nombre de usuario para iniciar sesión en el servidor. Es necesaria cuando el servidor aplica la autenticación. De lo contrario, no se debe especificar.

Si el tipo es JMSQueue, se da soporte al parámetro siguiente.

- providerUrl: El URL del proveedor JNDI (sensible a mayúsculas y minúsculas).
- connectionFactoryJNDI: El nombre JNDI de la fábrica de conexiones JMS.
- messageQueueJNDI: El nombre JNDI de la cola JMS donde se envían y recuperan los mensajes desencadenados.
- enableConsumer: Si un consumidor de estos mensajes desencadenados debe iniciarse en Interact. Esta propiedad se debe establecer en true. Si no se especifica, se utiliza el valor predeterminado (false).
- initialContextFactory: El nombre completo de la clase de la fábrica de contexto inicial JNDI. Si se utiliza WebLogic, el valor de este parámetro debe ser `weblogic.jndi.WLInitialContextFactory`.

Interact | triggeredMessage | gateways | <nombrePasarela>

Las propiedades de configuración de esta categoría definen los valores para una pasarela específica en los mensajes desencadenados.

Interact no da soporte a múltiples instancias de la misma pasarela. Todos los archivos de configuración de la pasarela deben estar accesibles desde cada nodo de ejecución de Interact. En una configuración distribuida, asegúrese de que los archivos se conservan en una ubicación compartida.

category name

Descripción

Esta propiedad define el nombre de esta pasarela. Debe ser exclusivo entre todas las pasarelas.

className

Descripción

Esta propiedad define el nombre de clase completo de esta implementación de la pasarela.

classPath

Descripción

Esta propiedad define el URI del archivo JAR que incluye la implementación de esta pasarela. Si se deja en blanco, se utiliza la vía de acceso a clases de la aplicación Interact.

Por ejemplo, en un sistema Windows, si el archivo JAR de la pasarela está disponible en el directorio `C:\IBM\EMM\EmailGateway\IBM_Interact_OMO_OutboundGateway_Silverpop_1.0\lib\OMO_OutboundGateway_Silverpop.jar`, la classPath debe ser `file:///C:/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/`

OMO_OutboundGateway_Silverpop.jar. En un sistema UNIX, si el archivo jar de la pasarela está disponible en el directorio /opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar, la classpath debe ser file:///opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar.

Interact | triggeredMessage | gateways | <nombrePasarela> | Parameter Data

Las propiedades de configuración de esta categoría definen los parámetros para una pasarela específica en los mensajes desencadenados.

category name

Descripción

Esta propiedad define el nombre de este parámetro. El nombre debe ser exclusivo entre todos los parámetros de dicha pasarela.

value

Descripción

Esta propiedad define los parámetros que necesita esta pasarela, con el formato de pares de nombre y valor. Se da soporte a los parámetros siguientes para todas las pasarelas.

Nota: Todos los parámetros para los mensajes desencadenados distinguen entre mayúsculas y minúsculas y se deben especificar como se muestran aquí.

- validationTimeoutMillis: El periodo de tiempo de espera en milisegundos para la validación de una oferta mediante esta pasarela. El valor predeterminado es 500.
- deliveryTimeoutMillis: El periodo de tiempo de espera en milisegundos para la entrega de una oferta mediante esta pasarela. El valor predeterminado es 1000.

Interact | triggeredMessage | channels

Las propiedades de configuración de esta categoría definen los valores para todos los canales de los mensajes desencadenados.

type

Descripción

Esta propiedad define el nodo raíz de los valores relacionados con una pasarela específica. El valor predeterminado utiliza el selector de canales incorporado, que está basado en la lista de canales definidos en la interfaz de usuario de los mensajes desencadenados. Si se selecciona el valor predeterminado, los valores de className y classPath se deben dejar en blanco. El valor Custom utiliza la implementación del cliente de IChannelSelector.

Valores válidos

Default | Custom

className

Descripción

Esta propiedad define el nombre de clase completo de la implementación del selector de canales por parte del cliente. Este valor es necesario si el tipo es Custom.

classPath

Descripción

Esta propiedad define el URL del archivo JAR que incluye la implementación de la implementación del selector de canales por parte del cliente. Si se deja en blanco, se utiliza la vía de acceso a clases de la aplicación Interact.

Interact | triggeredMessage | channels | Parameter Data

Las propiedades de configuración de esta categoría definen los parámetros de un canal específico en los mensajes desencadenados.

category name

Descripción

Esta propiedad define el nombre de este parámetro. El nombre debe ser exclusivo entre todos los parámetros de dicho canal.

value

Descripción

Esta propiedad define los parámetros que necesita el canal, con el formato de pares de nombre y valor.

Si utiliza **Canales preferidos de cliente** para su canal, debe crear

Interact | triggeredMessage | channels | <nombreCanal>

Las propiedades de configuración de esta categoría definen los valores para un canal específico en los mensajes desencadenados.

category name

Descripción

Esta propiedad define el nombre del canal a través del cual se envían las ofertas. Debe coincidir con las definidas durante el diseño en **Campaign | partitions | <partition[N]> | Interact | outboundChannels**.

Interact | triggeredMessage | channels | <channelName> | <nombreManejador>

Las propiedades de configuración de esta categoría definen los valores para un manejador específico en los mensajes desencadenados que se utiliza para enviar las ofertas.

category name

Descripción

Esta propiedad define el nombre del manejador que utiliza el canal para enviar las ofertas.

dispatcher

Descripción

Esta propiedad define el nombre del asignador que utiliza este manejador para enviar ofertas a la pasarela. Debe ser uno de los definidos en **interact | triggeredMessage | dispatchers**.

gateway

Descripción

Esta propiedad define el nombre de la pasarela a la que este manejador envía finalmente las ofertas. Debe ser una de las definidas en **interact | triggeredMessage | gateways**.

mode

Descripción

Esta propiedad define la modalidad de uso de este manejador. Si se selecciona Failover, este manejador solo se utiliza cuando todos los manejadores que tienen propiedades más altas definidas en este canal fallan a la hora de enviar las ofertas. Si se selecciona Addon, se utiliza este manejador independientemente de si los otros manejadores han enviado correctamente las ofertas.

priority

Descripción

Esta propiedad define la prioridad de este manejador. El primer motor intenta utilizar el manejador con la prioridad más alta para el envío de ofertas.

Valores válidos

Cualquier entero

Valor predeterminado

100

Interact | ETL | patternStateETL

Las propiedades de configuración de esta categoría definen los valores para el proceso de ETL.

Nuevo nombre de categoría

Descripción

Proporcione un nombre para identificar de forma exclusiva esta configuración. Debe proporcionar exactamente el mismo nombre cuando ejecute el proceso de ETL autónomo. Para su comodidad cuando especifique este nombre en la línea de comandos, evite utilizar espacios y signos de puntuación, por ejemplo, `ETLProfile1`.

runOnceADay

Descripción

Determina si el proceso de ETL autónomo de esta configuración se debe ejecutar una vez al día. Las respuestas válidas son **Sí** o **No**. Si responde **No**, el valor **processSleepIntervalInMinutes** determina el plan de ejecución del proceso.

preferredStartTime

Descripción

Es la hora preferida en la que se debe iniciar el proceso de ETL autónomo. Especifique la hora con el formato HH:MM:SS AM/PM, por ejemplo, `01:00:00 AM`.

preferredEndTime

Descripción

Es la hora preferida en la que se debe detener el proceso de ETL autónomo. Especifique la hora con el formato HH:MM:SS AM/PM, por ejemplo, `08:00:00 AM`.

processSleepIntervalInMinutes

Descripción

Si no ha configurado el proceso de ETL autónomo para que se ejecute una vez al día (como se ha especificado en la propiedad **runOnceADay**), esta propiedad especifica el intervalo entre ejecuciones del proceso de ETL. Por ejemplo, si especifica 15 aquí, el proceso de ETL autónomo esperará 15 minutos después de detener su ejecución antes de iniciar el proceso de nuevo.

maxJDBCInsertBatchSize

Descripción

El número máximo de registros de un lote de JDBC antes de confirmar la consulta. De forma predeterminada, se establece en 5000. Tenga en cuenta que no es el número máximo de registros que ETL procesa en una iteración. Durante cada iteración, ETL procesa todos los registros disponibles en la tabla UACI_EVENTPATTERNSTATE. No obstante, todos esos registros están divididos en fragmentos de **maxJDBCInsertSize**.

maxJDBCFetchBatchSize

Descripción

El número máximo de registros de un lote de JDBC que se captan de la base de datos de preparación.

Puede que deba aumentar este valor para ajustar el rendimiento de ETL.

communicationPort

Descripción

El puerto de red donde el proceso de ETL autónomo recibe las peticiones de detención. En circunstancias normales, no debería haber ninguna razón para cambiar el valor predeterminado de esta propiedad.

queueLength

Descripción

Este valor se utiliza para ajustar el rendimiento. Se captan colecciones de datos de estado de patrón y se transforman en objetos que se añaden a una cola para procesarse y grabarse en la base de datos. Esta propiedad controla el tamaño de la cola.

completionNotificationScript

Descripción

Especifica la ruta absoluta de un script que se ejecuta cuando finaliza el proceso de ETL. Si especifica un script, se pasan tres argumentos al script de notificación de finalización: hora de inicio, hora de finalización y número total de registros de patrones de eventos procesados. La hora de inicio y la hora de finalización son valores numéricos que representan el número de milisegundos transcurridos desde 1970.

Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS

Las propiedades de configuración de esta categoría definen los valores para el DS de tiempo de ejecución de ETL.

type

Descripción

Una lista de tipos soportados de bases de datos para el origen de datos que está definiendo.

dsname

Descripción

El nombre JNDI del origen de datos. Este nombre también se debe utilizar en la configuración del origen de datos del usuario para garantizar que el usuario tenga acceso a los orígenes de datos de destino y de ejecución.

driver

Descripción

Es el nombre del controlador JDBC como, por ejemplo:

Oracle: `oracle.jdbc.OracleDriver`

Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2: `com.ibm.db2.jcc.DB2Driver`

serverURL

Descripción

Es el URL del origen de datos como, por ejemplo:

Oracle: `jdbc:oracle:thin:@ <host_base_datos>:<puerto_base_datos>:<servicio_base_datos>`

Microsoft SQL Server: `jdbc:sqlserver:// <host_base_datos>:<puerto_base_datos> ;databaseName= <nombre_base_datos>`

IBM DB2: `jdbc:db2:// <host_base_datos>:<puerto_base_datos>/ <nombre_base_datos>`

connectionpoolSize

Descripción

Un valor que indica el tamaño de la agrupación de conexiones. Se proporciona con fines de ajuste del rendimiento. Los datos de estado de patrón se leen y transforman simultáneamente dependiendo de las conexiones de base de datos disponibles. Aumentar el tamaño de la agrupación de conexiones permite más conexiones de base de datos simultáneas, supeditado a las limitaciones de memoria y capacidades de lectura/escritura de la base de datos. Por ejemplo, si este valor se establece en 4, se ejecutarán cuatro trabajos simultáneamente. Si tiene una gran cantidad de datos, puede ser necesario aumentar este valor hasta un número como 10 o 20, siempre que lo permitan la memoria y el rendimiento de la base de datos.

schema

Descripción

Es el nombre del esquema de base de datos al que se conecta esta configuración.

connectionRetryPeriod

Descripción

La propiedad `ConnectionRetryPeriod` especifica el número de segundos durante los cuales Interact reintenta automáticamente la solicitud de conexión con la base de datos cuando se produce un error. Interact intenta reconectarse automáticamente a la base de datos durante este periodo de tiempo antes de informar de un error o una anomalía de base de datos. Si el valor se establece en 0, Interact lo reintenta indefinidamente; si el valor se establece en -1, no se realiza ningún reintento.

connectionRetryDelay

Descripción

La propiedad `ConnectionRetryDelay` especifica el número de segundos que Interact espera antes de intentar reconectarse a la base de datos después de un error. Si el valor se establece en -1, no se realiza ningún reintento.

Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS

Las propiedades de configuración de esta categoría definen los valores para el DS de destino de ETL.

type

Descripción

Una lista de tipos soportados de bases de datos para el origen de datos que está definiendo.

dsname

Descripción

El nombre JNDI del origen de datos. Este nombre también se debe utilizar en la configuración del origen de datos del usuario para garantizar que el usuario tenga acceso a los orígenes de datos de destino y de ejecución.

driver

Descripción

Es el nombre del controlador JDBC como, por ejemplo:

Oracle: `oracle.jdbc.OracleDriver`

Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2: `com.ibm.db2.jcc.DB2Driver`

serverURL

Descripción

Es el URL del origen de datos como, por ejemplo:

Oracle: `jdbc:oracle:thin:@ <host_base_datos>:<puerto_base_datos>:<servicio_base_datos>`

Microsoft SQL Server: `jdbc:sqlserver:// <host_base_datos>:<puerto_base_datos> ;databaseName= <nombre_base_datos>`

IBM DB2: `jdbc:db2:// <host_base_datos>:<puerto_base_datos>/ <nombre_base_datos>`

connectionpoolSize

Descripción

Un valor que indica el tamaño de la agrupación de conexiones. Se proporciona con fines de ajuste del rendimiento. Los datos de estado de patrón se leen y transforman simultáneamente dependiendo de las conexiones de base de datos disponibles. Aumentar el tamaño de la agrupación de conexiones permite más conexiones de base de datos simultáneas, supeditado a las limitaciones de memoria y capacidades de lectura/escritura de la base de datos. Por ejemplo, si este valor se establece en 4, se ejecutarán cuatro trabajos simultáneamente. Si tiene una gran cantidad de datos, puede ser necesario aumentar este valor hasta un número como 10 o 20, siempre que lo permitan la memoria y el rendimiento de la base de datos.

schema

Descripción

Es el nombre del esquema de base de datos al que se conecta esta configuración.

connectionRetryPeriod

Descripción

La propiedad `ConnectionRetryPeriod` especifica el número de segundos durante los cuales Interact reintenta automáticamente la solicitud de conexión con la base de datos cuando se produce un error. Interact intenta reconectarse automáticamente a la base de datos durante este periodo de tiempo antes de informar de un error o una anomalía de base de datos. Si

el valor se establece en 0, Interact lo reintenta indefinidamente; si el valor se establece en -1, no se realiza ningún reintento.

connectionRetryDelay

Descripción

La propiedad `ConnectionRetryDelay` especifica el número de segundos que Interact espera antes de intentar reconectarse a la base de datos después de un error. Si el valor se establece en -1, no se realiza ningún reintento.

Interact | ETL | patternStateETL | <patternStateETLName> | Report

Las propiedades de configuración de esta categoría definen los valores para el proceso de agregación de informes de ETL.

enable

Descripción

Habilita o inhabilita la integración de los informes con ETL. Esta propiedad está establecida en disable de forma predeterminada.

Si se establece en `disable`, esta propiedad inhabilita las actualizaciones en la tabla UARI_DELTA_PATTERNS. No inhabilita completamente la creación de informes.

Nota: Para inhabilitar la integración de informes con ETL, también debe modificar el desencadenante TR_AGGREGATE_DELTA_PATTERNS para inhabilitarlo en la tabla de preparación UACI_ETLPATTERNSTATERUN.

retryAttemptsIfAggregationRunning

Descripción

El número de veces que el ETL intenta comprobar si se ha completado la agregación de informes si se ha establecido el indicador de bloqueo. Esta propiedad está establecida en 3 de forma predeterminada.

sleepBeforeRetryDurationInMinutes

Descripción

Tiempo de inactividad en minutos entre dos intentos consecutivos. Esta propiedad está establecida en 5 de forma predeterminada.

aggregationRunningCheckSql

Descripción

Esta propiedad permite definir un SQL personalizado, que puede ejecutarse para ver si se ha establecido el indicador de bloqueo de la agregación de informe. De forma predeterminada, esta propiedad está vacía.

Cuando no se establece esta propiedad, el ETL ejecuta el siguiente SQL para obtener el indicador de bloqueo.

```
select count(1) AS ACTIVERUNS from uari_pattern_lock where islock='Y'
=> If ACTIVERUNS is > 0, lock is set
```

aggregationRunningCheck

Descripción

Habilita o inhabilita la comprobación de la ejecución de la agregación de

informe antes de que se realice la ejecución del ETL. Esta propiedad está establecida en enable de forma predeterminada.

Apéndice C. Propiedades de configuración del entorno de diseño de Interact

Esta sección describe todas las propiedades de configuración del entorno de diseño de Interact.

Campaña | particiones | partición[n] | informes

La propiedad **Campaign | partitions | partition[n] | reports** define los distintos tipos de carpetas para los informes.

offerAnalysisTabCachedFolder

Descripción

La propiedad `offerAnalysisTabCachedFolder` especifica la ubicación de la carpeta que contiene la especificación de los informes de oferta expandidos listados en la pestaña Análisis cuando accede a ella pulsando en el enlace Análisis del panel de navegación. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

Descripción

La propiedad `segmentAnalysisTabOnDemandFolder` especifica la ubicación de la carpeta que contiene los informes de segmento listados en la pestaña Análisis de un segmento. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

Descripción

La propiedad `offerAnalysisTabOnDemandFolder` especifica la ubicación de la carpeta que contiene los informes de oferta listados en la pestaña Análisis de una oferta. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

segmentAnalysisTabCachedFolder

Descripción

La propiedad `segmentAnalysisTabCachedFolder` especifica la ubicación de la carpeta que contiene la especificación de los informes de segmentos

expandidos listados en la pestaña Análisis cuando accede a ella pulsando en el enlace Análisis del panel de navegación. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']
```

analysisSectionFolder

Descripción

La propiedad `analysisSectionFolder` especifica la ubicación de la carpeta raíz en la que se almacenan las especificaciones de informes. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign']
```

campaignAnalysisTabOnDemandFolder

Descripción

La propiedad `campaignAnalysisTabOnDemandFolder` especifica la ubicación de la carpeta que contiene los informes de campaña listados en la pestaña Análisis de un segmento. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']
```

campaignAnalysisTabCachedFolder

Descripción

La propiedad `campaignAnalysisTabCachedFolder` especifica la ubicación de la carpeta que contiene la especificación de los informes de campaña expandidos listados en la pestaña Análisis cuando accede a ella pulsando en el enlace Análisis del panel de navegación. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']/folder[@name='cached']
```

campaignAnalysisTabEmessageOnDemandFolder

Descripción

La propiedad `campaignAnalysisTabEmessageOnDemandFolder` especifica la ubicación de la carpeta que contiene los informes de eMessage listados en la pestaña Análisis de un segmento. La ruta se especifica usando la notación XPath.

Valor predeterminado

```
/content/folder[@name='Affinium Campaign']/folder[@name='eMessage
Reports']
```

campaignAnalysisTabInteractOnDemandFolder

Descripción

Cadena de carpeta del servidor de informes para los informes de Interact.

Valor predeterminado

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

Disponibilidad

Esta propiedad solo se aplica si se ha instalado Interact.

interactiveChannelAnalysisTabOnDemandFolder

Descripción

La cadena de la carpeta del servidor de informes para los informes de la pestaña Análisis del canal interactivo.

Valor predeterminado

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

Disponibilidad

Esta propiedad solo se aplica si se ha instalado Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking

Estas propiedades de configuración definen los valores del módulo del historial de contactos y respuestas de Interact.

isEnabled

Descripción

Si se establece en `yes`, habilita el módulo del historial de contactos y respuestas de Interact, que copia el historial de contactos y respuestas de Interact desde las tablas de preparación del tiempo de ejecución de Interact en las tablas de historial de contactos y respuestas de Campaign. La propiedad `interactInstalled` también debe establecerse en `yes`.

Valor predeterminado

no

Valores válidos

yes | no

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

runOnceADay

Descripción

Especifica si se debe ejecutar el ETL del historial de contactos y respuestas una vez al día. Si establece esta propiedad en `Yes`, el ETL se ejecuta durante el intervalo planificado especificado por `preferredStartTime` y `preferredEndTime`.

Si el ETL tarda más de 24 horas en ejecutarse y, por lo tanto, no llega a la hora de inicio del día siguiente, omitirá ese día y se ejecutará a la hora planificada del día siguiente. Por ejemplo, si ETL está configurado para ejecutarse entre la 1:00 a.m. y las 3:00 a.m., y el proceso se inicia a la 1:00 a.m. del lunes y termina a las 2:00 a.m. del martes, la siguiente ejecución, planificada originalmente para la 1:00 a.m. del martes, se omitirá y el siguiente ETL empezará a la 1:00 a.m. del miércoles.

La planificación de ETL no tiene en cuenta los cambios del horario de verano. Por ejemplo, si el ETL está planificado para ejecutarse entre la 1:00 a.m. y las 3:00 a.m., puede ejecutarse a las 12:00 a.m. o las 2:00 a.m. cuando se produce el cambio de horario de verano.

Valor predeterminado

No

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

processSleepIntervalInMinutes

Descripción

El número de minutos que espera el módulo del historial de contactos y respuestas de Interact entre la copia de los datos de las tablas de preparación de tiempo de ejecución de Interact en las tablas del historial de contactos y respuestas de Campaign.

Valor predeterminado

60

Valores válidos

Un entero mayor que cero.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

preferredStartTime

Descripción

La hora preferida para iniciar el proceso de ETL diario. Esta propiedad, cuando se utiliza con la propiedad preferredEndTime, configura el intervalo de tiempo preferido en el que desea ejecutar el ETL. El ETL se iniciará durante el intervalo de tiempo especificado y procesará como máximo el número de registros especificado utilizando `maxJDBCFetchBatchSize`. El formato es HH:mm:ss AM o PM, utilizando un reloj de 12 horas.

Valor predeterminado

12:00:00 AM

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

preferredEndTime

Descripción

La hora preferida para finalizar el proceso de ETL diario. Esta propiedad, cuando se utiliza con la propiedad preferredStartTime, configura el intervalo de tiempo preferido en el que desea ejecutar el ETL. El ETL se iniciará durante el intervalo de tiempo especificado y procesará como máximo el número de registros especificado utilizando `maxJDBCFetchBatchSize`. El formato es HH:mm:ss AM o PM, utilizando un reloj de 12 horas.

Valor predeterminado

2:00:00 AM

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

purgeOrphanResponseThresholdInMinutes

Descripción

El número de minutos que espera el módulo del historial de contactos y respuestas de Interact antes de depurar las respuestas sin un contacto correspondiente. Esto evita que se registren respuestas sin que se registren contactos.

Valor predeterminado

180

Valores válidos

Un entero mayor que cero.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

maxJDBCInsertBatchSize

Descripción

El número máximo de registros de un lote de JDBC antes de confirmar la consulta. No es el número máximo de registros que procesa el módulo del historial de contactos y respuestas de Interact en una iteración. Durante cada iteración, el módulo del historial de contactos y respuestas de Interact procesa todos los registros disponibles de las tablas de preparación. No obstante, todos esos registros están divididos en fragmentos de `maxJDBCInsertSize`.

Valor predeterminado

1000

Valores válidos

Un entero mayor que cero.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

maxJDBCFetchBatchSize

Descripción

El número máximo de registros de un lote de JDBC que se captan de la base de datos de preparación. Es probable que deba aumentar este valor para ajustar el rendimiento del módulo del historial de contactos y respuestas.

Por ejemplo, para procesar 2,5 millones de registros del historial de contactos al día, debe establecer `maxJDBCFetchBatchSize` en un número mayor que 2.500.000 para que se procesen todos los registros de un día.

A continuación, puede establecer `maxJDBCFetchChunkSize` y `maxJDBCInsertBatchSize` en valores menores (en este ejemplo, quizá 50.000 y 10.000, respectivamente). También se podrán procesar algunos registros del día siguiente, pero se retendrán hasta el día siguiente.

Valor predeterminado

1000

Valores válidos

Un entero mayor que cero

maxJDBCFetchChunkSize

Descripción

El número máximo de un tamaño de fragmento de JDBC de lectura de datos durante el proceso de ETL (Extracción, Transformación y Carga). En algunos casos, un tamaño de fragmento mayor que el tamaño de inserción puede mejorar la velocidad del proceso de ETL.

Valor predeterminado

1000

Valores válidos

Un entero mayor que cero

deleteProcessedRecords

Descripción

Especifica si se retienen los registros del historial de contactos y el historial de respuestas una vez procesados.

Valor predeterminado

Sí

completionNotificationScript

Descripción

Especifica la ruta absoluta de un script que se ejecuta cuando finaliza el ETL. Si especifica un script, se pasan cinco argumentos al script de notificación de finalización: hora de inicio, hora de finalización, el número total de registros CH procesados, el número total de registros RH procesados y el estado. La hora de inicio y la hora de finalización son valores numéricos que representan el número de milisegundos transcurridos desde 1970. El argumento de estado indica si el trabajo ETL

se ha realizado correctamente o no. 0 indica un trabajo ETL correcto. 1 indica un error y que hay algunos errores en el trabajo ETL.

Valor predeterminado

Ninguno

fetchSize

Descripción

Permite establecer JDBC fetchSize cuando recupera los registros de las tablas de preparación.

En las bases de datos Oracle especialmente, ajuste el valor en el número de registros que JDBC debe recuperar con cada viaje de ida y vuelta de red. Para lotes grandes de 100K o más, intente con 10000. Tenga cuidado de no utilizar un valor demasiado grande, ya que tendrá un impacto en el uso de la memoria y las ventajas serán insignificantes, si no negativas.

Valor predeterminado

Ninguno

daysBackInHistoryToLookupContact

Descripción

Limita los registros que se buscan durante las consultas de historial de respuestas a las que se encuentran dentro del número de días en el pasado especificados. Para bases de datos con una gran cantidad de registros de historial de repuestas, esto puede reducir el tiempo de proceso en las consultas limitando el periodo de búsqueda al número de días especificado.

El valor predeterminado de 0 indica que se buscan todos los registros.

Valor predeterminado

0 (cero)

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

Estas propiedades de configuración definen el origen de datos del módulo del historial de contactos y respuestas de Interact.

jndiName

Descripción

Utilice la propiedad `systemTablesDataSource` para identificar el origen de datos JNDI (Java Naming and Directory Interface) que está definido en el servidor de aplicaciones (WebSphere o WebLogic) para las tablas de ejecución de Interact.

La base de datos de tiempo de ejecución de Interact es la base de datos que se completa con los scripts dll `aci_runtime` y `aci_populate_runtime` y, por ejemplo, contiene las siguientes tablas (entre otras): `UACI_CHOfferAttrib` y `UACI_DefaultedStat`.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

databaseType

Descripción

Tipo de base de datos para el origen de datos de tiempo de ejecución de Interact.

Valor predeterminado

SQLServer

Valores válidos

SQLServer | Oracle | DB2

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

schemaName

Descripción

El nombre del esquema que contiene las tablas de preparación del módulo de historial de contactos y respuestas. Debería ser el mismo que el de las tablas del entorno de ejecución.

No tiene que definir un esquema.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

Estas propiedades de configuración definen el tipo de contacto de la campaña que se correlaciona con un 'contact' para la creación de informes o el aprendizaje.

contacted

Descripción

El valor asignado a la columna `ContactStatusID` de la tabla `UA_DtlContactHist` de las tablas del sistema de Campaign para un contacto de oferta. El valor debe ser una entrada válida de la tabla `UA_ContactStatus`. Consulte la publicación *Campaign Guía del administrador* para obtener detalles sobre cómo añadir tipos de contacto.

Valor predeterminado

2

Valores válidos

Un entero mayor que cero.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

Estas propiedades de configuración definen las respuestas para aceptar o rechazar los informes y el aprendizaje.

accept

Descripción

El valor asignado a la columna `ResponseTypeID` de la tabla `UA_ResponseHistory` en las tablas del sistema de Campaign para una oferta aceptada. El valor debe ser una entrada válida en la tabla `UA_UsrResponseType`. Debe asignar a la columna `CountsAsResponse` el valor `1`, una respuesta.

Consulte la publicación *Campaign Administrator's Guide* para obtener información detallada sobre cómo añadir tipos de respuesta.

Valor predeterminado

3

Valores válidos

Un entero mayor que cero.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

reject

Descripción

El valor asignado a la columna `ResponseTypeID` de la tabla `UA_ResponseHistory` en las tablas del sistema de Campaign para una oferta rechazada. El valor debe ser una entrada válida en la tabla `UA_UsrResponseType`. Debe asignar a la columna `CountsAsResponse` el valor `2`, un rechazo. Consulte la publicación *Campaign Administrator's Guide* para obtener información detallada sobre cómo añadir tipos de respuesta.

Valor predeterminado

8

Valores válidos

Un entero mayor que cero.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | report

Estas propiedades de configuración definen los nombres de informe cuando se realiza la integración con Cognos.

interactiveCellPerformanceByOfferReportName

Descripción

Nombre del informe Rendimiento de celda interactiva por oferta. Este nombre debe coincidir con el nombre de este informe en el servidor Cognos.

Valor predeterminado

Rendimiento de celda interactiva por oferta

treatmentRuleInventoryReportName

Descripción

Nombre del informe Inventario de reglas de tratamiento. Este nombre debe coincidir con el nombre de este informe en el servidor Cognos.

Valor predeterminado

Inventario de reglas de tratamiento de canal

deploymentHistoryReportName

Descripción

Nombre del informe Historial de despliegues. Este nombre debe coincidir con el nombre de este informe en el servidor Cognos.

Valor predeterminado

Historial de despliegues de canal

Campaign | partitions | partition[n] | Interact | learning

Estas propiedades de configuración permiten ajustar el módulo de aprendizaje incorporado.

confidenceLevel

Descripción

Un porcentaje que indica la confianza que se desea que tenga la utilidad de aprendizaje antes de cambiar de la exploración a la explotación. Un valor 0 desactiva efectivamente la exploración.

Esta propiedad sólo es aplicable si la propiedad `Interact > offerserving > optimizationType` del tiempo de ejecución de Interact se establece en `BuiltInLearning`.

Valor predeterminado

95

Valores válidos

Un entero entre 0 y 95, divisible por 5 o 99.

validateonDeployment

Descripción

Si está establecido en `No`, Interact no valida el módulo de aprendizaje al desplegar. Si está establecido en `yes`, Interact valida el módulo de aprendizaje al desplegar.

Valor predeterminado

`No`

Valores válidos

Yes | No

maxAttributeNames

Descripción

El número máximo de atributos de aprendizaje que supervisa la utilidad de aprendizaje de Interact.

Esta propiedad sólo es aplicable si la propiedad `Interact > offerserving > optimizationType` del tiempo de ejecución de Interact se establece en `BuiltInLearning`.

Valor predeterminado

`10`

Valores válidos

Un entero cualquiera.

maxAttributeValues

Descripción

El número máximo de valores para los que el módulo de aprendizaje de Interact realiza un seguimiento para cada atributo de aprendizaje.

Esta propiedad sólo es aplicable si la propiedad `Interact > offerserving > optimizationType` del tiempo de ejecución de Interact se establece en `BuiltInLearning`.

Valor predeterminado

5

otherAttributeValue

Descripción

El nombre predeterminado del valor de atributo que se utiliza para representar todos los valores de atributo más allá de `maxAttributeValues`.

Esta propiedad sólo es aplicable si la propiedad `Interact > offerserving > optimizationType` del tiempo de ejecución de Interact se establece en `BuiltInLearning`.

Valor predeterminado

`Other`

Valores válidos

Una cadena o un número.

Ejemplo

Si maxAttributeValues se establece en 3 y otherAttributeValue se establece en other, el módulo de aprendizaje realiza un seguimiento de los tres primeros valores. Los demás valores se asignan a la otra categoría. Por ejemplo, si realiza un seguimiento del atributo de visitante de color de cabello, y los cinco primeros visitantes tienen el cabello de color moreno, castaño, rubio, pelirrojo y gris, la utilidad de aprendizaje realiza un seguimiento de los colores de cabello moreno, castaño y rubio. Los colores pelirrojo y gris se agrupan en el `otherAttributeValue`, otros.

percentRandomSelection

Descripción

El porcentaje de tiempo que el módulo de aprendizaje presenta una oferta aleatoria. Por ejemplo, si percentRandomSelection se establece en 5, esto significa que el módulo de aprendizaje presenta una oferta aleatoria el 5% del tiempo (5 de cada 100 recomendaciones), independentemente de la puntuación. Habilitar `percentRandomSelection` altera temporalmente la propiedad de configuración `offerTieBreakMethod`. Cuando `percentRandomSelection` está habilitado, esta propiedad está establecida independientemente de si la formación está activada o desactivada o si se utiliza el aprendizaje incorporado o externo.

Valor predeterminado

5

Valores válidos

Un entero del 0 (que inhabilitaa la característica `percentRandomSelection`) al 100.

recencyWeightingFactor

Descripción

La representación decimal de un porcentaje del conjunto de datos definido por `recencyWeightingPeriod`. Por ejemplo, el valor predeterminado 0,15 significa que el 15% de los datos utilizados por la utilidad de aprendizaje provienen de `recencyWeightingPeriod`.

Esta propiedad sólo es aplicable si la propiedad `Interact > offerserving > optimizationType` del tiempo de ejecución de Interact se establece en `BuiltInLearning`.

Valor predeterminado

0,15

Valores válidos

Un valor decimal menor que 1.

recencyWeightingPeriod

Descripción

El tamaño en horas de datos a los que el módulo de aprendizaje ha otorgado el porcentaje `recencyWeightingFactor` de ponderación. Por ejemplo, el valor predeterminado 120 significa que el `recencyWeightingFactor` de los datos utilizados por el módulo de aprendizaje provienen de las últimas 120 horas.

Esta propiedad sólo es aplicable si `optimizationType` se ha establecido en `builtInLearning`.

Valor predeterminado

120

minPresentCountThreshold

Descripción

El número mínimo de veces que debe presentarse una oferta antes de que sus datos se utilicen en los cálculos y el módulo de aprendizaje entre en el modo de exploración.

Valor predeterminado

0

Valores válidos

Un entero mayor o igual que cero.

enablePruning

Descripción

Si se establece en `Yes`, el módulo de aprendizaje de Interact determina con algoritmos cuándo un atributo de aprendizaje (estándar o dinámico) no es predictivo. Si un atributo de aprendizaje no es predictivo, el módulo de aprendizaje no lo considerará cuando determine la ponderación de una oferta. Esto continúa hasta que el módulo de aprendizaje agregue datos de aprendizaje.

Si se establece en `No`, el módulo de aprendizaje siempre utiliza todos los atributos de aprendizaje. Si no se podan los atributos no predictivos, el módulo de aprendizaje no tendrá la precisión que debería.

Valor predeterminado

Sí

Valores válidos

Yes | No

Campaign | partitions | partition[n] | Interact | learning | learningAttributes | [learningAttribute]

Estas propiedades de configuración definen los atributos de aprendizaje.

attributeName

Descripción

Cada `attributeName` es el nombre de un atributo de visitante que se desea supervisar con el módulo de aprendizaje. Debe coincidir con el nombre de un par nombre-valor en los datos de sesión.

Esta propiedad sólo es aplicable si la propiedad `Interact > offerserving > optimizationType` del tiempo de ejecución de Interact se establece en `BuiltInLearning`.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Campaign | partitions | partition[n] | Interact | deployment

Estas propiedades de configuración definen los valores de despliegue.

chunkSize

Descripción

El tamaño máximo de fragmentación en KB para cada paquete de despliegue de Interact.

Valor predeterminado

500

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup]

Estas propiedades de configuración definen los valores del grupo de servidores.

serverGroupName

Descripción

El nombre del grupo de servidores de ejecución de Interact. Es el nombre que aparece en la pestaña Resumen del canal interactivo.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

Estas propiedades de configuración definen los servidores de ejecución de Interact.

instanceURL

Descripción

El URL del servidor de ejecución de Interact. Un grupo de servidores puede contener varios servidores de ejecución de Interact; no obstante, cada servidor se debe crear con una nueva categoría.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Ejemplo

`http://server:port/interact`

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | flowchart

Estas propiedades de configuración definen el entorno de ejecución de Interact utilizado para las ejecuciones de prueba de diagramas de flujo interactivos.

serverGroup

Descripción

El nombre del grupo de servidores de Interact que Campaign utiliza para realizar una ejecución de prueba. Este nombre debe coincidir con el nombre de categoría que crea en `serverGroups`.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

dataSource

Descripción

Utilice la propiedad `dataSource` para identificar el origen de datos físico que Campaign utiliza cuando realiza las ejecuciones de prueba de diagramas de flujo interactivos. Esta propiedad debe coincidir con el origen de datos definido por la propiedad `Campaign > partitions > partitionN > dataSources` del origen de datos de ejecución de prueba definido para el tiempo de diseño de Interact.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

eventPatternPrefix

Descripción

La propiedad `eventPatternPrefix` es un valor de cadena que se añade como prefijo a los nombres de patrón de evento para permitir su utilización en expresiones en procesos Selección o Decisión de diagramas de flujo interactivos.

Tenga en cuenta que si cambia este valor, debe desplegar cambios globales en el canal interactivo para que esta configuración actualizada se aplique.

Valor predeterminado

`EventPattern`

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

Estas propiedades de configuración definen el código de celda predeterminado para la tabla de ofertas predeterminadas. Sólo debe configurar estas propiedades si está definiendo asignaciones de oferta globales.

DefaultCellCode

Descripción

El código de celda predeterminado que Interact utiliza si no define un código de celda en la tabla de ofertas predeterminadas.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Valores válidos

Una cadena que coincide con el formato de código de celda definido en Campaign

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

Estas propiedades de configuración definen el código de celda predeterminado para la tabla offersBySQL. Sólo debe configurar estas propiedades si utiliza consultas SQL para obtener el conjunto deseado de ofertas candidatas.

DefaultCellCode

Descripción

El código de celda predeterminado que Interact utiliza para una oferta en las tablas OffersBySQL que tiene un valor nulo en la columna de código de celda (o si la columna de código de celda no aparece en absoluto). El valor debe ser un código de celda válido.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Valores válidos

Una cadena que coincide con el formato de código de celda definido en Campaign

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

Estas propiedades de configuración definen el código de celda predeterminado para la tabla de anulación de puntuaciones. Sólo debe configurar estas propiedades si está definiendo asignaciones de oferta individuales.

DefaultCellCode

Descripción

El código de celda predeterminado que Interact utiliza si no define un código de celda en la tabla de anulación de puntuaciones.

Valor predeterminado

No se ha definido ningún valor predeterminado.

Valores válidos

Una cadena que coincide con el formato de código de celda definido en Campaign

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | server | internal

Las propiedades de esta categoría especifican valores de integración y los límites del ID interno de la partición de Campaign seleccionada. Si su instalación de Campaign tiene varias particiones, establezca estas propiedades para cada partición que desee que se vea afectada.

internalIdLowerLimit

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Las propiedades internalIdUpperLimit y internalIdLowerLimit restringen los ID internos de Campaign dentro del rango especificado. Tenga en cuenta que los valores son inclusivos: es decir, Campaign puede utilizar tanto el límite inferior como el superior.

Valor predeterminado

0 (cero)

internalIdUpperLimit

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Las propiedades internalIdUpperLimit y internalIdLowerLimit restringen los ID internos de Campaign dentro del rango especificado. Los valores son inclusivos: es decir, Campaign puede utilizar tanto el límite inferior como el superior. Si está instalado Distributed Marketing, establezca el valor en 2147483647.

Valor predeterminado

4294967295

eMessageInstalled

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Indica que eMessage está instalado. Si selecciona Yes, las características de eMessage están disponibles en la interfaz de Campaign.

El instalador de IBM define esta propiedad en Yes para la partición predeterminada de la instalación de eMessage. Si tiene particiones adicionales donde haya instalado eMessage, debe configurar esta propiedad manualmente.

Valor predeterminado

No

Valores válidos

Yes | No

interactInstalled

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Tras instalar el entorno de diseño de Interact, esta propiedad de configuración se debe establecer en Yes para habilitar el entorno de diseño de Interact en Campaign.

Si Interact no está instalado, establézcala en No. Si se establece esta propiedad en No no se eliminan los menús y las opciones de Interact de la interfaz de usuario. Para eliminar los menús y las opciones, debe anular manualmente el registro de Interact utilizando el programa de utilidad configTool.

Valor predeterminado

No

Valores válidos

Yes | No

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

MO_UC_integration

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Habilita la integración con Marketing Operations para esta partición, si la integración está habilitada en los valores de configuración de **Platform**. Si desea más información, consulte *IBM Marketing Operations y Campaign Integration Guide*.

Valor predeterminado

No

Valores válidos

Yes | No

MO_UC_BottomUpTargetCells

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Para esta partición, permite células de modo ascendente en hojas de cálculo de celdas objetivo, si **MO_UC_integration** está habilitado. Si se establece en Yes, están visibles tanto las celdas objetivo de modo descendente como las de modo ascendente, pero las celdas objetivo de modo ascendente son de sólo lectura. Si desea más información, consulte *IBM Marketing Operations y Campaign Integration Guide*.

Valor predeterminado

No

Valores válidos

Yes | No

Legacy_campaigns

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Para esta partición, habilita el acceso a campañas creadas antes de integrar Marketing Operations y Campaign. Sólo se aplica si **MO_UC_integration** está establecido en Yes. Las campañas heredadas también incluyen campañas creadas en Campaign 7.x y enlazadas a proyectos de Plan 7.x. Si desea más información, consulte *IBM Marketing Operations y Campaign Integration Guide*.

Valor predeterminado

No

Valores válidos

Yes | No

IBM Marketing Operations - Integración de ofertas

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Habilita la posibilidad de utilizar Marketing Operations para realizar tareas de gestión del ciclo de vida de las ofertas en esta partición, si **MO_UC_integration** está habilitado para esta partición. En los valores de configuración de **Platform**, la integración de ofertas debe estar habilitada. Si desea más información, consulte *IBM Marketing Operations y Campaign Integration Guide*.

Valor predeterminado

No

Valores válidos

Yes | No

UC_CM_integration

Categoría de configuración

Campaign|partitions|partition[n]|server|internal

Descripción

Habilita la integración de segmentos en línea de Digital Analytics para una partición de Campaign. Si establece este valor en Yes, el cuadro del proceso Selección en un diagrama de flujo proporciona la opción de seleccionar **Segmentos de Digital Analytics**. Para configurar la integración de Digital Analytics para cada partición, elija **Valores > Configuración > Campaign | particiones | partición[n] | Coremetrics**.

Valor predeterminado

No

Valores válidos

Yes | No

numRowsReadToParseDelimitedFile

Categoría de configuración

`Campaign|partitions|partition[n]|server|internal`

Descripción

Esta propiedad se utiliza cuando se correlaciona un archivo delimitado como una tabla de usuario. También la utiliza el cuadro de proceso Puntuación al importar un archivo de salida de puntuación desde IBM SPSS Modeler Advantage Enterprise Marketing Management Edition. Para importar o correlacionar un archivo delimitado, Campaign deberá analizar el archivo para identificar las columnas, tipos de datos (tipos de campo) y anchos de columna (longitudes de campo).

El valor predeterminado de 100 significa que Campaign examina las primeras 50 y las últimas 50 entradas de línea en el archivo delimitado. A continuación, Campaign asigna la longitud de campo basándose en el valor más grande encontrado en estas entradas. En la mayoría de los casos, el valor predeterminado es suficiente para determinar las longitudes de campo. Sin embargo, en archivos delimitados muy grandes, un campo posterior podría exceder la longitud estimada que calcula Campaign, que puede provocar un error durante el tiempo de ejecución del diagrama de flujo. Por lo tanto, si está correlacionando un archivo muy grande, puede aumentar este valor para que Campaign examine más entradas de línea. Por ejemplo, un valor de 200 consigue que Campaign examine las primeras 100 entradas de línea y las últimas 100 entradas de línea del archivo.

Un valor de 0 examina todo el archivo. Por regla general, esto solo es necesario si importa o correlaciona archivos que tienen anchos de datos variables de campos que no se pueden identificar leyendo las primeras y las últimas líneas. La lectura de todo el archivo, en casos de archivos extremadamente grandes, puede aumentar el tiempo de proceso necesario para la correlación de tablas y las ejecuciones del cuadro de proceso Puntuación.

Valor predeterminado

100

Valores válidos

0 (todas las líneas) o cualquier entero positivo

Campaign | supervisión

Las propiedades de esta categoría especifican si la característica de supervisión operativa está habilitada, el URL del servidor de Supervisión operativa y el comportamiento de la memoria caché. La Supervisión operativa le muestra y le permite controlar diagramas de flujo activos.

cacheCleanupInterval

Descripción

La propiedad `cacheCleanupInterval` especifica el intervalo, en segundos, entre las limpiezas automáticas de la memoria caché del estado de diagrama de flujo.

Esta propiedad no está disponible en versiones de Campaign anteriores a la 7.0.

Valor predeterminado

600 (10 minutos)

cacheRunCompleteTime

Descripción

La propiedad cacheRunCompleteTime especifica el tiempo, en minutos, que las ejecuciones completadas se mantienen en caché y se muestran en la página Supervisión.

Esta propiedad no está disponible en versiones de Campaign anteriores a la 7.0.

Valor predeterminado

4320

monitorEnabled

Descripción

La propiedad monitorEnabled especifica si el supervisor está activado.

Esta propiedad no está disponible en versiones de Campaign anteriores a la 7.0.

Valor predeterminado

FALSE

Valores válidos

TRUE | FALSE

serverURL

Descripción

La propiedad Campaign > supervisión > serverURL especifica el URL del servidor de Supervisión operativa. Esta valor es obligatorio; modifique el valor si el URL del servidor Supervisión operativa no es el valor predeterminado.

Si Campaign está configurado para utilizar comunicaciones a través de la capa de sockets seguros (SSL), establezca el valor de esta propiedad para que utilice HTTPS. Por ejemplo: serverURL=https://*host:puerto_SSL*/Campaign/OperationMonitor donde:

- *host* es el nombre o dirección IP de la máquina en la que está instalada la aplicación web.
- *Puerto_SSL* es el puerto SSL de la aplicación web.

Observe la cadena https en el URL.

Valor predeterminado

http://localhost:7001/Campaign/OperationMonitor

monitorEnabledForInteract

Descripción

Si se establece en TRUE, habilita el servidor del conector JMX de Campaign para Interact. Campaign no tiene seguridad JMX.

Si se establece en FALSE, no se puede conectar con el servidor del conector JMX de Campaign.

Esta supervisión de JMX es únicamente para el módulo de historial de respuestas y contactos de Interact.

Valor predeterminado

FALSE

Valores válidos

TRUE | FALSE

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

protocol

Descripción

Protocolo de escucha para el servidor del conector JMX de Campaign, si monitorEnabledForInteract se establece en yes.

Esta supervisión de JMX es únicamente para el módulo de historial de respuestas y contactos de Interact.

Valor predeterminado

JMXMP

Valores válidos

JMXMP | RMI

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

port

Descripción

Puerto de escucha para el servidor del conector JMX de Campaign, si monitorEnabledForInteract se establece en yes.

Esta supervisión de JMX es únicamente para el módulo de historial de respuestas y contactos de Interact.

Valor predeterminado

2004

Valores válidos

Un número entero entre 1025 y 65535.

Disponibilidad

Esta propiedad sólo es aplicable si tiene instalado Interact.

Campaign | partitions | partition[n] | Interact | outboundChannels

Estas propiedades de configuración permiten ajustar los canales de salida para los mensajes desencadenados.

category name

Descripción

Esta propiedad define el nombre de este canal de salida. El nombre debe ser exclusivo entre todos los canales de salida.

name

Descripción

El nombre del canal de salida.

Nota: Debe reiniciar el servidor de aplicaciones para que los cambios entren en vigor.

Campaign | partitions | partition[n] | Interact | outboundChannels | Parameter Data

Estas propiedades de configuración permiten ajustar los canales de salida para los mensajes desencadenados.

category name

Descripción

Esta propiedad define el nombre de este parámetro. El nombre debe ser exclusivo entre todos los parámetros de dicho canal de salida.

value

Descripción

Esta propiedad define los parámetros que necesita esta pasarela de salida, con el formato de pares de nombre y valor.

Apéndice D. Personalización de ofertas en tiempo real en el lado del cliente

Puede haber situaciones en las que desee proporcionar personalización de ofertas en tiempo real sin implementar código Java de bajo nivel o llamadas SOAP al servidor Interact. Por ejemplo, cuando un visitante carga inicialmente una página web donde el contenido de Javascript es la única programación ampliada disponible, o cuando un visitante abre un mensaje de correo electrónico donde sólo es posible contenido HTML. IBM Interact proporciona varios conectores que proporcionan gestión de ofertas en tiempo real en situaciones en las que solo se tiene control sobre el contenido web que se carga en el lado del cliente, o en las que se desea simplificar la interfaz a Interact.

La instalación de Interact incluye dos conectores para personalización de ofertas iniciada en el lado del cliente:

- "Acerca de Interact Message Connector". Mediante Message Connector, el contenido web de los mensajes de correo electrónico (por ejemplo) u otro soporte electrónico puede contener etiquetas de enlaces e imágenes para realizar llamadas al servidor Interact para presentación de ofertas de carga de página o páginas de destino de pulsación.
- "Acerca de Interact Web Connector" en la página 314. Mediante Web Connector (también denominado JS Connector), las páginas web pueden utilizar JavaScript del lado del cliente para gestionar el arbitraje, presentación e historial de contactos/respuestas de ofertas en la presentación de ofertas al cargar página o las páginas de destino de pulsación.

Acerca de Interact Message Connector

Interact Message Connector permite que los mensajes de correo electrónico y otros soportes electrónicos realicen llamadas a IBM Interact para permitir que se presenten ofertas personalizadas cuando se abra o el cliente pulse el mensaje y vaya al sitio especificado. Esto se consigue mediante la utilización de dos etiquetas clave: la etiqueta de imagen (`IMG`), que carga las ofertas personalizadas al abrir, o la etiqueta de enlace (`A`), que captura información cuando el cliente pulsa y lo redirige a una página de destino específica.

Ejemplo

El ejemplo siguiente muestra código HTML que se podría incluir en una ubicación de marketing (por ejemplo, en un mensaje de correo electrónico) que contenga un URL de etiqueta `IMG` (que pasa información cuando se abre el documento en el servidor Interact y recupera la imagen de oferta adecuada en la respuesta) y un URL de etiqueta `A` (que determina qué información se pasa al servidor Interact cuando se pulsa):

```
<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>
```

En el ejemplo siguiente, una etiqueta `IMG` está incluida en un código `A`, lo que causa el comportamiento siguiente:

1. Cuando se abre el mensaje de correo electrónico, Message Connector recibe una solicitud que contiene información codificada en la etiqueta IMG: el IDmsj e IDEnlace de este mensaje y los parámetros de cliente que incluyen ID de usuario, nivel de ingresos y tipo de ingresos.
2. Esta información se pasa mediante una llamada a la API al servidor de ejecución de Interact.
3. El servidor de ejecución devuelve una oferta a Message Connector, que recibe el URL de la imagen de la oferta, y proporciona ese URL (con los parámetros adicionales incluidos) y redirecciona la solicitud de imagen a ese URL de oferta.
4. El cliente ve la oferta como una imagen.

En este punto, el cliente puede pulsar dicha imagen para responder a la oferta de alguna forma. Esa pulsación, mediante la etiqueta A y su atributo HREF especificado (que especifica el URL de destino), envía otra solicitud a Message Connector de una página de destino enlazada al URL de esa oferta. A continuación, el navegador del cliente se redirecciona a la página de destino tal como está configurado en la oferta.

Tenga en cuenta que una etiqueta A de pulsación no es estrictamente necesaria; la oferta puede constar solo de una imagen, como un vale para que el cliente lo imprima.

Instalación de Message Connector

Los archivos que necesita para instalar, desplegar y ejecutar Message Connector se han incluido automáticamente con la instalación del servidor de ejecución de IBM Interact. Esta sección resume los pasos necesarios para hacer que Message Connector este listo para su utilización.

La instalación y el despliegue de Message Connector implican las tareas siguientes:

- Opcionalmente, la configuración de los valores predeterminados de Message Connector, tal como se describe en "Configuración de Message Connector".
- La creación de tablas de base de datos necesarias para almacenar los datos de transacciones de Message Connector, tal como se describe en "Creación de tablas de Message Connector" en la página 309.
- La instalación de la aplicación web de Message Connector, tal como se describe en "Despliegue y ejecución de Message Connector" en la página 310.
- La creación de las etiquetas IMG y A en sus ubicaciones de marketing (correos electrónicos o páginas web, por ejemplo) necesarias para llamar a ofertas de Message Connector cuando se abran y pulsen, tal como se describe en "Creación de los enlaces de Message Connector" en la página 311.

Configuración de Message Connector

Para desplegar Message Connector, debe personalizar el archivo de configuración incluido con la instalación para que coincida con su entorno específico. Puede modificar el archivo XML denominado MessageConnectorConfig.xml que se encuentra en el directorio de Message Connector en el servidor de ejecución de Interact, similar a <inicio_Interact>/msgconnector/config/MessageConnectorConfig.xml.

Acerca de esta tarea

El archivo MessageConnectorConfig.xml contiene algunos valores de configuración que son necesarios, y otros que son opcionales. Los valores que utilice deben estar

personalizados para su instalación específica. Siga los pasos siguientes para modificar la configuración.

Procedimiento

1. Si Message Connector está desplegado y en ejecución en el servidor de aplicaciones web, anule el despliegue de Message Connector antes de continuar.
2. En el servidor de ejecución de Interact, abra el archivo `MessageConnectorConfig.xml` en cualquier editor de texto o XML.
3. Modifique los valores de configuración según se requiera, asegurándose de que los siguientes valores *necesarios* sean correctos para su instalación.
   - `<URL_interact>`, URL del servidor de ejecución de Interact al que se deben conectar las etiquetas de página de Message Connector y en el que se está ejecutando Message Connector.
   - `<enlace_error_imagen>`, URL al que redirigirá Message Connector si se produce un error al procesar una solicitud para una imagen de oferta.
   - `<enlace_error_página_destino>`, URL al que redirigirá Message Connector si se produce un error al procesar una solicitud para una página de destino de oferta.
   - `<niveles_audiencia>`, una sección del archivo de configuración que contiene uno o más valores de nivel de audiencia, y que especifica el nivel de audiencia predeterminado si el enlace de Message Connector no ha especificado ninguno. Debe haber como mínimo un nivel de audiencia configurado.

   Todos los valores de configuración se describen más detalladamente en "Valores de configuración de Message Connector".
4. Cuando haya completado los cambios de configuración, guarde y cierre el archivo `MessageConnectorConfig.xml`.
5. Continúe configurando y desplegando Message Connector.

Valores de configuración de Message Connector:

Para configurar Message Connector, puede modificar el archivo XML denominado `MessageConnectorConfig.xml` que se encuentra en el directorio de Message Connector en el servidor de ejecución de Interact, normalmente `<inicio_Interact>/msgconnector/config/MessageConnectorConfig.xml`. A continuación se describe cada una de las configuraciones de este archivo XML. Tenga en cuenta que si modifica este archivo una vez que Message Connector esté desplegado y en ejecución, debe asegurarse de anular el despliegue y volver a desplegar Message Connector o reiniciar el servidor de aplicaciones para volver a cargar estos valores una vez que haya acabado de modificar el archivo.

Valores generales

La tabla siguiente contiene una lista de valores opcionales y necesarios incluidos en la sección `generalSettings` del archivo `MessageConnectorConfig.xml`.

Tabla 24. Valores generales de Message Connector

Elemento	Descripción	Valor predeterminado
<interactURL>	URL del servidor de ejecución de Interact para manejar las llamadas de las etiquetas de página de Message Connector, como por ejemplo el servidor de ejecución en el que se está ejecutando Message Connector. Este elemento es necesario.	http://localhost:7001/interact
<defaultDateTimeFormat>	Formato de fecha predeterminado.	MM/dd/aaaa
<log4jConfigFileLocation>	Ubicación del archivo de propiedades Log4j. Es relativa a la variable de entorno *$MESSAGE_CONNECTOR_HOME* si está establecida; de lo contrario, este valor es relativo a la ruta raíz de la aplicación web de Message Connector.	config/ MessageConnectorLog4j.properties

Valores de parámetros predeterminados

La tabla siguiente contiene una lista de valores predeterminados y necesarios contenidos en la sección defaultParameterValues del archivo MessageConnectorConfig.xml.

Tabla 25. Valores de parámetros predeterminados de Message Connector

Elemento	Descripción	Valor predeterminado
<interactiveChannel>	Nombre del canal interactivo predeterminado.	
<interactionPoint>	Nombre del punto de interacción predeterminado.	
<debugFlag>	Determina si la depuración está habilitada. Los valores permitidos son true y false.	false
<contactEventName>	Nombre predeterminado del evento de contacto que se publica.	
<acceptEventName>	Nombre predeterminado del evento de aceptación que se publica.	
<imageUrlAttribute>	Nombre de atributo de oferta predeterminado que contiene el URL de la imagen de oferta, si no se especifica ninguno en el enlace de Message Connector.	
<landingPageUrlAttribute>	URL predeterminado para la página de destino de pulsación si no se especifica ninguno en el enlace de Message Connector.	

Valores de comportamiento

La tabla siguiente contiene una lista de los valores opcionales y necesarios contenidos en la sección behaviorSettings del archivo MessageConnectorConfig.xml.

Tabla 26. Valores de comportamiento de Message Connector

Elemento	Descripción	Valor predeterminado
<imageErrorLink>	URL al que redirige el conector si se produce un error al procesar una solicitud para una imagen de oferta. Este valor es necesario.	/images/default.jpg
<landingPageErrorLink>	URL al que redirige el conector si se produce un error al procesar una solicitud de una página de destino de pulsación. Este valor es necesario.	/jsp/default.jsp
<alwaysUseExistingOffer>	Determina si la oferta en caché se debe devolver, incluso si ya ha caducado. Los valores permitidos son true y false.	false
<offerExpireAction>	Acción que se realizará si se encuentra la oferta original pero ya ha caducado. Los valores permitidos son: • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

Valores de almacenamiento

La tabla siguiente contiene una lista de valores opcionales y predeterminados contenidos en la sección storageSettings del archivo MessageConnectorConfig.xml.

Tabla 27. Valores de almacenamiento de MessageConnector

Elemento	Descripción	Valor predeterminado
<persistenceMode>	Cuando la memoria caché persiste nuevas entradas en la base de datos. Los valores permitidos son WRITE-BEHIND (donde los datos se graban en la memoria caché inicialmente, y se actualizan en la base de datos posteriormente) y WRITE-THROUGH (donde los datos se graban en la memoria caché y en la base de datos simultáneamente).	WRITE-THROUGH
<maxCacheSize>	Número máximo de entradas en la memoria caché.	5000
<maxPersistenceBatchSize>	Tamaño máximo de proceso por lotes al persistir entradas en la base de datos.	200
<macCachePersistInterval>	Tiempo máximo en segundos que una entrada permanece en la memoria caché antes de que se persista en la base de datos.	3
<maxElementOnDisk>	Número máximo de entradas en la memoria caché de disco.	5000
<cacheEntryTimeToExpireInSeconds>	Periodo máximo de tiempo que las entradas en la memoria caché de disco permanecerán antes de caducar.	60000

Tabla 27. Valores de almacenamiento de MessageConnector (continuación)

Elemento	Descripción	Valor predeterminado
<jdbcSettings>	Sección del archivo XML que contiene información específica si se utiliza una conexión JDBC. Se excluye mutuamente con la sección <dataSourceSettings>.	Configurada de forma predeterminada para conectar a una base de datos de SQLServer configurada en el servidor local, pero si habilita esta sección debe proporcionar los valores de JDBC y las credenciales para iniciar sesión.
<dataSourceSettings>	Sección del archivo XML que contiene información específica si se utiliza una conexión de origen de datos. Se excluye mutuamente con la sección <jdbcSettings>.	Configurado de manera predeterminada para conectar al origen de datos InteractDS definido en el servidor de aplicaciones web local.

Niveles de audiencia

La siguiente tabla contiene una lista de los valores opcionales y necesarios contenidos en la sección audienceLevels del archivo MessageConnectorConfig.xml.

Tenga en cuenta que el elemento audienceLevels se utiliza opcionalmente para especificar el nivel de audiencia predeterminado que se utilizará si no se especifica ninguno en el enlace de Message Connector, como en el ejemplo siguiente:

```
<audienceLevels default="Cliente">
```

En este ejemplo, el valor del atributo predeterminado coincide con el nombre de un audienceLevel definido en esta sección. Debe haber como mínimo un nivel de audiencia definido en este archivo de configuración.

Tabla 28. Valor de nivel de audiencia de MessageConnector

Elemento	Elemento	Descripción	Valor predeterminado
<audienceLevel>		Elemento que contiene la configuración del nivel de audiencia. Proporcione un atributo de nombre, como en <audienceLevel name="Cliente">	
	<messageLogTable>	Nombre de la tabla de registro. Este valor es necesario.	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<field>	Definición de uno o más campos de ID de audiencia para este audienceLevel.	
	<name>	Nombre del campo de ID de audiencia, tal como se especifica en el tiempo de ejecución de Interact.	
	<httpParameterName>	Nombre de parámetro correspondiente para este campo de ID de audiencia.	
	<dbColumnName>	Nombre de columna correspondiente en la base de datos para este campo de ID de audiencia.	

Tabla 28. Valor de nivel de audiencia de MessageConnector (continuación)

Elemento	Elemento	Descripción	Valor predeterminado
	<type>	Tipo del campo de ID de audiencia, tal como se especifica en el tiempo de ejecución de Interact. Los valores pueden ser de tipo serie o numérico.	

Creación de tablas de Message Connector

Para desplegar IBM Interact Message Connector, primero debe crear las tablas en la base de datos donde están almacenados los datos de tiempo de ejecución de Interact. Creará una tabla para cada nivel de audiencia que haya definido. Para cada nivel de audiencia, Interact utilizará las tablas que cree para registrar información sobre transacciones de Message Connector.

Acerca de esta tarea

Utilice el cliente de base de datos para ejecutar el script SQL de Message Connector en la base de datos o el esquema apropiado para crear las tablas necesarias. Los scripts SQL para la base de datos soportada se instalan automáticamente al instalar el servidor de ejecución de Interact. Consulte las hojas de trabajo que ha completado en la publicación *IBM Interact Guía de instalación* para ver detalles sobre cómo conectarse a la base de datos que contiene las tablas de ejecución de Interact.

Procedimiento

1. Lance el cliente de base de datos y conéctese a la base de datos en la que están almacenadas actualmente las tablas de ejecución de Interact.
2. Ejecute el script adecuado del directorio <*Interact_home*>/msgconnector/scripts/ddl. La lista siguiente muestra los scripts SQL de muestra que se pueden utilizar para crear manualmente tablas de Message Connector:

Tabla 29. Scripts de creación de tablas de Message Connector

Tipo de origen de datos	Nombre de script
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

Tenga en cuenta que estos scripts se proporcionan como muestras. Es posible que utilice un convenio de denominación o estructura distinto para los valores de ID de audiencia, por lo que puede que necesite modificar el script antes de ejecutarlo. Normalmente, un método recomendado es tener una tabla dedicada para cada nivel de audiencia.

Las tablas se crean para contener la siguiente información:

Tabla 30. Información creada por los scripts SQL de muestra

Nombre de columna	Descripción
IDRegistro	Clave primaria de esta entrada.
IDMensaje	Identificador exclusivo de cada instancia de mensaje.
IDEnlace	Identificador exclusivo de cada enlace en el soporte electrónico (por ejemplo, un mensaje de correo electrónico).

Tabla 30. Información creada por los scripts SQL de muestra (continuación)

Nombre de columna	Descripción
URLImagenOferta	URL a la imagen relacionada de la oferta devuelta.
URLPáginaDestinoOferta	URL a la página de destino relacionada de la oferta devuelta.
CódigoTratamiento	Código de tratamiento de la oferta devuelta.
FechaCaducidadOferta	Fecha y hora de caducidad de la oferta devuelta.
FechaContactoOferta	Fecha y hora en la que la oferta se ha devuelto al cliente.
IDAudiencia	ID de audiencia del soporte electrónico.

Tenga en cuenta lo siguiente sobre esta tabla:

- En función del nivel de audiencia, habrá una columna IDAudiencia para cada componente de la clave de audiencia.
- La combinación de IDMensaje, IDEnlace e IDAudiencia forma una clave exclusiva de esta tabla.

Cuando los scripts hayan acabado de ejecutarse, se habrán creado las tablas necesarias para Message Connector.

Resultados

Ahora está preparado para desplegar la aplicación web Message Connector.

Despliegue y ejecución de Message Connector

Message Connector de IBM Interact se despliega como una aplicación web autónoma en un servidor de aplicaciones web soportado.

Antes de empezar

Para desplegar Message Connector, asegúrese de que se hayan completado las tareas siguientes:

- Debe haber instalado el servidor de ejecución de IBM Interact. La aplicación Message Connector desplegable se instala automáticamente con el servidor de ejecución, y está lista para su despliegue desde el directorio de inicio de Interact.
- También debe haber ejecutado los scripts SQL proporcionados con la instalación para crear las tablas necesarias en la base de datos de ejecución de Interact para que las utilice Message Connector tal como se describe en "Creación de tablas de Message Connector" en la página 309

Acerca de esta tarea

Exactamente de la misma forma en que se despliegan las aplicaciones de IBM en un servidor de aplicaciones web antes de que se ejecuten, se debe desplegar la aplicación Message Connector para que esté disponible para proporcionar las ofertas.

Procedimiento

1. Conéctese a la interfaz de gestión del servidor de aplicaciones web con los privilegios necesarios para desplegar una aplicación.
2. Siga las instrucciones del servidor de aplicaciones web para desplegar y ejecutar le archivo denominado <*inicio_Interact*>/msgconnector/MessageConnector.war Sustituya <*inicio_Interact*> por el directorio real en el que está instalado el servidor de ejecución de Interact.

Resultados

Message Connector ahora estará disponible para su uso. Una vez que haya configurado la instalación de Interact para crear los datos subyacentes que utilizará Message Connector para proporcionar ofertas, como por ejemplo canales interactivos y estrategias, diagramas de flujo, ofertas, etc., puede crear los enlaces en el soporte electrónico que aceptará Message Connector.

Creación de los enlaces de Message Connector

Para utilizar Message Connector para proporcionar imágenes de oferta personalizadas cuando un usuario final interactúe con el soporte electrónico (por ejemplo, abriendo un mensaje de correo electrónico) y páginas de destino personalizadas cuando el usuario final pulse en la oferta, debe crear los enlaces que se incorporarán en el mensaje. Esta sección proporciona un resumen de las etiquetas HTML de estos enlaces.

Acerca de esta tarea

Independientemente del sistema que utilice para generar los mensajes salientes a los usuarios finales, debe generar las etiquetas HTML para que contengan los campos adecuados (proporcionados en las etiquetas HTML como atributos) que contienen la información que desea pasar al servidor de ejecución de Interact. Siga los pasos siguientes para configurar la información mínima necesaria para un mensaje de Message Connector.

Tenga en cuenta que aunque las instrucciones aquí se refieren específicamente a mensajes que contienen enlaces de Message Connector, puede seguir los mismos pasos y la misma configuración para añadir enlaces a páginas web y otros soportes electrónicos.

Procedimiento

1. Cree el enlace `IMG` que aparecerá en el mensaje con, como mínimo, los parámetros siguientes:
   - `IDmsj`, que indica el identificador exclusivo para este mensaje.
   - `IDEnlace`, que indica el identificador exclusivo para el enlace en el mensaje.
   - `IDAudiencia`, el identificador de la audiencia a la que pertenece el destinatario del mensaje.

     Tenga en cuenta que si el ID de audiencia es un ID compuesto, todos estos componentes se deben incluir en el enlace.

   También puede incluir parámetros opcionales, que incluyen nivel de audiencia, nombre de canal interactivo, nombre de punto de interacción, URL de ubicación de imagen y los parámetros de personalización no utilizados específicamente por Message Connector.
2. Opcionalmente, cree un enlace `A` que incluya un enlace `IMG` de forma que, cuando el usuario pulse la imagen, el navegador cargue una página que contenga la oferta para el usuario. El enlace `A` debe contener también los tres parámetros listados anteriormente (`IDmsj`, `IDEnlace` y `IDAudiencia`), además de los parámetros adicionales (nivel de audiencia, nombre de canal interactivo y nombre de punto interactivo) y parámetros personalizados no utilizados específicamente por Message Connector. Tenga en cuenta que el enlace `A` muy probablemente contendrá un enlace `IMG` de Message Connector, pero también puede aparecer sólo, según se requiera. Si el enlace sí contiene un enlace `IMG`, el

enlace IMG podría contener el mismo conjunto de parámetros que el enlace A que lo incluye (incluidos los parámetros opcionales o personalizados).

3. Cuando los enlaces se hayan definido correctamente, genere y envíe los mensajes de correo electrónico.

Resultados

Para obtener información detallada sobre los parámetros disponibles, y los enlaces de muestra, consulte “Parámetros de solicitud HTTP de etiqueta "IMG" y "A"”

Parámetros de solicitud HTTP de etiqueta "IMG" y "A"

Cuando Message Connector recibe una solicitud, ya sea porque un usuario final ha abierto un correo electrónico que contiene una etiqueta IMG codificada por Message Connector o porque el usuario final ha pulsado una etiqueta A, analiza los parámetros incluidos con la solicitud para devolver los datos de oferta adecuados. Esta sección proporciona una lista de los parámetros que se pueden incluir en el URL solicitante (la etiqueta IMG (cargada automáticamente cuando se visualiza una imagen etiquetada al abrir un correo electrónico) o la etiqueta A (cargada cuando la persona que visualiza el correo electrónico pulsa el mensaje y va al sitio especificado).

Parámetros

Cuando Message Connector recibe una solicitud, analiza los parámetros incluidos con la solicitud. Estos parámetros incluyen algunos de los siguientes, o todos ellos:

Nombre de parámetro	Descripción	¿Necesario?	Valor predeterminado
IDmsj	Identificador exclusivo de la instancia de correo electrónico o página web.	Sí	Ninguno. Lo proporciona el sistema que crea la instancia exclusiva del mensaje de correo electrónico o página web que contiene la etiqueta.
IDEnlace	Identificador exclusivo del enlace de este correo electrónico o página web.	Sí	Ninguno. Lo proporciona el sistema que crea la instancia exclusiva del mensaje de correo electrónico o página web que contiene la etiqueta.
nivelAudiencia	Nivel de audiencia al que pertenece el destinatario de esta comunicación.	No	audienceLevel especificado como valor predeterminado en el elemento audienceLevels que se encuentra en el archivo MessageConnectorConfig.xml.
ic	Nombre del canal interactivo de destino (IC)	No	Valor del elemento interactiveChannel que se encuentra en la sección defaultParameterValues del archivo MessageConnectorConfig.xml, que es "interactiveChannel" de forma predeterminada.
ip	Nombre del punto de interacción (IP) que se aplica	No	Valor del elemento interactionPoint que se encuentra en la sección defaultParameterValues del archivo MessageConnectorConfig.xml, que es "headBanner" de forma predeterminada.
URLImagenoferta	URL de la imagen de oferta de destino para el URL IMG en el mensaje.	No	Ninguno.
AtrURLImagenOferta	Nombre del atributo de oferta que tiene el URL de la imagen de oferta de destino	No	Valor del elemento imageUrlAttribute que se encuentra en la sección defaultParameterValues del archivo MessageConnectorConfig.xml.
URLPáginaDestinoOferta	URL de la página de destino correspondiente a la oferta de destino.	No	Ninguno.

Nombre de parámetro	Descripción	¿Necesario?	Valor predeterminado
AtribPáginaDestinoOferta	Nombre del atributo de oferta que tiene el URL de la página de destino correspondiente a la oferta de destino.	No	Valor del elemento landingPageUrlAttribute que se encuentra en la sección defaultParameterValues del archivo MessageConnectorConfig.xml.
eventoContacto	Nombre del evento de contacto.	No	Valor del elemento contactEventName que se encuentra en la sección defaultParameterValues del archivo MessageConnectorConfig.xml, que es "contact" de forma predeterminada.
eventoRespuesta	Nombre del evento de aceptación.	No	Valor del elemento acceptEventName que se encuentra en la sección defaultParameterValues del archivo MessageConnectorConfig.xml, que es "accept" de forma predeterminada.
depuración	Indicador de depuración. Establezca este parámetro en "true" solo para la resolución de problemas y si se lo indica el personal de soporte técnico de IBM .	No	Valor del elemento debugFlag que se encuentra en la sección defaultParameterValues del archivo MessageConnectorConfig.xml, que es "false" de forma predeterminada.
<ID de audiencia>	ID de audiencia de este usuario. El nombre de este parámetro se define en el archivo de configuración.	Sí	Ninguno.

Cuando Message Connector recibe un parámetro que no se reconoce (es decir, no aparece en la lista anterior, se maneja de dos posibles maneras:

- Si se proporciona un parámetro no reconocido (por ejemplo, "attribute", como en attribute="attrValue") y hay un parámetro coincidente del mismo nombre más la palabra "Type" (por ejemplo, "attributeType", como en attributeType="string"), esto hace que Message Connector cree un parámetro de Interact coincidente y lo pase al tiempo de ejecución de Interact.

  Los valores del parámetro Type pueden ser los siguientes:

  – string
  – numeric
  – datetime

    Para un parámetro de tipo "datetime," Message Connector busca también un parámetro con el mismo nombre más la palabra "Pattern" (por ejemplo, "atributoPattern") cuyo valor es un formato válido de fecha/hora. Por ejemplo, podría proporciona el parámetro attributePattern="MM/dd/aaaa".

    Tenga en cuenta que si especifica un tipo de parámetro de "datetime" pero no proporciona un patrón de fecha coincidente, se utilizará el valor especificado en el archivo de configuración de Message Connector (que se encuentra en <directorio_instalación>/msgconnector/config/MessageConnectorConfig.xml) en el servidor Interact.
- Si se proporciona un parámetro no reconocido y no hay ningún valor Type coincidente, Message Connector pasa ese parámetro al URL de redireccionamiento de destino.

Para todos los parámetros no reconocidos, Message Connector los pasa al servidor de ejecución de Interact sin procesarlos o guardarlos.

Código de Message Connector de ejemplo

La siguiente etiqueta A contiene un ejemplo de un conjunto de enlaces de Message Connector que podría aparecer en un mensaje de correo electrónico:

```
<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>
```

En este ejemplo, la etiqueta `IMG` se carga automáticamente cuando se abre el mensaje de correo electrónico. Recuperando la imagen de la página especificada, el mensaje pasa los parámetros para el identificador de mensaje exclusivo (IDmsj), identificador de enlace exclusivo (IDEnlace) e identificador de usuario exclusivo (ID_usuario), junto con dos parámetros adicionales (códigoIngresos y tipoIngresos) para que se pasen al tiempo de ejecución de Interact.

La etiqueta `A` proporciona el atributo HREF (Hypertext Reference) que convierte la imagen de oferta en un enlace del mensaje de correo electrónico que se puede pulsar. Si el visor del mensaje, al ver la imagen, pulsa y va a la página de destino, el identificador de mensaje exclusivo (`IDmsj`), identificador de enlace (`IDEnlace`) e identificador de usuario (`ID_usuario`) se pasan al servidor, así como un parámetro adicional (`recomendación`) que se pasa al URL de redireccionamiento de destino.

Acerca de Interact Web Connector

Interact WebConnector (también denominado JavaScript Connector, o JSConnector) proporciona un servicio en el servidor de ejecución de Interact que permite que el código JavaScript llame a la API de Interact Java. Esto permite a las páginas web realizar llamadas a Interact para la personalización de ofertas en tiempo real utilizando solo código JavaScript incorporado, sin tener que basarse en lenguajes de desarrollo web (por ejemplo, Java, PHP, JSP, etc.). Por ejemplo, podría incorporar un pequeño fragmento de código JavaScript en cada página del sitio web que proporcione ofertas recomendadas por Interact de forma que cada vez que se cargue la página se realicen llamadas a la API de Interact para garantizar que se visualizan las mejores ofertas en la página de destino para el visitante del sitio.

Utilice Interact Web Connector en situaciones donde desee visualizar ofertas a los visitantes de una página donde es posible que no tenga control programático en el lado del servidor sobre la visualización de la página (como tendría, por ejemplo, con PHP u otros scripts basados en el servidor), pero aún puede incorporar código JavaScript en el contenido de la página que ejecutará el navegador web del visitante.

Consejo: Los archivos de Interact Web Connector se instalan automáticamente en el servidor de ejecución de Interact, en el directorio <*inicio_Interact*>/`jsconnector`. En el directorio <*inicio_Interact*>/`jsconnector`, encontrará un archivo ReadMe.txt que contiene notas importantes y detalles sobre las características de Web Connector, así como archivos de muestra y el código fuente de Web Connector que se utilizará como base para desarrollar sus propias soluciones. Si no encuentra aquí información que responda a sus preguntas, consulte el directorio `jsconnector` para obtener más información.

Instalación de Web Connector en el servidor de ejecución

Una instancia de Web Connector se instala automáticamente con el servidor de ejecución de IBM Interact y se habilita de forma predeterminada. Sin embargo, existen algunos valores que debe modificar para configurar y utilizar Web Connector.

Acerca de esta tarea

Los valores que debe modificar para poder utilizar el Web Connector instalado en el servidor de ejecución se añaden a la configuración del servidor de aplicaciones web. Por este motivo, necesitará reiniciar el servidor de aplicaciones web después de completar estos pasos.

Procedimiento

1. Para el servidor de aplicaciones web en el que está instalado el servidor de ejecución de Interact, establezca las siguientes propiedades Java:

   `-DUI_JSCONNECTOR_ENABLE_INPROCESS=true`

   `-DUI_JSCONNECTOR_HOME=<inicio_jsconnector>`

   Sustituya `<inicio_jsconnector>` por la ruta al directorio `jsconnector` del servidor de ejecución, que es `<directorio_inicio_Interact>/jsconnector`.

   La manera en la que establece las propiedades Java depende del servidor de aplicaciones web. Por ejemplo, en WebLogic, editaría el archivo `startWebLogic.sh` o `startWebLogic.cmd` para actualizar el valor `JAVA_OPTIONS`, como en este ejemplo:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
   jsconnector"
   ```

   En WebSphere Application Server, establecería esta propiedad en el panel de la máquina virtual Java de la consola de administración.

   Consulte la documentación del servidor de aplicaciones web para ver instrucciones específicas sobre cómo establecer las propiedades Java.
2. Reinicie el servidor de aplicaciones web si ya está en ejecución, o inicie ahora el servidor de aplicaciones web, para asegurarse de que se utilizan las nuevas propiedades Java.

Resultados

Cuando el servidor de aplicaciones web haya completado su proceso de inicio, habrá finalizado la instalación de Web Connector en el servidor de ejecución. El paso siguiente será conectarse a esta página web de Configuración en `http://<host>:<puerto>/interact/jsp/WebConnector.jsp`, donde `<host>` es el nombre de servidor de ejecución de Interact, y `<puerto>` es el puerto en el que Web Connector está a la escucha, tal como ha especificado el servidor de aplicaciones web.

Instalación de Web Connector como aplicación web independiente

Una instancia de Web Connector se instala automáticamente con el servidor de ejecución de IBM Interact y se habilita de forma predeterminada. Sin embargo, también puede desplegar Web Connector como su propia aplicación web (por ejemplo, en un servidor de aplicaciones web en un sistema individual) y configurarlo para comunicarse con el servidor de ejecución de Interact remoto.

Acerca de esta tarea

Estas instrucciones describen el proceso de desplegar Web Connector como una aplicación web individual con acceso a un servidor de ejecución de Interact remoto.

Para desplegar Web Connector, debe tener instalado el servidor de ejecución de IBM Interact y debe tener un servidor de aplicaciones web en otro sistema con acceso a la red (no bloqueado por ningún cortafuegos) al servidor de ejecución de Interact.

Procedimiento

1. Copie el directorio `jsconnector` que contiene los archivos de Web Connector del servidor de ejecución de Interact en el sistema donde el servidor de aplicaciones web (por ejemplo, WebSphere Application Server) ya está configurado y en ejecución. Puede encontrar el directorio `jsconnector` en el directorio de instalación de Interact.
2. En el sistema donde desplegará la instancia de Web Connector, configure el archivo `jsconnector/jsconnector.xml` utilizando cualquier editor de texto o XML para modificar el atributo `interactURL`.

   Se establece de forma predeterminada en `http://localhost:7001/interact`, pero debe modificarlo para que coincida con el URL del servidor de ejecución remoto de Interact, como por ejemplo `http://runtime.example.com:7011/interact`.

   Una vez que haya desplegado Web Connector, podrá utilizar una interfaz web para personalizar los demás valores del archivo `jsconnector.xml`. Consulte "Configuración de Web Connector" en la página 317 para obtener más información.
3. Para el servidor de aplicaciones web en el que desplegará Web Connector, establezca la siguiente propiedad Java:

   `-DUI_JSCONNECTOR_HOME=<inicio_jsconnector>`

   Sustituya `<inicio_jsconnector>` por la ruta real donde ha copiado el directorio `jsconnector` en el servidor de aplicaciones web.

   La manera en la que establece las propiedades Java depende del servidor de aplicaciones web. Por ejemplo, en WebLogic, editaría el archivo `startWebLogic.sh` o `startWebLogic.cmd` para actualizar el valor `JAVA_OPTIONS`, como en este ejemplo:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/InteractFiles/
   jsconnector"
   ```

   En WebSphere Application Server, establecería esta propiedad en el panel de la máquina virtual Java de la consola de administración.

   Consulte la documentación del servidor de aplicaciones web para ver instrucciones específicas sobre cómo establecer las propiedades Java.
4. Reinicie el servidor de aplicaciones web si ya estaba en ejecución, o inícielo en este paso, para asegurarse de que se utiliza la nueva propiedad Java.

   Espere a que el servidor de aplicaciones web complete su proceso de inicio antes de continuar.
5. Conéctese a la interfaz de gestión del servidor de aplicaciones web con los privilegios necesarios para desplegar una aplicación.
6. Siga las instrucciones del servidor de aplicaciones web para desplegar y ejecutar el archivo siguiente: `jsConnector/jsConnector.war`

Resultados

Web Connector ahora está desplegado en la aplicación web. Una vez que tenga el servidor Interact completamente configurado activo y en ejecución, el paso siguiente es conectarse a la página de Configuración de Web Connector en `http://<host>:<puerto>/interact/jsp/WebConnector.jsp`, donde *<host>* es el

sistema que ejecuta el servidor de aplicaciones web en el que acaba de desplegar Web Connector y <*puerto*> es el puerto en el que Web Connector está a la escucha, tal como especifica el servidor de aplicaciones web.

Configuración de Web Connector

Los valores de configuración de Interact Web Connector se almacenan en un archivo denominado `jsconnector.xml` que se almacena en el sistema donde está desplegado Web Connector (por ejemplo, el propio servidor de ejecución de Interact o en un sistema distinto que ejecute el servidor de aplicaciones web). Puede editar el archivo `jsconnector.xml` directamente utilizando cualquier editor de texto o XML; sin embargo, una forma fácil de configurar casi todos los valores de configuración disponibles es utilizar la página de configuración de Web Connector desde el navegador web.

Antes de empezar

Para utilizar la interfaz web para configurar Web Connector, debe instalar y desplegar la aplicación web que proporciona Web Connector. En el servidor de ejecución de Interact se instala automáticamente una instancia de Web Connector al instalar y desplegar Interact. En cualquier otro servidor de aplicaciones web, debe instalar y desplegar la aplicación web de Web Connector tal como se describe en "Instalación de Web Connector como aplicación web independiente" en la página 315.

Procedimiento

1. Abra el navegador web soportado y abra un URL similar al siguiente:

   `http://<host>:<puerto>/interact/jsp/WebConnector.jsp`

   - Sustituya <*host*> por el servidor en el que se ejecuta Web Connector, por ejemplo, el nombre de host del servidor de ejecución o el nombre del servidor en el que se ha desplegado una instancia individual de Web Connector.
   - Sustituya <*puerto*> por el número de puerto en el que la aplicación web de Web Connector está a la escucha de conexiones, el cual normalmente coincide con el puerto predeterminado del servidor de aplicaciones web.
2. En la página Configuraciones que aparece, complete las secciones siguientes:

Tabla 31. Resumen de valores de configuración de Web Connector.

Sección	Configuración
Valores básicos	Utilice la página Valores básicos para configurar el comportamiento global de Web Connector para el sitio en el que desplegará las páginas etiquetadas. Estos valores incluyen el URL base del sitio, información que requiere utilizar Interact sobre los visitantes del sitio, y valores similares que se aplican a todas las páginas que tiene previsto etiquetar con el código de Web Connector. Consulte "Opciones básicas de la configuración de WebConnector" en la página 319 para obtener más detalles.

Tabla 31. Resumen de valores de configuración de Web Connector (continuación).

Sección	Configuración
Tipos de visualización HTML	Utilice la página Tipos de visualización HTML para determinar el código HTML que se proporcionará para cada punto de interacción en la página. Puede elegir en la lista de plantillas predeterminadas (archivos .flt) que contienen alguna combinación de código de hoja de estilo en cascada (CSS), código HTML y código Javascript para utilizar para cada punto de interacción. Puede utilizar las plantillas tal como se proporcionan, personalizarlas según se requiera o crear las suyas propias. Los valores de configuración de esta página se corresponden con la sección `interactionPoints` del archivo de configuración `jsconnector.xml`. Consulte "Tipos de visualización HTML de la configuración de WebConnector" en la página 321 para obtener más detalles.
Páginas ampliadas	Utilice la página Páginas ampliadas para correlacionar valores específicos de página con un patrón de URL. Por ejemplo, podría configurar una correlación de página como por ejemplo que cualquier URL que contuviera el texto "index.htm" se visualizara en la página de bienvenida general, con eventos de carga de página específicos y puntos de interacción definidos para esa correlación. Los valores de configuración de esta página se corresponden con la sección `pageMapping` del archivo de configuración `jsconnector.xml`. Consulte "Páginas ampliadas de configuración de WebConnector" en la página 324 para obtener más detalles.

3. En la página Valores básicos, verifique que los valores del sitio sean válidos para la instalación y, opcionalmente, especifique el modo de depuración (no recomendada a menos que está realizando la resolución de un problema), la integración de etiqueta de página de Digital Analytics for On Premises y los valores predeterminados para la mayoría de puntos de interacción y, a continuación, pulse el enlace Tipos de visualización HTML en Configuraciones.
4. En la página Tipos de visualización HTML, siga estos pasos para añadir o modificar plantillas de visualización que definan los puntos de interacción en la página web del cliente.

   De forma predeterminada, las plantillas de visualización (archivos .flt) se almacenan en `<inicio_jsconnector>/conf/html`.

   a. Seleccione en la lista el archivo .flt que desee examinar o utilizar como punto de partida, o pulse Añadir un tipo para crear una nueva plantilla de punto de interacción en blanco para utilizar.

      La información sobre el contenido de la plantilla, si la hay, aparece junto a la lista de plantillas.

   b. Opcionalmente, modifique el nombre de la plantilla en el campo **Nombre de archivo para este tipo de visualización**. En el caso de una plantilla nueva, actualice `CHANGE_ME.flt` a algo más descriptivo.

      Si renombra aquí la plantilla, Web Connector creará un nuevo archivo con ese nombre la próxima vez que se guarde la plantilla. Las plantillas se guardan cuando se modifica el cuerpo del texto y a continuación se pasa a otro campo.

   c. Modifique o complete la información de Fragmento de código HTML según se requiera, incluyendo el código de hojas de estilo (CSS), JavaScript y HTML que desee incluir. Tenga en cuenta que también puede incluir variables que sustituyan parámetros de Interact en tiempo de ejecución. Por

ejemplo, ${offer.HighlightTitle} se sustituye automáticamente por el título de la oferta en la ubicación especificada del punto de interacción.

Utilice los ejemplos que aparecen a continuación del campo Fragmento HTML para ver indicaciones sobre cómo dar formato a los bloques de código CSS, JavaScript o HTML.

5. Utilice la página Páginas ampliadas según se requiera para configurar las correlaciones de página que determinan cómo se manejan patrones de URL específicos en las páginas.
6. Cuando haya acabado de establecer las propiedades de configuración, pulse **Desplegar los cambios**. Al pulsar **Desplegar los cambios** se realizarán las acciones siguientes:
   - Se visualiza la etiqueta de página de IBM Interact Web Connector, que contiene el código JavaScript que se puede copiar de la página de Web Connector e insertar en sus páginas web.
   - Se realiza una copia de seguridad del archivo de configuración existente de Web Connector en el servidor Interact (el archivo jsconnector.xml en el servidor donde está instalado Web Connector) y se crea un nuevo archivo de configuración con los valores que ha definido.

     Los archivos de configuración de copia de seguridad se almacenan en <*inicio_jsconnector*>/conf/archive/jsconnector.xml.<*fecha*>.<*hora*>, como en jsconnector.xml.20111113.214933.750-0500 (donde la serie de fecha es 20111113 y la serie de hora, incluyendo el indicador de huso horario, es 214933.750-0500)

Resultados

Ahora ha completado la configuración de Web Connector.

Para modificar la configuración, puede volver al inicio de estos pasos y realizarlos de nuevo con nuevos valores, o puede abrir el archivo de configuración (<inicio_Interact>/jsconnector/conf/jsconnector.xml) en cualquier editor de texto o XML y modificarlo según se requiera.

Opciones básicas de la configuración de WebConnector

Utilice la página Valores básicos de la página Configuraciones de Web Connector para configurar el comportamiento global de Web Connector para el sitio en el que estará desplegando las páginas etiquetadas. Estos valores incluyen el URL base del sitio, información que requiere utilizar Interact sobre los visitantes del sitio, y valores similares que se aplican a todas las páginas que tiene previsto etiquetar con el código de Web Connector.

Valores del sitio

Las opciones de configuración de Valores del sitio son valores globales que afectan al comportamiento global de la instalación del Web Connector que está configurando. Puede especificar los valores siguientes:

Tabla 32. Valores del sitio para la instalación de Web Connector

Valor	Descripción	Valor equivalente en jsconnector.xml
URL de la API de Interact	URL base del servidor de ejecución de Interact. **Nota:** Este valor sólo se utiliza si Web Connector no se ejecuta en el servidor de ejecución de Interact (es decir, lo ha desplegado por separado).	*<URLinteract>*
URL de Web Connector	URL base utilizado para generar el URL de pulsación.	*<URLjsConnector>*
Nombre de canal interactivo para el sitio web de destino	Nombre del canal interactivo que se ha definido en el servidor Interact que representa esta correlación de página.	*<canalInteractivo>*
Nivel de audiencia de visitantes	El nivel de audiencia de Campaign para el visitante entrante; se utiliza en la llamada a la API al tiempo de ejecución de Interact.	*<nivelAudiencia>*
Nombre de campo de ID de audiencia en la tabla de perfil	Nombre del campo IDAudiencia que se utilizará en la llamada a la API a Interact. Tenga en cuenta que actualmente no se da soporte a identificadores de audiencia de varios campos.	*<campoIDAudiencia>*
Tipo de datos del campo de ID de audiencia	Tipo de datos del campo de ID de audiencia ("numérico" o "serie") que se utilizará en la llamada a la API a Interact.	*<tipoCampoIDAudiencia>*
Nombre de cookie que representa el ID de sesión	Nombre de la cookie que contendrá el ID de sesión.	*<cookieIDsesión>*
Nombre de cookie que representa el ID de visitante	Nombre de la cookie que contendrá el ID de visitante.	*<cookieIDvisitante>*

Características opcionales

Las opciones de configuración de Características opcionales son valores globales opcionales de la instalación del Web Connector que está configurando. Puede especificar los valores siguientes:

Tabla 33. Valores opcionales del sitio para la instalación de Web Connector

Valor	Descripción	Valor equivalente en jsconnector.xml
Habilitar modo de depuración	Especifica (con una respuesta de `sí` o `no`) si se utilizará un modo de depuración especial. Si habilita esta característica, el contenido devuelto de Web Connector incluye una llamada de Javascript a 'alerta', que informa al cliente de la correlación de página determinada que se acaba de producir. El cliente debe tener una entrada en el archivo especificado por el valor *<authorizedDebugClients>* para obtener la alerta.	*<habilitarModoDepuración>*

Tabla 33. Valores opcionales del sitio para la instalación de Web Connector (continuación)

Valor	Descripción	Valor equivalente en jsconnector.xml
Archivo autorizado de host de clientes de depuración	Ruta de un archivo que contiene la lista de hosts o direcciones IP (Internet Protocol) que califican para el modo de depuración. Para que se pueda recopilar la información de depuración, el nombre de host o la dirección IP de un cliente debe aparecer en el archivo especificado.	*<clientesDepuraciónAutorizados>*
Habilitar integración de etiqueta de página de Digital Analytics for On Premises	Especifica (con una respuesta de `sí` o `no`) si Web Connector debe adjuntar la etiqueta especificada de IBM Digital Analytics for On Premises al final del contenido de la página.	`<habilitarEtiquetasNetInsight>`
Archivo de plantilla HTML de etiqueta de Digital Analytics for On Premises	Plantilla HTML/Javascript utilizada para integrar una llamada con la etiqueta de Digital Analytics for On Premises. Normalmente, debe aceptar el valor predeterminado a menos que se le indique que proporcione una plantilla distinta.	`<etiquetaNetInsight>`

Tipos de visualización HTML de la configuración de WebConnector

Utilice la página Tipos de visualización HTML para determinar el código HTML que se proporcionará para cada punto de interacción en la página. Puede elegir en la lista de plantillas predeterminadas (archivos .flt) que contienen alguna combinación de código de hoja de estilo en cascada (CSS), código HTML y código JavaScript para utilizar para cada punto de interacción. Puede utilizar las plantillas tal como se proporcionan, personalizarlas según se requiera o crear las suyas propias.

Nota: Los valores de configuración de esta página se corresponden con la sección `interactionPoints` del archivo de configuración `jsconnector.xml`.

El punto de interacción también puede contener marcadores de posición (zonas) en las que se pueden soltar automáticamente atributos de oferta. Por ejemplo, podría incluir `${offer.TREATMENT_CODE}`, que se sustituiría con el código de tratamiento asignado a esa oferta durante la interacción.

Las plantillas que aparecen en esta página se cargan automáticamente desde los archivos almacenados en el directorio `<Interact_home>/jsconnector/conf/html` del servidor de Web Connector. Las nuevas plantillas que cree aquí se almacenarán también en ese directorio.

Para utilizar la página Tipos de visualización HTML para visualizar o modificar cualquiera de las plantillas existentes, seleccione el archivo `.flt` en la lista.

Para crear una nueva plantilla en la página Tipo de visualización HTML, pulse **Añadir un tipo**.

Independientemente del método que elija para crear o modificar una plantilla, la siguiente información aparecerá junto a la lista de plantillas:

Valor	Descripción	Valor equivalente en jsconnector.xml
Nombre de archivo para este tipo de visualización	Nombre asignado a la plantilla que está editando. Este nombre debe ser válido para el sistema operativo en el que se ejecuta Web Connector; por ejemplo, no puede utilizar una barra inclinada (/) en el nombre si el sistema operativo es Microsoft Windows. Si está creando una nueva plantilla, este campo está preestablecido en `CHANGE_ME.flt`. Debe cambiarlo a un valor significativo antes de continuar.	*<fragmento_código_html>*

Valor	Descripción	Valor equivalente en jsconnector.xml
Fragmento de código HTML	Contenido específico que Web Connector debe insertar en el punto de interacción en la página web. Este fragmento de código puede contener código HTML, información de formato CSS o JavaScript que se ejecutará en la página. Cada uno de estos tres tipos de contenido debe estar entre los códigos BEGIN y END, como en los ejemplos siguientes: • ${BEGIN_HTML} <*su contenido HTML*> ${END_HTML} • ${BEGIN_CSS} <*su información de hoja de estilo específica del punto de interacción*> ${END_CSS} • ${BEGIN_JAVASCRIPT} <*su código JavaScript específico del punto de interacción*> ${END_JAVASCRIPT} Puede especificar también diversos códigos especiales predefinidos que se sustituyen automáticamente cuando se carga la página, incluyendo lo siguiente: • `${logAsAccept}`: una macro que toma dos parámetros (un URL de destino, y el TreatmentCode utilizado para identificar la aceptación de la oferta) y lo sustituye por el URL de pulsación. • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}` • `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` Cada uno de los códigos de oferta que se lista aquí representa atributos de oferta definidos en la plantilla de oferta en IBM Campaign que ha utilizado el usuario de marketing para crear las ofertas que devuelve Interact. Tenga en cuenta que Web Connector utiliza un motor de plantillas denominado FreeMarker que proporciona muchas opciones adicionales que puede encontrar útiles al configurar códigos en las plantillas de páginas. Consulte http://freemarker.org/docs/index.html para obtener más información.	No hay ningún equivalente porque el fragmento de código HTML reside en su propio archivo aparte de jsconnector.xml.

Valor	Descripción	Valor equivalente en jsconnector.xml
Códigos especiales de ejemplo	Contiene muestras de tipos de los códigos especiales, incluyendo códigos que identifican bloques como HTML, CSS o JAVASCRIPT, y zonas para soltar que se pueden insertar para hacer referencia a metadatos de oferta específicos.	Sin equivalente.

Los cambios que realice en esta página se guardarán automáticamente al navegar a otra página de configuración de Web Connector.

Páginas ampliadas de configuración de WebConnector

Utilice la página Páginas ampliadas para correlacionar valores específicos de página con un patrón de URL. Por ejemplo, podría configurar una correlación de página como por ejemplo que cualquier URL entrante que contuviera el texto "index.htm" se visualiza en la página de bienvenida general, con eventos de carga de página específicos y puntos de interacción definidos para esa correlación.

Nota: Los valores de configuración de esta página se corresponden con la sección `pageMapping` del archivo de configuración `jsconnector.xml`.

Para utilizar la página Páginas ampliadas para crear una nueva correlación de página, pulse el enlace **Añadir una página** y complete la información necesaria para la correlación.

Información sobre la página

Las opciones de configuración de Información sobre la página para la correlación de página definen el patrón de URL que actúa como el desencadenante de esta correlación, además de algunos valores adicionales para la manera en la que Interact maneja esta correlación de página.

Valor	Descripción	Valor equivalente en jsconnector.xml
URL contiene	Es el patrón de URL que debe vigilar Web Connector en la solicitud de página entrante. Por ejemplo, si el URL solicitante contiene "mortgage.htm", podría correlacionarlo con su página de información de hipoteca.	*<patrón_url>*
Nombre descriptivo de esta página o conjunto de páginas	Nombre descriptivo para su propia referencia que describe para qué es esta correlación de página, por ejemplo, "Página de información de hipoteca".	*<nombre_descriptivo>*
También devolver ofertas como datos JSON para uso de JavaScript	Lista desplegable para indicar si desea que Web Connector incluya los datos de oferta sin formato con el formato de notación de objeto JavaScript (http://www.json.org/) al final del contenido de la página.	*<habilitarRetornoDatosSinFormato>*

Eventos a activar (desencadenar) cuando se realice una visita a esta página o conjunto de páginas

Este conjunto de opciones de configuración de la correlación de página define el patrón de URL que actúa como desencadenante de esta correlación, además de algunos valores adicionales para la manera en la que Interact maneja esta correlación de página.

Nota: Los valores de configuración de esta sección se corresponden con la sección `<pageLoadEvents>` de `jsconnector.xml`.

Valor	Descripción	Valor equivalente en jsconnector.xml
Eventos individuales	Lista de eventos disponibles para esta página o este conjunto de páginas. Los eventos de esta lista son los que ha definido en Interact. Seleccione uno o más eventos que desee que se produzcan cuando se cargue la página. La secuencia de las llamadas a la API de Interact es la siguiente: 1. `startSession` 2. `postEvent` para cada evento de carga de página individual (siempre que haya definido los eventos individuales en Interact) 3. Para cada punto de interacción: • `getOffers` • `postEvent(ContactEvent)`	*<evento>*

Puntos de interacción (ubicaciones de visualización de oferta) en esta página o este conjunto de páginas

Este conjunto de opciones de configuración para la correlación de página le permite seleccionar qué puntos de interacción aparecerán en la página de Interact.

Nota: Los valores de configuración de esta sección se corresponden con la sección `<correlaciónPágina> | <página> | <puntosInteracción>` de `jsconnector.xml`.

Valor	Descripción	Valor equivalente en jsconnector.xml
Casilla de verificación del nombre del punto de interacción	Cada punto de interacción que se ha definido en el archivo de configuración aparece en esta sección de la página. Si selecciona la casilla de verificación junto al nombre del punto de interacción, se visualizará un número de opciones disponibles para ese punto de interacción.	*<puntoInteracción>*
ID de elemento HTML (Interact establecerá innerHTML)	Nombre del elemento HTML que debe recibir el contenido para este punto de interacción. Por ejemplo, si ha especificado `<div id="welcomebanner">` en la página, especificaría `welcomebanner` (el valor de ID) en este campo.	*<IDElementoHTML>*

Valor	Descripción	Valor equivalente en jsconnector.xml
Tipo de visualización HTML	Lista desplegable que le permite seleccionar el tipo de visualización HTML (los fragmentos de código HTML, o archivos .flt, definidos en una página de configuración anterior de Web Connector) para utilizar para este punto de interacción.	*<fragmento_código_html>*
Número máximo de ofertas a presentar (si se trata de un carrusel o flipbook)	Número máximo de ofertas que debe recibir Web Connector del servidor Interact para este punto de interacción. Este campo es opcional y se aplica solo para un punto de interacción que actualiza regularmente las ofertas presentadas sin volver a cargar la página, como en el escenario de carrusel donde se recuperan varias ofertas de forma que se pueden hacer disponibles una cada vez.	*<NúmeroMáxOfertas>*
Evento a activar cuando se presente la oferta	Nombre del evento de contacto que se publicará para este punto de interacción.	*<eventoContacto>*
Evento a activar cuando se acepte la oferta	Nombre del evento de aceptación que se publicará para este punto de interacción cuando se pulse el enlace de la oferta.	*<eventoAceptación>*
Evento a activar cuando se rechace la oferta	Nombre del evento de rechazo que se publicará para este punto de interacción. **Nota:** En este momento esta característica no se utiliza aún.	*<eventoRechazo>*

Opciones de configuración de Web Connector

Normalmente, puede utilizar una interfaz gráfica de Web Connector para configurar los valores de Web Connector. Todos los valores que especifica se almacenan también en un archivo denominado jsconnector.xml, que se encuentra en el directorio `jsconnector/conf`. Aquí se describen todos los parámetros que se guardan en el archivo de configuración jsconnector.xml.

Parámetros y sus descripciones

Los parámetros siguientes se almacenan en el archivo `jsconnector.xml` y se utilizan para las interacciones de Web Connector. Hay dos maneras de modificar estos valores:

- Mediante la página web de Configuración de Web Connector que está automáticamente disponible una vez que se ha desplegado e iniciado la aplicación Web Connector. Para utilizar la página web de Configuración, utilice el navegador web para abrir un URL similar al siguiente: `http://<host>:<puerto>/interact/jsp/WebConnector.jsp`.

  Los cambios que realiza en la página web de Administración se almacenan en el archivo `jsconnector.xml` en el servidor donde está desplegado Web Connector.
- Edite el archivo `jsconnector.xml` directamente mediante cualquier editor de texto o XML. Asegúrese de que se siente cómodo editando etiquetas y valores XML antes de utilizar este método.

Nota: Cada vez que edita manualmente el archivo `jsconnector.xml`, puede volver a cargar estos valores abriendo la página de Administración de Web Connector (que se encuentra en `http://<host>:<puerto>/interact/jsp/jsconnector.jsp`) y pulsando **Volver a cargar configuración**.

La tabla siguiente describe las opciones de configuración que se pueden establecer cuando aparecen en el archivo `jsconnector.xml`.

Tabla 34. Opciones de configuración de Web Connector

Grupo de parámetros	Parámetro	Descripción
`defaultPageBehavior`		
	`friendlyName`	Identificador legible por personas para el patrón de URL para que se visualice en la página de configuración web de Web Connector.
	`interactURL`	URL base del servidor de ejecución de Interact. Nota: Debe establecer este parámetro solo si el servicio de Web Connector (jsconnector) se ejecuta como aplicación web desplegada. No necesita establecer este parámetro si WebConnector se ejecuta automáticamente como parte del servidor de ejecución de Interact.
	`jsConnectorURL`	URL base utilizado para generar el URL de pulsación, por ejemplo, `http://host:puerto/jsconnector/clickThru`
	`interactiveChannel`	Nombre del canal interactivo que representa esta correlación de página.
	`sessionIdCookie`	Nombre de la cookie que contiene el ID de sesión que se utiliza en las llamadas a la API a Interact.
	`visitorIdCookie`	Nombre de la cookie que contiene el ID de audiencia.
	`audienceLevel`	Nivel de audiencia de campaña para el visitante entrante, utilizado en la llamada a la API al tiempo de ejecución de Interact.
	`audienceIdField`	Nombre del campo `IDAudiencia` utilizado en la llamada a la API al tiempo de ejecución de Interact. **Nota:** Nota: Actualmente no hay soporte para identificadores de audiencia de varios campos.
	`audienceIdFieldType`	Tipo de datos del campo de ID de audiencia [`numérico | serie`] utilizado en la llamada a la API al tiempo de ejecución de Interact
	`audienceLevelCookie`	Nombre de la cookie que contendrá el nivel de audiencia. Es opcional. Si no establece este parámetro, el sistema utiliza lo que se define para `audienceLevel`.
	`relyOnExistingSession`	Se utiliza en la llamada a la API al tiempo de ejecución de Interact. Normalmente, este parámetro se establece en "true".

Tabla 34. Opciones de configuración de Web Connector (continuación)

Grupo de parámetros	Parámetro	Descripción
	enableInteractAPIDebug	Se utiliza en la llamada a la API al tiempo de ejecución de Interact para habilitar la salida de depuración a los archivos de registro.
	pageLoadEvents	Evento que se publicará una vez que se cargue esta página determinada. Especifique uno o más eventos en esta etiqueta, con un formato similar a <evento>evento1</evento>.
	interactionPointValues	Todos los elementos de esta categoría actúan como valores predeterminados para los valores que faltan en las categorías específicas de puntos de interacción.
	interactionPointValuescontactEvent	Nombre predeterminado del evento de contacto que se publicará para este punto de interacción específico.
	interactionPointValuesacceptEvent	Nombre predeterminado del evento de aceptación que se publicará para este punto de interacción específico.
	interactionPointValuesrejectEvent	Nombre predeterminado del evento de rechazo que se publicará para este punto de interacción específico. (Nota: en este momento, esta característica no se utiliza.)
	interactionPointValueshtmlSnippet	Nombre predeterminado de la plantilla HTML que se utilizará para este punto de interacción.
	interactionPointValuesmaxNumberOfOffers	Número máx. predeterminado de ofertas que se recuperarán de Interact para este punto de interacción.
	interactionPointValueshtmlElementId	Nombre predeterminado del elemento HTML que recibirá el contenido de este punto de interacción.
	interactionPoints	Esta categoría contiene la configuración de cada punto de interacción. Para las propiedades que faltan, el sistema se basará en lo que se ha configurado en la categoría interactionPointValues.
	interactionPointname	Nombre del punto de interacción (IP).
	interactionPointcontactEvent	Nombre del evento de contacto que se publicará para este punto de contacto específico.
	interactionPointacceptEvent	Nombre del evento de aceptación que se publicará para este punto de contacto específico.
	interactionPointrejectEvent	Nombre del evento de rechazo que se publicará para este punto de contacto específico. (Tenga en cuenta que esta característica no se utiliza aún.)
	interactionPointhtmlSnippet	Nombre de la plantilla HTML que se utilizará para este punto de interacción.

Tabla 34. Opciones de configuración de Web Connector (continuación)

Grupo de parámetros	Parámetro	Descripción
	interactionPointmaxNumberOfOffers	Número máximo de ofertas que se recuperarán de Interact para este punto de interacción
	interactionPointhtmlElementId	Nombre del elemento HTML para recuperar el contenido para este punto de interacción.
	enableDebugMode	Indicador booleano (valores aceptables: true o false) para activar el modo de depuración especial. Si se establece en true, el contenido devuelto de Web Connector incluye una llamada de JavaScript a 'alerta' que informa al cliente sobre la correlación de página determinada que se acaba de producir. El cliente debe tener una entrada en el archivo authorizedDebugClients para generar la alerta.
	authorizedDebugClients	Archivo utilizado por el modo de depuración especial que consta de una lista de nombres de host o direcciones IP (IP) que califican para el modo de depuración.
	enableRawDataReturn	Indicador booleano (valores aceptables: true o false) para determinar si Web Connector se conecta a datos de oferta sin formato con formato JSON al final del contenido.
	enableNetInsightTagging	Indicador booleano (valores aceptables: true o false) para determinar si Web Connector se conecta al indicador de Digital Analytics for On Premises al final del contenido.
	apiSequence	Representa una implementación de la interfaz de APISequence, que dicta la secuencia de las llamadas a la API realizadas por Web Connector cuando se llama a un pageTag. De forma predeterminada, la implementación utiliza una secuencia de StartSession, pageLoadEvents, getOffers y logContact, donde los dos últimos son específicos de cada punto de interacción.
	clickThruApiSequence	Representa una implementación de la interfaz de APISequence, que dicta una secuencia de las llamadas a la API realizadas por Web Connector cuando se llama a un clickThru. De forma predeterminada, la implementación utiliza una secuencia de StartSession y logAccept.
	netInsightTag	Representa la plantilla HTML y JavaScript utilizada para integrar una llamada con la etiqueta de Digital Analytics for On Premises. Normalmente, no debería ser necesario cambiar esta opción.

Utilización de la página de administración de Web Connector

Web Connector incluye una página de administración que proporciona algunas herramientas para ayudarle a gestionar y probar la configuración tal como se podría utilizar con patrones de URL específicos. También puede utilizar la página de administración para volver a cargar la configuración que ha modificado.

Acerca de la página de administración

Mediante el navegador web soportado, puede abrir `http://host:puerto/interact/jsp/jsconnector.jsp`, donde *host:puerto* es el nombre de host en el que se ejecuta Web Connector y el puerto en el que está a la escucha de conexiones, por ejemplo, `runtime.example.com:7001`

Puede utilizar la página de administración de cualquiera de las formas siguientes:

Tabla 35. Opciones de la página de administración de Web Connector

Opción	Finalidad
Volver a cargar configuración	Pulse el enlace **Volver a cargar configuración** para volver a cargar en la memoria los cambios de configuración que se hayan guardado en disco. Esto es necesario si ha realizado cambios directamente en el archivo de configuracion jsconnector.xml del conector web en lugar de mediante las páginas web de configuración.
Ver configuración	Visualice la configuración de WebConnector en función del patrón de URL que especifique en el campo **Ver configuración**. Cuando especifique el URL de una página y pulse **Ver configuración**, Web Connector devolverá la configuración que utilizará el sistema en función de esta correlación de patrón. Si no se encuentra ninguna coincidencia, se devolverá la configuración predeterminada. Esto es útil para probar si se está utilizando la configuración correcta para una página determinada.
Ejecutar etiqueta de página	Si se completan los campos de esta página y se pulsa **Ejecutar etiqueta de página**, Web Connector devolverá el resultado `pageTag` en función del patrón de URL. Esto simula la llamada a una etiqueta de página. La diferencia entre llamar a pageTag desde esta herramienta y utilizando un sitio web real es que si se utiliza esta página de administración se visualizarán los errores o excepciones. En el caso de un sitio web real, las excepciones no se devolverán y serán visibles solo en el archivo de registro de Web Connector.

Página de Web Connector de muestra

Por ejemplo, se ha incluido un archivo denominado `WebConnectorTestPageSA.html` en Interact Web Connector (en el directorio`<Interact_Home/jsconnector/webapp/html`) que muestra cuántas características de Web Connector se etiquetarían en una página. Por comodidad, esta misma página de muestra se presenta aquí.

Página HTML de Web Connector de muestra

```
<?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 es una página de prueba que contiene pageTag de WebConnector.  Dado que
el nombre de este archivo tiene incorporado TestPage, WebConnector detectará una
coincidencia de patrón de URL con el patrón de URL "testpage" de la versión
predeterminada de jsconnector.xml - aquí se aplicará la definición de configuración
correlacionada con ese patrón de URL "testpage". Esto significa que esta
página debería tener los ID de elemento HTML correspondientes a las IP de este
patrón de URL (es decir, 'welcomebanner', 'crosssellcarousel' y 'textservicemessage')
################################################################################ */

/* ###################################################################
Esta sección establece las cookies para sessionId y visitorId.
Tenga en cuenta que en un sitio web de producción real, esto lo hace
muy probablemente el componente de inicio de sesión. En el caso de la
prueba, se hace aquí... el nombre de la cookie debe coincidir con lo
que se ha configurado en 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("IDSesión","123");
     setCookie("IDCliente","1");

   /* ####################################################
   Ahora se configuran los ID de elemento HTML correspondientes a las IP
   #################################################### */
     document.writeln("<div id='welcomebanner'> Esto se debe cambiar, "
    + "si no algo es incorrecto <\/div>");
     document.writeln("<div id='crosssellcarousel'> Esto se debe cambiar, "
    + "si no algo es incorrecto <\/div>");
     document.writeln("<div id='textservicemessage'> Esto se debe cambiar, "
    + "si no algo es incorrecto <\/div>");
   //]]&gt;
</script><!--
################################################################################
Esto es lo que se pega del archivo pageTag.txt en el directorio conf de la
instalación de WebConnector... la var unicaWebConnectorBaseURL se debe
ajustar para adecuarse al entorno local de WebConnector
################################################################################
-->
<!-- BEGIN: IBM Interact Web Connector Page Tag -->
<!--
# ****************************************************************************
#  Licensed Materials - Property of IBM
#  IBM Interact
#  (c) Copyright IBM Corporation 2001, 2012.
#  US Government Users Restricted Rights - Use, duplication or disclosure
#  restricted by GSA ADP Schedule Contract with IBM Corp.
# ****************************************************************************
-->
<script language="javascript" type="text/javascript">
//<![CDATA[
   var unicaWebConnectorBaseURL=
        "[CHANGE ME - http://host:port/<jsconnector>/pageTag]";
   var unicaURLData = "ok=Y";
   try {
    unicaURLData += "&url=" + escape(location.href)
   } catch (err) {}
   try {
    unicaURLData += "&title=" + escape(document.title)
   } catch (err) {}
   try {
```

```
 unicaURLData += "&referrer=" + escape(document.referrer)
} catch (err) {}
try {
 unicaURLData += "&cookie=" + escape(document.cookie)
} catch (err) {}
try {
 unicaURLData += "&browser=" + escape(navigator.userAgent)
} catch (err) {}
try {
 unicaURLData += "&screensize=" +
     escape(screen.width + "x" + screen.height)
} catch (err) {}
try {
 if (affiliateSitesForUnicaTag) {
     var unica_asv = "";
     document.write("<style id=\"unica_asht1\" type=\"text/css\"> "
       + "p#unica_ashtp a {border:1px #000000 solid; height:100px "
       + "!important;width:100px "
       + "!important; display:block !important; overflow:hidden "
       + "!important;} p#unica_ashtp a:visited {height:999px !important;"
       + "width:999px !important;} <\/style>");
     var unica_ase = document.getElementById("unica_asht1");
     for (var unica_as in affiliateSitesForUnicaTag) {
      var unica_asArr = affiliateSitesForUnicaTag[unica_as];
      var unica_ashbv = false;
      for (var unica_asIndex = 0; unica_asIndex <
       unica_asArr.length && unica_ashbv == false;
  unica_asIndex++)
{
       var unica_asURL = unica_asArr[unica_asIndex];
       document.write("<p id=\"unica_ashtp\" style=\"position:absolute; "
       + "top:0;left:-10000px;height:20px;width:20px;overflow:hidden; \
   margin:0;padding:0;visibility:visible;\"> \
   <a href=\"" + unica_asURL + "\">" + unica_as + " <\/a><\/p>");
       var unica_ae = document.getElementById("unica_ashtp").childNodes[0];
       if (unica_ae.currentStyle) {
        if (parseFloat(unica_ae.currentStyle["width"]) > 900)
           unica_ashbv = true
       } else if (window.getComputedStyle) {
        if (parseFloat(document.defaultView.getComputedStyle
           (unica_ae, null).getPropertyValue("width")) > 900)
        unica_ashbv = true
       }
       unica_ae.parentNode.parentNode.removeChild(unica_ae.parentNode)
      }
      if (unica_ashbv == true) {
       unica_asv += (unica_asv == "" ? "" : ";") + unica_as
      }
     }
     unica_ase.parentNode.removeChild(unica_ase);
     unicaURLData += "&affiliates=" + escape(unica_asv)
 }
} catch (err) {}
document.write("<script language='javascript' "
  + " type='text/javascript' src='" + unicaWebConnectorBaseURL + "?"
 + unicaURLData + "'><\/script>");
//]]&gt;
</script>
<style type="text/css">
/*<![CDATA[*/
   .unicainteractoffer {display:none !important;}
   /*]]&gt;*/
</style>
   <title>Página de Interact Web Connector de muestra</title>
   </head>
   <body>
   <!-- END: IBM Interact Web Connector Page Tag -->
```

```
   <!--
################################################################################
end of pageTag paste
################################################################################
-->
   </body>
</html>
```

Apéndice E. Integración de Interact y Digital Recommendations

IBM Interact se puede combinar con IBM Digital Recommendations para proporcionar recomendaciones de productos dirigidas por Interact. Ambos productos pueden proporcionar recomendaciones de productos para ofertas, pero utilizando métodos diferentes. Digital Recommendations utiliza el comportamiento web de un visitante (filtrado de colaboración) para crear correlaciones entre visitantes y ofertas recomendadas. Interact se basa en el comportamiento pasado del cliente, atributos, historial, y menos en las ofertas a nivel de visualización, y aprende qué ofertas se ajustan mejor al perfil de comportamiento de un cliente (basándose en datos demográficos y otra información sobre el cliente). Las tasas de aceptación de ofertas ayudan a crear un modelo predictivo a través del autoaprendizaje. Cuando se utilizan ambos productos, Interact puede utilizar un perfil personal para definir ofertas que pasarán un ID de categoría a Digital Recommendations y recuperarán productos recomendados basándose en su popularidad para ser mostrados al visitante como parte de las ofertas seleccionadas. Esto puede proporcionar mejores recomendaciones para los clientes que darán como resultado más pulsaciones en enlaces y mejores resultados que la actuación en solitario de cualquiera de los dos productos.

Las secciones siguientes describen cómo funciona esta integración, y cómo utilizar la aplicación de muestra proporcionada para crear su propia integración de oferta personalizada.

Descripción general de la integración de Interact con Digital Recommendations

Esta sección describe cómo IBM Interact se puede integrar con IBM Digital Recommendations para proporcionar recomendaciones de productos dirigidas por Interact, e incluye una descripción del proceso y los mecanismos mediante los cuales se realiza la integración.

IBM Interact se integra con IBM Digital Recommendations a través de la API REST (Representational State Transfer), que se proporciona al instalar Digital Recommendations. Mediante la realización de llamadas a la API REST con el ID de categoría adecuado, Interact puede recuperar productos recomendados e incluirlos en la información sobre ofertas que se muestra en la página personalizada que está viendo el visitante.

Cuando un visitante abre el URL de la página web (tal como la página JSP de muestra que se incluye con la instalación de Interact), la página llama a Interact para captar una oferta. Si la oferta se ha configurado dentro de Interact con los parámetros correctos, en el caso más simple tienen lugar los pasos siguientes:

1. La lógica de página identifica el ID de cliente del visitante.
2. Se realiza una llamada a la API a Interact y se pasa la información necesaria para generar una oferta para ese cliente.
3. La oferta devuelta proporciona la página web con al menos tres atributos: el URL de la imagen de la oferta, el URL de la página de destino cuando el cliente pulsa el enlace y el ID de categoría que se debe utilizar para determinar qué productos recomendar.

4. El ID de categoría se utiliza para llamar a Digital Recommendations a fin de recuperar los productos recomendados. Estos productos están en formato JSON (JavaScript Object Notation) y ordenados por nivel de ventas dentro de esa categoría.
5. La oferta y los productos se visualizan en el navegador del visitante.

Esta integración es útil para combinar juntos la recomendación de ofertas y las recomendaciones de productos. Por ejemplo, en una página web puede tener dos puntos de interacción: uno para una oferta, y otro para las recomendaciones correspondientes a esa oferta. Para conseguir esto, la página web hace una llamada a Interact para realizar una segmentación en tiempo real y determinar la mejor oferta (por ejemplo, para un descuento del 10% en todos los electrodomésticos pequeños). Cuando la página recibe la oferta desde Interact, esa oferta contiene el ID de categoría (en este ejemplo, para electrodomésticos pequeños). La página pasa el ID de categoría de electrodomésticos pequeños a Digital Recommendations utilizando una llamada a la API, y en respuesta recibe las mejores recomendaciones de productos para esa categoría de acuerdo con el nivel de ventas.

En un caso más simple, una página web hace una llamada a Interact sólo para encontrar una categoría (por ejemplo, cubertería de gama alta) que coincida con el perfil del cliente. La página pasa el ID de categoría recibido a Digital Recommendations y en respuesta recibe las recomendaciones de productos de cubertería.

Requisitos previos de la integración

Antes de poder utilizar la integración Digital Recommendations - Interact, compruebe que se cumplan los requisitos previos descritos en esta sección.

Asegúrese de que se cumplan los siguientes requisitos previos:

- Está familiarizado con el uso de la API de Interact tal como está documentada en la *Guía del administrador* y la ayuda en línea.
- Está familiarizado con la API REST de Digital Recommendations tal como está descrita en la documentación del desarrollador de Digital Recommendations.
- Tiene un conocimiento básico de HTML, JavaScript, CSS y JSON (JavaScript Object Notation).

  JSON es importante porque la API de REST de Digital Recommendations devuelve la información solicitada sobre productos en forma de datos con formato JSON.
- Está familiarizado con la codificación de páginas web en el extremo servidor, pues la aplicación de demostración proporcionada con Interact utiliza JSP (aunque JSP no es necesario).
- Tiene una cuenta válida de Digital Recommendations y la lista de los ID de categoría que piensa transferir a Interact para recuperar recomendaciones de productos (productos más vendidos o más populares dentro de la categoría especificada).
- Tiene el enlace de la API REST de Digital Recommendations (un URL correspondiente el entorno utilizado de Digital Recommendations).

  Consulte la aplicación de muestra que se incluye con la instalación de Interact para obtener un ejemplo, o consulte el código de muestra en "Utilización del proyecto de muestra de integración" en la página 338 para obtener más información.

Configuración de una oferta para la integración con Digital Recommendations

Para que su página web pueda invocar Digital Analytics Digital Recommendations a fin de recuperar un producto recomendado, primero debe configurar la oferta de IBM Interact con la información necesaria a fin de pasarla a Digital Recommendations.

Acerca de esta tarea

Para configurar una oferta a fin de enlazarla con Digital Recommendations, compruebe primero que se cumplen las condiciones siguientes:

- Asegúrese de que el servidor de ejecución de Interact se ha configurado y se ejecuta correctamente.
- Asegúrese de que el servidor de ejecución puede establecer una conexión con el servidor de Digital Recommendations, y que el cortafuegos no impide el establecimiento de una conexión web estándar de salida (puerto 80).

Para configurar una oferta para la integración con Digital Recommendations, realice los pasos siguientes.

Procedimiento

1. Cree o edite una oferta para Interact.

   Para obtener información sobre cómo crear y modificar ofertas, consulte la publicación *IBM Interact User's Guide*, y la documentación de IBM Campaign.
2. Además de los otros valores contenidos en la oferta, asegúrese de que la oferta incluye los atributos de oferta siguientes:
   - El URL (localizador universal de recursos) que enlaza con la imagen correspondiente a la oferta.
   - El URL que enlaza con la página de destino de la oferta.
   - Un ID de categoría de Digital Recommendations asociado a la oferta.

     Puede recuperar el ID de categoría manualmente desde la configuración de Digital Recommendations. Interact no puede recuperar valores de ID de categoría directamente.

   En la aplicación web de demostración que se incluye con la instalación de Interact, estos atributos de oferta se denominan `ImageURL`, `ClickThruURL` y `CategoryID`. Puede utilizar cualquier nombre que sea descriptivo para usted, siempre que la aplicación web concuerde con los valores esperados por la oferta.

   Por ejemplo, puede definir un oferta llamada "10PercentOff" que contiene esos atributos, donde el ID de categoría (tal como se recupera de la configuración de Digital Recommendations) es `PROD1161127`, el URL de la oferta es `http://www.example.com/success`, y el URL de la imagen que se debe mostrar para la oferta es `http://localhost:7001/sampleIO/img/10PercentOffer.jpg` (un URL que, en este caso, es local respecto del servidor de ejecución de Interact).
3. Defina las reglas de tratamiento de un canal interactivo para incluir esta oferta, y despliegue el canal interactivo de la forma habitual.

Resultados

La oferta está ahora definida con la información necesaria para la integración con Digital Recommendations. El trabajo restante para permitir que Digital

Recommendations proporcione recomendaciones de productos a Interact se realiza configurando páginas web para realizar las llamadas a la API apropiadas.

Cuando configure la aplicación web para presentar la página integrada a los visitantes, asegúrese de que los archivos siguientes estén incluidos en el directorio `WEB-INF/lib`:

- *Interact_Home*`/lib/interact_client.jar`, archivo necesario para gestionar las llamadas realizadas desde la página web a la API de Interact.
- *Interact_Home*`/lib/JSON4J_Apache.jar`, archivo necesario para gestionar los datos devueltos por la llamada a la API REST de Digital Recommendations, que devuelve datos con formato JSON.

Consulte "Utilización del proyecto de muestra de integración" para obtener más información sobre cómo presentar las ofertas a los clientes.

Utilización del proyecto de muestra de integración

Cada instalación de tiempo de ejecución de Interact incluye un proyecto de muestra que describe el proceso de la integración Digital Recommendations - Interact. El proyecto de muestra describe cómo crear una página web que llama a una oferta que contiene un ID de categoría, el cual se pasa a Digital Recommendations para recuperar una lista de productos recomendados para su presentación en los puntos de interacción de la página.

Descripción general

Puede utilizar el proyecto de muestra tal como se proporciona si desea probar el proceso de integración, o puede utilizar el proyecto como punto de partida para desarrollar sus propias páginas personalizadas. El proyecto de muestra se encuentra en el archivo siguiente:

directorio_inicio_Interact`/samples/IntelligentOfferIntegration/MySampleStore.jsp`

Este archivo, además de contener un ejemplo completo funcional del proceso de integración, también contiene comentarios extensos que describen qué debe configurar en Interact, qué debe personalizar en el archivo `.jsp` , y cómo desplegar la página correctamente para ejecutarla con la instalación.

MySampleStore.jsp

Para su comodidad, el archivo MySampleStore.jsp se muestra aquí. Esposible que esta muestra se actualice en releases subsiguientes de Interact; utilice pues el archivo que se incluye con la instalación como punto de partida para cualquier ejemplo que necesite.

```
<!--
# ********************************************************************************
#  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());
    }
%>
```

Apéndice F. Integración de Interact y Digital Data Exchange

Con Digital Data Exchange, el sitio web se puede enlazar a Interact para proporcionar un potente motor de ejecución Omni-Channel que entrega las mejores ofertas a los canales óptimos y evoluciona (aprende) desde los comentarios de ofertas para aumentar de forma continuada la efectividad del marketing.

Puede utilizar esta herramienta si el equipo de marketing utiliza Interact para la gestión de ofertas de Omni-Channel y desea ampliar estas ofertas de inteligencia personalizadas al sitio web.

IBM Digital Data Exchange integra IBM y soluciones de marketing de terceros con información de cliente digital a través de la API de sindicación de datos en tiempo real y una solución de gestión de etiquetas de nivel empresarial.

Sin IBM Digital Data Exchange, los usuarios de marketing dependen de la TI para enlazar Interact a su sitio web y llamar a la API Interact desde distintas páginas web. Con IBM Digital Data Exchange, los usuarios de marketing pueden ignorar la TI e ir directamente a IBM Digital Data Exchange para incluir etiquetas de IBM Digital Data Exchange en distintas páginas web.

Requisitos previos

Antes de poder utilizar la integración de Interact y Digital Data Exchange, asegúrese de que cumple los requisitos previos descritos en esta sección.

Asegúrese de que los requisitos previos siguientes son ciertos.

- Está familiarizado con la API de JavaScript Interact, tal como se documenta en la Guía de administrador y la ayuda en línea.
- Está familiarizado con las etiquetas y los grupos de páginas de Digital Data Exchange.
- Tiene una cuenta válida de Digital Data Exchange.
- Su archivo `interactapi.js` está alojada de forma pública y se puede acceder en la configuración de **Proveedor**.

Integración de IBM Interact con el sitio web a través de IBM Digital Data Exchange

Utilice estos pasos para integrarInteract con el sitio web a través de Digital Data Exchange.

Procedimiento

1. Especifique la ubicación del archivo `Interactapi.js`.
   a. Vaya hasta **Proveedores** > **Configuración de proveedor** en Digital Data Exchange.
   b. Seleccione IBM Interact en el desplegable **Proveedor**.
   c. En **Vía de acceso de biblioteca**, especifique el URL donde ha alojado `Interactapi.js`. No incluya el protocolo (http o https) en este URL.
   d. En **Vía de acceso en servlet Rest público**, añada la vía de acceso al servlet Rest.

2. Vaya hasta **Gestionar** > **Configuración global** en Digital Data Exchange para especificar el nombre de objeto para utilizar como identificador de página en **Identificador de página exclusivo**. Por ejemplo, puede establecer el nombre de objeto en `digitalData.pageInstanceID`.
3. Incluya el archivo `eluminate.js` y un identificador en la página web donde desea que Digital Data Exchange inserte las etiquetas. Deberá proporcionar a cada página web un identificador exclusivo, de forma que Digital Data Exchange distingue entre varias páginas.

   Por ejemplo, puede añadir el script siguiente a la página de inicio.

```
<!-- Establecimiento del identificador de página -->
          <script>
                  digitalData={pageInstanceID:"INTERACT_HomePage"};
          </script>

<!-- Incluir el script eluminate -->
         <script type="text/javascript" src="http://libs.
                  coremetrics.com/eluminate.js">
         </script>
         <script type="text/javascript">
                  cmSetClientID("51310000|INTERACTTEST",false,"data.
                  coremetrics.com",document.domain);
         </script>
```

4. En Digital Data Exchange, cree etiquetas, segmentos de código, funciones y otros elementos que desea añadir a la página web.
5. Cree grupos de páginas para definir qué desea que aparezca llenado en cada página.

   Consulte la Guía de usuario de IBM Digital Data Exchange si desea más información.

Etiquetas Interact en Digital Data Exchange

Utilice las etiquetas predeterminadas de Digital Data Exchange para definir variaciones de las etiquetas que son apropiadas para las páginas web donde se representan los datos de distintas ubicaciones. Una vez definidas, estas etiquetas se añaden a la lista de etiquetas de Interact. Las etiquetas pueden no tener campos para definir o es posible que no tengan campos de etiqueta necesarios y se puede utilizar directamente.

Las etiquetas de Interact siguientes están disponibles en Digital Data Exchange bajo **Etiquetas**.

- Finalizar sesión
- Obtener ofertas
- Cargar biblioteca
- Publicar evento
- Establecer audiencia
- Iniciar sesión

Para utilizar las etiquetas de Interact, edite las etiquetas para definir el Campo de etiqueta, Método, Nombre de objeto, Tipo de datos y Modificador para cada etiqueta de Interact.

Las etiquetas Publicar evento, Establecer audiencia e Iniciar sesiones aceptan campos de etiqueta personalizados. Utilice el icono Añadir del campo de etiqueta y, después, pulse el icono Editar para definir el parámetro personalizado. El proceso es el mismo que cualquier definición de parámetro con la excepción de

que el nombre del parámetro se puede editar y debe incluir el nombre del parámetro, dos puntos y el tipo de datos de parámetro. El orden del parámetro personalizado en la etiqueta se puede modificar con las flechas hacia arriba y abajo.

Las etiquetas también se pueden enlazar a funciones JavaScript u objetos HTML, de forma que se activan después de que se activa la función o en un evento de objeto HTML.

Si desea más información sobre cómo definir, enlazar y trabajar con etiquetas, consulte la Guía de usuario de IBM Digital Data Exchange.

Si desea casos de uso detallados de la integración de Interact y Digital Data Exchange, consulte https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Finalizar sesión

La etiqueta Finalizar sesión marca el final de una sesión web.

Loas campos de etiqueta siguientes están disponibles para la etiqueta Finalizar sesión.

Tabla 36. Etiquetas Finalizar sesión

Campo de etiqueta	Descripción
*ID de sesión	Identifica el ID de sesión.
Nombre de función de devolución de llamada cuando se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método finalizar sesión se realiza correctamente.
Nombre de función de devolución de llamada cuando no se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método finalizar sesión falla.

Es necesario el **Campo de etiqueta** marcado con un *.

Obtener ofertas

Utilice la etiqueta Obtener ofertas para solicitar ofertas del servidor de tiempo de ejecución.

Los campos de etiqueta siguientes están disponibles para la etiqueta Obtener ofertas.

Tabla 37. Etiquetas Obtener ofertas

Campo de etiqueta	Descripción
*ID de sesión	Identifica el ID de sesión.
*Nombre de punto de interacción	Identifica el nombre del punto de interacción al que hace referencia este método. Este nombre debe coincidir exactamente con el nombre del punto de interacción definido en el canal interactivo.
*Número solicitado	Identifica el número de ofertas solicitado.

Tabla 37. Etiquetas Obtener ofertas (continuación)

Campo de etiqueta	Descripción
Nombre de función de devolución de llamada cuando se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método obtener ofertas se realiza correctamente.
Nombre de función de devolución de llamada cuando no se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método obtener ofertas falla.

Es necesario el **Campo de etiqueta** marcado con un *.

La etiqueta Obtener ofertas se debe asignar a un grupo de páginas cuyo contenedor está establecido en `Default`.

Cargar biblioteca

La etiqueta Cargar biblioteca carga la biblioteca JavaScript de Interact en la sección de cabecera de la página.

La etiqueta Cargar biblioteca no tiene parámetros. Toma la ubicación de la biblioteca de `Vía de acceso de biblioteca` en **Configuración del proveedor**. Se debe incluir en un grupo de páginas utilizando un contenedor establecido en `Head` y se deberá ejecutar en cada página que tiene las etiquetas Interact.

Importante: Ninguna de las demás etiquetas funcionará si no está incluida la etiqueta Cargar biblioteca. El archivo interact.js no se carga, si esta etiqueta no está incluida.

Publicar evento

Utilice la etiqueta Publicar evento para ejecutar cualquier evento definido en el canal interactivo.

Los campos de etiqueta siguientes están disponibles para la etiqueta Publicar evento.

Tabla 38. Etiquetas Publicar evento

Campo de etiqueta	Descripción
*ID de sesión	Identifica el ID de sesión.
*Nombre de evento	Identifica el nombre del evento. El nombre del evento debe coincidir con el nombre del evento tal como está definido en el canal interactivo. Este nombre es sensible a mayúsculas y minúsculas.
Nombre de función de devolución de llamada cuando se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método publicar evento se realiza correctamente.
Nombre de función de devolución de llamada cuando no se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método publicar eventos falla.

Es necesario el **Campo de etiqueta** marcado con un *.

Los parámetros opcionales se pueden añadir con la característica de campo de etiqueta personalizada. Los nombres de etiqueta personalizados deben estar formados por el nombre del parámetro, dos puntos y el tipo de datos.

Establecer audiencia

Utilice la etiqueta Establecer audiencia para establecer el ID de audiencia y el nivel para un visitante.

Los campos de etiqueta siguientes están disponibles para la etiqueta Establecer audiencia.

Tabla 39. Etiquetas Establecer audiencia

Campo de etiqueta	Descripción
*ID de sesión	Identifica el ID de sesión.
*ID de audiencia	Identifica el ID de audiencia. Los nombres deben coincidir con los nombres de columna física de cualquier tabla que contiene el ID de audiencia. El ID de audiencia no puede contener más de 17 dígitos significativos. Si el ID de audiencia tiene más de 17 dígitos significativos se debe particionar o se debe cambiar el ID de audiencia en una serie.
*Nivel de audiencia	Define el nivel de audiencia.
Nombre de función de devolución de llamada cuando se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método establecer audiencia se realiza correctamente.
Nombre de función de devolución de llamada cuando no se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método establecer audiencia falla.

Es necesario el **Campo de etiqueta** marcado con un *****.

Los parámetros opcionales se pueden añadir con la característica de campo de etiqueta personalizada. Los nombres de etiqueta personalizados deben estar formados por el nombre del parámetro, dos puntos y el tipo de datos.

Iniciar sesión

La etiqueta Iniciar sesión crea y define una sesión web.

Los campos de etiqueta siguientes están disponibles para la etiqueta Iniciar sesión.

Tabla 40. Etiquetas Iniciar sesión

Campo de etiqueta	Descripción
*ID de sesión	Identifica el ID de sesión.
*Canal de interacción	Define el nombre del canal de interacción al que hace referencia esta sesión. Este nombre debe coincidir con el nombre del canal de interacción definido de forma exacta en la Campaña.
*ID de audiencia	Identifica el ID de audiencia. Los nombres deben coincidir con los nombres de columna física de cualquier tabla que contiene el ID de audiencia.
*Nivel de audiencia	Define el nivel de audiencia.

Tabla 40. Etiquetas Iniciar sesión (continuación)

Campo de etiqueta	Descripción
*Se basa en una sesión existente	Define si esta sesión utiliza una sesión nueva o existente
*Depurar	Habilita o inhabilita información de depuración.
Nombre de función de devolución de llamada cuando se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método iniciar sesión se realiza correctamente.
Nombre de función de devolución de llamada cuando no se realiza correctamente	Define el nombre de la función que se va a llamar cuando el método iniciar sesión falla.

Es necesario el **Campo de etiqueta** marcado con un *.

Los parámetros opcionales se pueden añadir con la característica de campo de etiqueta personalizada. Los nombres de etiqueta personalizados deben estar formados por el nombre del parámetro, dos puntos y el tipo de datos.

La etiqueta Iniciar sesión se debe asignar a un grupo de páginas cuyo contenedor está establecido en `Default`.

Valores de etiqueta de ejemplo

Este ejemplo muestra una configuración sencilla de los valores de etiqueta Iniciar sesión, Publicar eventos, Obtener ofertas y Finalizar sesión.

Para cualquier etiqueta, puede obtener los valores de campo de etiqueta en la cookie con el método de cookie o en el objeto JavaScript con el método javascriptobject.

Estas etiquetas soportan parámetros adicionales que este ejemplo sencillo no muestra. Puede encontrar más información sobre los parámetros adicionales en la Guía de usuario de IBM Digital Data Exchange.

Si desea casos de uso detallados de la integración de Interact y Digital Data Exchange, consulte https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Valores de ejemplo de la etiqueta Iniciar sesión

Pulse **Etiquetas** > **IBM Tags** > **IBM Interact** > **Tipo: Iniciar sesión** para crear una etiqueta Iniciar sesión. Edite la etiqueta con los valores siguientes.

Valores de ID de sesión

- **Método**: `Constante`
- **Constante**: `5555`
- **Tipo de datos**: `Serie`
- **Modificador**: `<nulo>`

Valores de Canal interactivo

- **Método**: `Constante`
- **Constante**: `WSCDemo`

- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de ID de audiencia
- **Método**: Constante
- **Constante**: USERS_ID,2002,numérico
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Nivel de audiencia
- **Método**: Constante
- **Constante**: WSCUserId
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Se basa en la sesión existente
- **Método**: Constante
- **Constante**: Falso
- **Tipo de datos**: Booleano
- **Modificador**: <nulo>

Depurar
- **Método**: Constante
- **Constante**: Verdadero
- **Tipo de datos**: Booleano
- **Modificador**: <nulo>

Valores de Nombre función de devolución de llamada cuando se realiza correctamente
- **Método**: Sin asignar
- **Valor**: <nulo>

Valores de Nombre de función de devolución de llamada cuando falla
- **Método**: Sin asignar
- **Valor**:<nulo>

Valores de ejemplo de la etiqueta Obtener ofertas

Pulse **Etiquetas** > **IBM Tags** > **IBM Interact** > **Tipo: Obtener ofertas** para crear una etiqueta Obtener ofertas. Edite la etiqueta con los valores siguientes.

Valores de ID de sesión
- **Método**: Constante
- **Constante**: 5555
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Nombre de punto de interacción
- **Método**: Constante

- **Constante**: AuroraHomepageHeaderBannerLeft
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Número solicitada
- **Método**: Constante
- **Constante**: 1
- **Tipo de datos**: entero
- **Modificador**: <nulo>

Valores de Nombre función de devolución de llamada cuando se realiza correctamente
- **Método**: Constante
- **Constante**: onOfferReturnSuccess
- **Tipo de datos**: serie
- **Modificador**: <nulo>

Valores de Nombre de función de devolución de llamada cuando falla
- **Método**: Constante
- **Constante**: onOfferReturnError
- **Tipo de datos**: serie
- **Modificador**: <nulo>

Valores de ejemplo de la etiqueta Publicar evento

Pulse **Etiquetas** > **IBM Tags** > **IBM Interact** > **Tipo: Publicar evento** para crear una etiqueta Publicar evento. Edite la etiqueta con los valores siguientes.

Valores de ID de sesión
- **Método**: Constante
- **Constante**: 5555
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Nombre de evento
- **Método**: Constante
- **Constante**: ACCEPTOFFER
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Nombre función de devolución de llamada cuando se realiza correctamente
- **Método**: Constante
- **Constante**: onSuccessTestFunction
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Nombre de función de devolución de llamada cuando falla
- **Método**: Constante

- **Constante**: onErrorTestFunction
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Campo de parámetro adicional
- **Campo de etiqueta**: UACIOfferTrackingCode:string
- **Método**: JavaScriptObject
- **Nombre de objeto**: oa.treatmentCode
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de ejemplo de la etiqueta Finalizar sesión

Pulse **Etiquetas** > **IBM Tags** > **IBM Interact** > **Tipo: Finalizar sesión** para crear una etiqueta Finalizar sesión. Edite la etiqueta con los valores siguientes.

Valores de ID de sesión
- **Método**: Constante
- **Constante**: 5555
- **Tipo de datos**: Serie
- **Modificador**: <nulo>

Valores de Nombre función de devolución de llamada cuando se realiza correctamente
- **Método**: Sin asignar
- **Valor**: <nulo>

Valores de Nombre de función de devolución de llamada cuando falla
- **Método**: Sin asignar
- **Valor**:<nulo>

Funciones de ejemplo

Para las funciones utilizadas para los valores de Nombre de función de devolución de llamada cuando se realiza correctamente o Nombre de función de devolución de llamada cuando falla, solo tendrá que especificar el nombre de la función cuando cree una etiqueta nueva si la función ya está presente en la página web.

También puede utilizar el programa de utilidades de Digital Data Exchange para crear funciones y añadirlas a las páginas web.

El ejemplo siguiente muestra cómo mostrar una oferta devuelta de Interact en la página web. Debe incluir este script en la página o utilizar el fragmento de código de Digital Data Exchange para inyectarlo.

```
<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>
```

Verificar la configuración de integración

Utilice la herramienta de prueba Digital Data Exchange y el archivo `Interact.log` para solucionar los problemas de configuración.

Puede utilizar la herramienta de pruebas Digital Data Exchange para comprobar la enciclopedia para ver si la configuración funciona como está previsto. Para abrirla herramienta de pruebas, pulse **Despliegue** > **Herramienta de pruebas** en Digital Data Exchange.

Consulte la Guía de usuario de IBM Digital Data Exchange si desea más información sobre la herramienta de pruebas.

Puede consultar el archivo `Interact.log` para ver detalles sobre las distintas llamadas de API Interact que se han realizado. Añada la función de devolución de llamada si se realiza correctamente y la función de devolución de llamada si falla a cada etiqueta para depurar las distintas llamadas.

Antes de contactar con el soporte técnico de IBM

Si encuentra un problema que no puede resolver consultando la documentación, el contacto de soporte designado por la empresa puede realizar una llamada al soporte técnico de IBM. Utilice estas directrices para asegurarse de que su problema se resuelve de forma eficiente y satisfactoria.

Si usted no es una de las personas responsables del contacto con el servicio de soporte técnico en su empresa, póngase en contacto con su administrador de IBM para obtener información.

Nota: El soporte técnico no escribe ni crea scripts de API. Para obtener ayuda al implementar nuestras ofertas de API, contacte con IBM Professional Services.

Información a recopilar

Antes de ponerse en contacto con el soporte técnico de IBM, recopile la información siguiente:

- Una breve descripción de la naturaleza del problema.
- Los mensajes de error detallados que ve cuando se produce el problema.
- Detalles de los pasos para reproducir el problema.
- Archivos de registro, archivos de sesión, archivos de configuración y archivos de datos relacionados con el problema.
- Información sobre el producto y el entorno del sistema del , que puede obtener tal como se describe en "Información del sistema".

Información de sistema

Cuando llame al soporte técnico de IBM, es posible que se le pida que proporcione información sobre su entorno.

Si el problema no le impide conectarse, gran parte de esta información está disponible en la página Acerca de, que proporciona información sobre las aplicaciones IBM instaladas.

Puede acceder a la página Acerca de seleccionando **Ayuda > Acerca de**. Si la página Acerca de no es accesible, consulte en el archivo `version.txt` que se encuentra en el directorio de instalación de la aplicación.

Información de contacto para el soporte técnico de IBM

Para conocer las formas de ponerse en contacto con el soporte técnico de IBM, consulte el sitio web de soporte técnico del producto IBM: (http://www.ibm.com/support/entry/portal/open_service_request).

Nota: Para especificar una solicitud de soporte, debe iniciar la sesión con una cuenta de IBM. Esta cuenta debe estar vinculada a su número de cliente de IBM Para obtener más información sobre cómo asociar la cuenta a su número de cliente de IBM, consulte **Support Resources>Entitled Software Support** en el portal de soporte.

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en EE.UU.

Es posible que IBM no ofrezca en otros países los productos, servicios o características que se describen en este documento. Consulte al representante local de IBM para obtener información sobre los productos y servicios disponibles actualmente en su localidad. Cualquier referencia a un producto, programa o servicio de IBM no pretende indicar o implicar que sólo se puede utilizar el producto, programa o servicio de IBM. Se puede utilizar en su lugar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ningún derecho de propiedad intelectual de IBM. Sin embargo, es responsabilidad del cliente evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o aplicaciones de patente pendientes que afecten a los temas tratados en este documento. La entrega de este documento no le otorga ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Para consultas sobre licencias relativas a la información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM en su país o envíe las consultas, por escrito, a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde estas disposiciones sean incompatibles: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUYENDO PERO NO LIMITÁNDOSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO. Algunos países no permiten la renuncia a garantías explícitas o implícitas en determinadas transacciones, por lo que puede que esta declaración no sea aplicable en su caso.

Esta información puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información incluida en este documento; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede realizar en cualquier momento mejoras o cambios en los productos o programas descritos en esta publicación sin previo aviso.

Cualquier referencia en esta información a sitios web que no sean de IBM se proporciona, únicamente, a efectos de comodidad y no sirve, en modo alguno, de endoso de dichos sitios web. El contenido de esos sitios web no forma parte del contenido de este producto de IBM, por lo que la utilización de dichos sitios es responsabilidad del usuario.

IBM podría usar o distribuir del modo que considere adecuado cualquier información que usted suministre, sin contraer por ello obligación alguna con usted.

Los titulares de licencias de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de información que se haya intercambiado, deben ponerse en contacto con:

IBM Corporation
B1WA LKG1
550 King Street
Littleton, MA 01460-1250
EE.UU.

Esta información puede estar disponible, sujeta a los términos y condiciones adecuados, incluido en algunos casos, el pago de una tasa.

IBM proporciona el programa bajo licencia descrito en este documento y todo el material bajo licencia disponible para el mismo, de acuerdo a lo estipulado en los términos del Acuerdo de cliente de IBM, el Acuerdo internacional de licencias de programas de IBM o cualquier acuerdo equivalente entre ambas partes.

Cualquier dato que se encuentre en este documento se ha determinado en un ambiente controlado. Por lo tanto, los resultado obtenido en otros entornos operativos podrían variar significativamente. Tal vez se hayan realizado mediciones en sistemas que estén en fase de desarrollo y no existe ninguna garantía de que esas mediciones vayan a ser iguales en los sistemas disponibles en el mercado. Además, algunas medidas se podrían haber estimado en extrapolación. Los resultados reales pueden variar. Los usuarios de este documento debería verificar los datos aplicables para sus entornos específicos.

La información relacionada con productos que no son de IBM se ha obtenido de los proveedores de dichos productos de sus anuncios publicados o de otras fuentes de disponibilidad pública. IBM no ha probado necesariamente esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni otras afirmaciones referentes a productos que no sean de IBM. Las preguntas relacionadas con las funciones de los productos que no son de IBM deberán dirigirse a los proveedores de estos productos.

Todas las sentencias relacionadas con la futura dirección de IBM o intento están sujetas al cambio o retirada sin previo aviso y sólo representan objetivos y metas.

Todos los precios de IBM que se muestran son precios de distribuidor recomendados por IBM, corresponden al momento actual y están sujetos a cambios sin aviso previo. Los precios de concesionario pueden variar.

Esta información contiene ejemplos de datos e informes utilizados en operaciones empresariales diarias. Para ilustrarlas de la forma más completa posible, los ejemplos pueden incluir nombres de personas, empresas, marcas y productos.

Todos estos nombres son ficticios y cualquier similitud a los nombres y direcciones que haya utilizado una empresa real es pura coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de muestra en lenguaje fuente, que se utilizan para complementar las explicaciones relacionadas con las técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier manera sin realizar ningún pago a IBM, a fin de desarrollar, utilizar, comercializar y distribuir programas de aplicación que se adecuen a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Estos ejemplos no se han probado de forma exhaustiva bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni dar por sentada la fiabilidad, la utilidad ni el funcionamiento de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin garantía de ninguna clase. IBM no será responsable de los daños debidos al uso de los programas de ejemplo.

Si está viendo esta copia software de la información, es posible que las fotografías y las ilustraciones en color no aparezcan.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., registradas en muchas jurisdicciones de todo el mundo. Otros nombres de producto y servicio pueden ser marcas registradas de IBM u otras empresas. Encontrará la lista actual de las marcas comerciales de IBM en el sitio web on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Consideraciones sobre la política de privacidad y los términos de uso

Los productos software de IBM Software, incluido el software como una solución de servicio, ("Ofertas de Software") pueden utilizar cookies u otras tecnologías para recopilar información de utilización de producto, para ayudar a mejorar la experiencia del usuario final, para adaptar las interacciones con el usuario final o para otros fines. Una cookie es un elemento de datos que un sitio web puede enviar al navegador, que a continuación se puede almacenar en el sistema como una etiqueta que identifica el sistema. En muchos casos, estas cookies no recopilan información personal. Si utiliza una Oferta de software que le permite recopilar información personal mediante cookies y tecnologías similares, a continuación le ofrecemos información específica.

Dependiendo de las configuraciones desplegadas, esta Oferta de software puede utilizar cookies de sesión y persistentes que recopilen el nombre de cada usuario y otra información personal para fines de gestión de sesiones, utilización de usuario mejorada u otros fines funcionales o de seguimiento de uso. Estas cookies pueden inhabilitarse, pero si se inhabilitan también se eliminará la funcionalidad que habilitan.

Distintas jurisdicciones regulan la recopilación de información personal mediante cookies y tecnologías similares. Si las configuraciones desplegadas para esta Oferta de Software le proporcionan como cliente la posibilidad de recopilar información personal sobre usuarios a través de cookies u otras tecnologías, debería buscar su

propio asesoramiento legal en relación a todas las leyes aplicables a dicha recopilación de datos, incluidos los requisitos para proporcionar avisos y el consentimiento cuando sea lo propio.

IBM requiere que los Clientes (1) proporcionen un enlace claro y visible a los términos de uso del sitio web del Cliente (por ejemplo, política de privacidad) que incluya un enlace a las prácticas de uso y recopilación de datos de IBM y del cliente; (2) notifiquen que IBM coloca, en nombre del Cliente, cookies y balizas web/gifs claras en el sistema del visitante, junto con una explicación de la finalidad de dicha tecnología; y (3) en la medida requerida por ley, obtengan el consentimiento de los visitantes del sitio web antes de la colocación de cookies y balizas web/gifs por parte del Cliente o IBM en nombre del Cliente, en dispositivos del visitante del sitio web

Si desea obtener más información sobre la utilización de las distintas tecnologías, incluidas las cookies, para estos propósitos, consulte la declaración de privacidad en línea de IBM en http://www.ibm.com/privacy/details/us/en, en la sección que se titula "Cookies, Web Beacons and Other Technologies".

Impreso en España