Versione 10 Release 1
Novembre 2017

IBM Interact Guida per l'amministratore

Note
Before using this information and the product it supports, read the information in “Notices” a pagina 383.

This edition applies to version 10, release 1, modification 0 of IBM Interact and to all subsequent releases and modifications until otherwise indicated in new editions.

Indice

Capitolo 1. Amministrazione di IBM Interact

Quando si amministra Interact, si configurano e si gestiscono utenti e ruoli, origini dati e funzioni facoltative del prodotto. Si esegue inoltre il monitoraggio e la manutenzione degli ambienti di progettazione e di runtime. Sono disponibili per l'uso API (application programming interface), specifiche per il prodotto.

L'amministrazione di Interact è composta da diverse attività. Queste attività possono includere, ma non solo:

- Gestione di utenti e ruoli
- Gestione delle origini dati
- Configurazione delle funzioni di presentazione delle offerte facoltative di Interact
- Monitoraggio e manutenzione delle prestazioni dell'ambiente di runtime

Prima di iniziare ad amministrare Interact, sarebbe opportuno conoscere alcuni concetti chiave sulla modalità di funzionamento di Interact, per semplificare tali attività. Le sezioni che seguono, descrivono le attività di amministrazione associate ad Interact.

La seconda parte della guida dell'amministratore descrive le API disponibili con Interact:

- API di Interact
- API ExternalCallout
- API di apprendimento

Concetti chiave di Interact

IBM® Interact è un motore interattivo che invia le offerte di marketing personalizzate a vari destinatari.

In questa sezione vengono descritti alcuni dei concetti chiave che è necessario comprendere prima di utilizzare Interact.

Livelli destinatario

Un livello destinatario è una raccolta di ID che possono essere utilizzati come obiettivo da una campagna. È possibile definire i livelli destinatario per determinare la serie corretta di destinatari per la campagna.

Ad esempio, una serie di campagne può utilizzare i livelli destinatario "Household", "Prospect", "Customer" e "Account." Ciascuno di questi livelli rappresenta una determinata vista dei dati di marketing disponibili per una campagna.

I livelli destinatario sono solitamente organizzati in modo gerarchico. Utilizzando gli esempi precedenti:

- Nucleo familiare è in alto nella gerarchia e ciascun nucleo familiare può contenere più clienti e uno o più potenziali clienti.
- "Cliente" è il successivo livello della gerarchia e a ciascun cliente possono essere assegnati più account.

- "Account" è il livello inferiore della gerarchia.

Altri esempi più complessi di gerarchie destinatario sono disponibili negli ambienti business-to-business, dove possono esistere livelli destinatario per business, società, divisioni, gruppi, individui, account e così via.

Questi livelli destinatario possono avere relazioni differenti l'uno con l'altro, ad esempio relazioni di tipo uno-a-uno, molti-a-uno o molti-a-molti. Mediante la definizione dei livelli destinatario, è possibile rappresentare questi concetti in Campaign in modo che gli utenti possano gestire le relazioni tra i diversi destinatari ai fini della determinazione degli obiettivi. Ad esempio, sebbene possano esistere più potenziali clienti per nucleo familiare, è possibile limitare il servizio di mailing a un solo potenziale cliente per nucleo familiare.

Ambiente di progettazione

Utilizzare l'ambiente di progettazione per configurare vari componenti di Interact e distribuirli nell'ambiente runtime.

L'ambiente di progettazione è la sede in cui è possibile completare la maggior parte delle operazioni di configurazione di Interact. Nell'ambiente di progettazione, definire eventi, punti di interazione, segmenti smart e regole di trattamento. Dopo aver configurato questi componenti, distribuirli nell'ambiente runtime.

L'ambiente di progettazione è installato con l'applicazione Web Campaign.

Eventi

Un evento è un'azione eseguita da un visitatore, che attiva un'azione in un ambiente runtime. Esempi di evento possono essere: l'inserimento di un visitatore in un segmento, la presentazione di un'offerta o la registrazione dei dati.

Gli eventi vengono creati prima in un canale interattivo e poi vengono attivati da una chiamata all'API di Interact mediante il metodo `postEvent`. Un evento può dare luogo ad una o più delle seguenti azioni definite nell'ambiente di progettazione di Interact.

- **Attiva risegmentazione**: l'ambiente runtime esegue nuovamente tutti i diagrammi di flusso interattivi per il livello destinatario corrente associato al canale interattivo, utilizzando i dati correnti nella sessione del visitatore.

  Quando si progetta l'interazione, a meno che non venga indicato un diagramma di flusso specifico, un'azione di risegmentazione esegue nuovamente tutti i diagrammi di flusso interattivi associati a questo canale interattivo con il livello destinatario corrente e le richieste di offerte attendono il completamento di tutti i diagrammi di flusso. L'eccessiva risegmentazione all'interno di una singola visita può influire sulle prestazioni del touchpoint in una modalità visibile al cliente.

  Assegnare il cliente a nuovi segmenti dopo che una significativa quantità di nuovi dati viene aggiunta all'oggetto sessione di runtime come ad esempio nuovi dati provenienti da richieste dall'API Interact (come la modifica del destinatario) o da azioni del cliente (come ad esempio l'aggiunta di nuovi elementi ad un elenco di preferenze o un carrello della spesa).
- **Registra contatto offerta**: l'ambiente di runtime contrassegna le offerte consigliate per il servizio di database per registrare le offerte nella cronologia dei contatti.

  Per le integrazioni, registrare il contatto dell'offerta nella stessa cella in cui si richiedono le offerte per rendere minimo il numero di richieste tra il touchpoint e il server di runtime.

Se il touchpoint non restituisce i codici trattamento per le offerte che Interact ha presentato al visitatore, l'ambiente di runtime registra l'ultimo elenco delle offerte consigliate.

- **Registra accettazione offerta**: l'ambiente di runtime contrassegna l'offerta selezionata per il servizio di database per registrare la cronologia delle risposte.
- **Registra rifiuto offerta**: l'ambiente di runtime contrassegna l'offerta selezionata per il servizio di database per registrare la cronologia delle risposte.
- **Attiva espressione utente**: un'*azione espressione* è un'azione che è possibile definire utilizzando le macro Interact, tra cui le funzioni, le variabili e gli operatori, incluso EXTERNALCALLOUT. È possibile assegnare il valore restituito dall'espressione all'attributo del profilo.

  Quando si fa clic sull'icona di modifica accanto a Attiva espressione utente, viene visualizzata la finestra di dialogo di modifica Espressione utente standard ed è possibile utilizzare questa finestra di dialogo per specificare il livello destinatario, il nome campo facoltativo a cui assegnare i risultati e la definizione dell'espressione stessa.
- **Attiva eventi**: è possibile utilizzare l'azione Attiva eventi per immettere un nome evento che si desidera attivare mediante questa azione. Se si immette un evento che è già definito, tale evento verrà attivato quando viene eseguita questa azione. Se il nome evento immesso non esiste, questa azione darà luogo alla creazione di tale evento utilizzando l'azione specificata.

È inoltre possibile utilizzare gli aventi per attivare le azioni definite dal metodo `postEvent`, tra cui la registrazione dei dati in una tabella, compresi i dati per l'apprendimento o l'attivazione di singoli diagrammi di flusso.

Gli eventi possono essere organizzati in categorie nell'ambiente di progettazione a seconda delle proprie esigenze. Le categorie non hanno alcuno scopo funzionale nell'ambiente di runtime.

Canali interattivi

Utilizzare i canali interattivi in Interact per coordinare tutti gli oggetti, i dati e le risorse del server coinvolti nel marketing interattivo.

Un canale interattivo è una rappresentazione in Campaign di un touchpoint in cui il metodo dell'interfaccia è una finestra di dialogo interattiva. Questa rappresentazione software viene utilizzata per coordinare tutti gli oggetti, i dati e le risorse server coinvolti nel marketing interattivo.

Un canale interattivo è uno strumento utilizzato per definire punti di interazione ed eventi. È anche possibile accedere a report per un canale interattivo dalla scheda dell'analisi di tale canale interattivo.

I canali interattivi contengono anche assegnazioni di server runtime di produzione e di gestione temporanea. È possibile creare diversi canali interattivi per organizzare eventi e punti di interazione, se si dispone di una sola serie di server runtime di produzione e di gestione temporanea o per dividere gli eventi e i punti di interazione in base al sistema rivolto al cliente.

Diagrammi di flusso interattivi

Utilizzare i diagrammi di flusso interattivi per dividere i clienti in segmenti e assegnare un profilo ad un segmento.

Un diagramma di flusso interattivo è correlato, sebbene sia leggermente diverso, ad un diagramma di flusso del batch di Campaign. I diagrammi di flusso interattivi svolgono la stessa funzione importante dei diagrammi di flusso batch, ovvero dividono i clienti in gruppi denominati segmenti. Nel caso dei diagrammi di flusso interattivi, tuttavia, i gruppi sono segmenti smart. Interact utilizza questi diagrammi di flusso interattivi per assegnare un profilo a un segmento quando un evento comportamentale o di sistema indica che occorre nuovamente eseguire la segmentazione di un visitatore.

I diagrammi di flusso interattivi contengono un sottoinsieme di processi dei diagrammi di flusso batch, nonché alcuni processi specifici dei diagrammi di flusso interattivi.

Nota: i diagrammi di flusso interattivi possono essere creati solo in una sessione Campaign.

Punti di interazione

Un punto di interazione è il luogo nel touchpoint in cui si desidera presentare un'offerta.

I punti di interazione prevedono un contenuto predefinito nel caso in cui l'ambiente runtime non disponga di contenuto appropriato da presentare. I punti di interazione possono essere organizzati in zone.

Offerte

Un'offerta rappresenta un solo messaggio di marketing che può essere recapitato in diversi modi.

In Campaign, si creano offerte che è possibile utilizzare in una o più campagne.

Le offerte possono essere riutilizzate come illustrato di seguito.

- In campagne differenti
- In momenti differenti
- Per gruppi differenti di persone (celle)
- Come "versioni" differenti, modificando i campi parametrizzati dell'offerta

Le offerte vengono assegnate ai punti di interazione nei touchpoint presentati ai visitatori.

Profili

Un profilo è la serie di dati del cliente utilizzati dall'ambiente runtime. Questi dati possono essere un sottoinsieme di dati del cliente disponibili nel database personalizzato, dati raccolti in tempo reale o una combinazione di entrambi.

I dati del cliente vengono utilizzati per gli scopi riportati di seguito.

- Assegnare un cliente a uno o più segmenti smart in scenari di interazione in tempo reale.

  È necessario una serie di dati del profilo per ciascun livello destinatario in base al quale si desidera eseguire la segmentazione. Ad esempio, se si esegue la segmentazione per ubicazione, è possibile includere solo il codice postale del cliente tra tutte le informazioni relative all'indirizzo di cui si dispone.
- Personalizzare le offerte

- Come attributi da registrare per l'apprendimento

  Ad esempio, è possibile configurare Interact per monitorare lo stato civile di un visitatore e il numero di visitatori di ciascuno stato che accetta una determinata offerta. L'ambiente runtime può quindi utilizzare tali informazioni per ridefinire la selezione di offerte.

Questi dati sono di sola lettura per l'ambiente runtime.

Ambiente runtime

L'ambiente runtime si collega al touchpoint ed esegue le interazioni. L'ambiente runtime può essere costituito da uno o molti server runtime collegati a un touchpoint.

L'ambiente runtime utilizza le informazioni distribuite dall'ambiente di progettazione insieme all'API di Interact per presentare le offerte al touchpoint.

Sessioni runtime

Per ciascun visitatore del touchpoint esiste una sessione runtime sul server runtime. Questa sessione contiene tutti i dati del visitatore che l'ambiente runtime utilizza per assegnare i visitatori ai segmenti e proporre le offerte.

Una sessione di runtime viene creata quando si utilizza la chiamata `startSession`.

Touchpoint

Un touchpoint è un'applicazione o un luogo in cui è possibile interagire con un cliente. Un touchpoint può essere un canale in cui il cliente avvia il contatto (interazione in ingresso) o da dove si contatta il cliente (interazione in uscita).

Esempi comuni sono le applicazioni del call center e siti Web. Utilizzando l'API di Interact, è possibile integrare Interact con i propri touchpoint per presentare le offerte ai clienti in base all'azione eseguita dai clienti nel touchpoint. I touchpoint sono anche chiamati sistemi CFS (Client-Facing System, sistemi rivolti al cliente).

Regole di trattamento

Le regole di trattamento assegnano un'offerta a un segmento smart. Queste assegnazioni sono ulteriormente limitate dalla zona definita dal cliente associata all'offerta nella regola di trattamento.

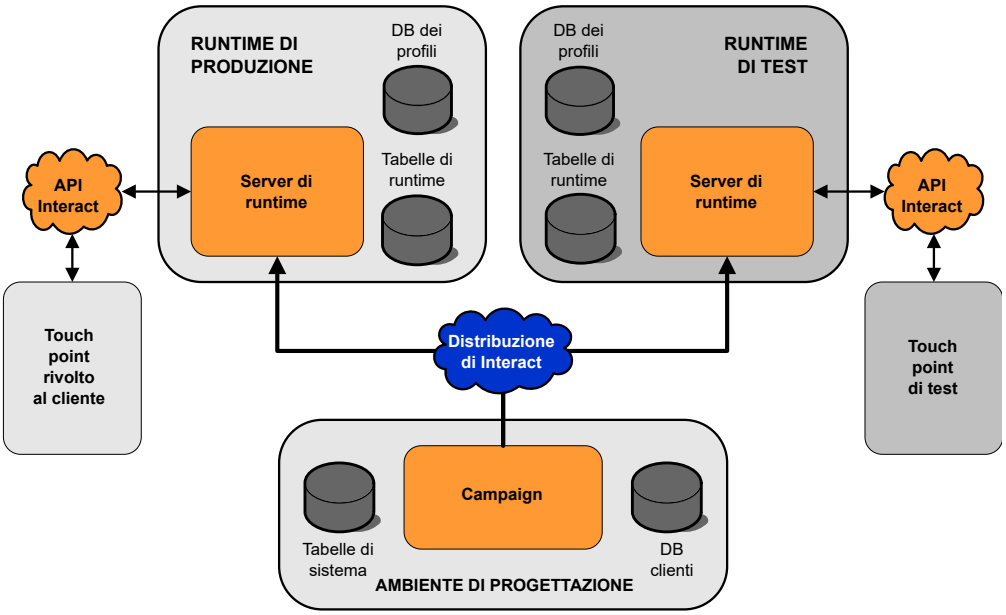
Ad esempio, è possibile assegnare una serie di offerte a un segmento smart nella zona "login", ma un insieme di offerte diverso per lo stesso segmento nella zona "dopo l'acquisto". Le regole di trattamento sono definite nella scheda della strategia di interazione di una campagna.

Ciascuna regola di trattamento prevede anche un punteggio marketing. Se un cliente è assegnato a più segmenti e, quindi, può ricevere più offerte, i punteggi marketing consentono di definire l'offerta che Interact deve consigliare. Le offerte consigliate dall'ambiente runtime possono essere influenzate da un modulo di apprendimento, un elenco di soppressione delle offerte e assegnazioni di offerte globali e individuali.

Architettura Interact

L'ambiente Interact è composto almeno da due componenti principali, l'ambiente di progettazione e l'ambiente di runtime. È possibile che siano presenti anche server di runtime per l'esecuzione di test facoltativi.

Nella figura seguente viene mostrata una panoramica dell'architettura di alto livello.



L'ambiente di progettazione consente di eseguire la maggior parte delle operazioni di configurazione di Interact. L'ambiente di progettazione è installato con l'applicazione Web di Campaign e fa riferimento alle tabelle di sistema di Campaign e ai database dei clienti. Mediante l'ambiente di progettazione, si possono definire i punti di interazione e gli eventi da utilizzare con l'API.

Dopo aver progettato e configurato la modalità in cui si desidera che l'ambiente di runtime gestisce le interazioni con i clienti, si distribuiscono tali dati al gruppo di server di staging per il test o a un gruppo di server di runtime di produzione per interazioni in tempo reale con i clienti.

L'API di Interact fornisce la connessione tra il touchpoint e il server di runtime. Si fa riferimento ad oggetti (punti di interazione ed eventi) creati nell'ambiente di progettazione mediante l'API di Interact e si utilizzano tali oggetti per richiedere informazioni al server di runtime.

Considerazioni sulla rete Interact

Un'installazione di produzione di Interact include almeno due macchine. In un ambiente di produzione di grandi dimensioni, con diversi database distribuiti e server di runtime Interact, l'installazione potrebbe estendersi a dozzine di macchine.

Per prestazioni ottimali, ci sono vari requisiti di topologia di rete da prendere in considerazione.

- Se l'implementazione dell'API di Interact inizia e termina le sessioni nella stessa chiamata, ad esempio:

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

  non è necessario abilitare la permanenza della sessione (sessioni permanenti) tra il bilanciamento del carico e i server di runtime Interact. È possibile configurare la gestione sessioni dei server di runtime Interact per il tipo di cache locale.
- Se l'implementazione dell'API di Interact utilizza più chiamate per iniziare e terminare sessioni, ad esempio:

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

  e si utilizza un bilanciamento del carico per i propri server di runtime Interact, si dovrebbe abilitare qualche tipo di permanenza il bilanciamento del carico (nota anche come sessioni permanenti). Se ciò non è possibile o se non si utilizza un bilanciamento del carico, configurare la gestione sessioni dei server Interact per un `cacheType` distribuito (Distributed). Se si sta utilizzando una cache distribuita, tutti i server di runtime Interact devono essere in grado di comunicare tramite multicast. Potrebbe essere necessario ottimizzare la propria rete in modo che la comunicazione tra i server Interact che utilizzano lo stesso indirizzo IP multicast e la stessa porta non ostacoli le prestazioni del sistema. Un bilanciamento del carico con le sessioni permanenti offre delle prestazioni migliori rispetto all'utilizzo di una cache distribuita.
- La memorizzazione in cache tra più gruppi di server non è supportata.
- Collocare i propri server Interact dell'ambiente di runtime, Marketing Platform, i bilanciamenti del carico e il touchpoint nella stessa ubicazione geografica per ottenere le migliori prestazioni. La fase di progettazione e il runtime possono essere situati in ubicazioni geografiche, ma si deve prevedere un distribuzione lenta.
- Stabilire una connessione di rete veloce (almeno 1Gb) tra il gruppo di server di produzione Interact ed il relativo touchpoint associato.
- La fase di progettazione richiede accesso http o https al runtime per il completamento delle attività di distribuzione. È necessario configurare eventuali firewall o altre applicazioni di rete in modo da consentire la distribuzione. È possibile che si debbano estendere le durate del timeout HTTP tra l'ambiente di progettazione e gli ambienti di runtime in caso di distribuzioni di grandi dimensioni.
- Il modulo della cronologia dei contatti e delle risposte richiede accesso al database della fase di progettazione (tabelle di sistema Campaign) e a quello di runtime (tabelle di runtime Interact). È necessario configurare in modo appropriato i database e la rete in uso, perché si possa verificare questo trasferimento di dati.

In un'installazione di test o staging, è possibile installare la fase di progettazione e il runtime Interact sulla stessa macchina. Questo scenario, però, non è consigliabile per un ambiente di produzione.

Sicurezza delle porte server Interact e della rete

Configurare Interact per proteggere le porte del server.

Runtime di Interact

Alcune di queste porte possono essere chiuse o non sono richieste da tutte le installazioni Interact a seconda della configurazione.

Porta server delle applicazioni Interact per HTTP
La porta predefinita su cui vengono gestite le richieste Interact.

Porta server delle applicazioni Interact per HTTPS
La porta SSL predefinita su cui vengono gestite le richieste Interact.

Porta Interact systemTablesDataSource
Consultare l'argomento sulla configurazione JDBC dell'origine dati in Marketing Platform.

PortaInteract learningTablesDataSource
Consultare l'argomento sulla configurazione JDBC dell'origine dati in Marketing Platform.

Porta Interact contactAndResponseHistoryDataSource
Consultare l'argomento sulla configurazione JDBC dell'origine dati in Marketing Platform.

Porta Interact prodUserDataSource
Consultare l'argomento sulla configurazione JDBC dell'origine dati in Marketing Platform.

Porta Interact testRunDataSource
Consultare l'argomento sulla configurazione JDBC dell'origine dati in Marketing Platform.

Porta di comunicazione ETL
Configurare questa porta in **Interact | ETL | patternStateETL | communicationPort** nelle proprietà di configurazione.

Porta multicast EHCache
Configurare questa porta in **Interact | cacheManagement | Cache | Managers | EHCache | Parameter Data | multicastPort** nelle proprietà di configurazione quando la modalità cache è distribuita.

Porta catalogo ExtremeScale
Configurare questa porta in **Interact | Cache Managers | Extreme Scale | Parameter Data | catalogURLs** nelle proprietà di configurazione.

Porta di monitoraggio JMX Interact
Configurare questa porta in **Interact | monitoring | port** nelle proprietà di configurazione o eseguire `-Dinteract.jmx.monitoring.port=portNumber`.

Porta Interact WebConnector

Questa porta generalmente è uguale alla porta del server Interact, ma è modificabile in `jsconnector.xml`.

Per le porte per qualsiasi prodotto integrato con Interact, consultare la documentazione relativa a tali prodotti.

Il monitoraggio JMX non è richiesto per la funzionalità Interact standard. Tuttavia, viene utilizzato per diagnostica e monitoraggio.

L'accesso alla porta JMX può essere disabilitato nella configurazione di Interact o limitato a un indirizzo IP specifico attraverso le configurazioni del firewall. Questa

è un'operazione consigliata a causa della vulnerabilità di JMX recentemente verificata nell'Apache Commons Library di terze parti.

La funzionalità in remoto di JMX in Apache Geronimo 3.x prima della 3.0.1, come utilizzata in IBM WebSphere Application Server (WAS) Community Edition 3.0.0.3 e altri prodotti, non implementa in modo appropriato il programma di caricamento classi RMI, ciò permette ai violatori in remoto di eseguire codice arbitrario tramite il connettore JMX per inviare un oggetto serializzato costruito. Consultare la pagina http://www-01.ibm.com/support/docview.wss?uid=swg21643282.

Porte di progettazione Interact

Alcune di queste porte possono essere chiuse o non sono richieste da tutte le installazioni Interact a seconda della configurazione.

Porta server delle applicazioni Campaign per HTTP
La porta predefinita su cui vengono gestite le richieste Interact.

Porta server delle applicazioni Campaign per HTTPS
La porta SSL predefinita su cui vengono gestite le richieste Interact.

Porta listener Campaign
La porta utilizzata internamente da Campaign per accettare le connessioni dal client Web.

Altre porte di progettazione Campaign
Consultare la documentazione di Campaign per ulteriori informazioni su queste porte.

Porta connettore JMX Campaign
Configurare questa porta in **Campaign | monitoring | port** nelle proprietà di configurazione solo per il monitoraggio della cronologia delle risposte dei contatti.

Porta del server di monitoraggio operativo Campaign
Configurare questa porta in **Campaign | monitoring | serverURL | communicationPort** nelle proprietà di configurazione.

Accesso a IBM Marketing Software

Utilizzare questa procedura per accedere a IBM Marketing Software.

Prima di iniziare

È richiesto quanto riportato di seguito.

- Una connessione intranet (rete) per accedere al server IBM Marketing Software.
- Un browser supportato installato sul computer.
- Nome utente e password per accedere a IBM Marketing Software.
- L'URL per accedere a IBM Marketing Software sulla rete.

L'URL è il seguente:

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

dove

host è il computer su cui è installato Marketing Platform.

domain.com è il dominio in cui risiede il computer host.

port è il numero di porta su cui è in ascolto il server delle applicazioni Marketing Platform.

Nota: la procedura riportata di seguito presuppone che il collegamento venga eseguito con un account con accesso admin a Marketing Platform.

Procedura

Accedere all'URL di IBM Marketing Software mediante il browser.

- Se IBM Marketing Software è configurato per l'integrazione con Windows Active Directory o con una piattaforma di controllo accesso Web e si è collegati a tale sistema, viene visualizzata la pagina del dashboard predefinita. L'accesso è completato.
- Se viene visualizzato il pannello di accesso, eseguire l'accesso utilizzando le credenziali dell'amministratore standard. In un ambiente con una sola partizione, utilizzare `asm_admin` specificando `password` come password. In un ambiente con più partizioni, utilizzare `platform_admin` specificando `password` come password.

  Viene chiesto di modificare la password. È possibile immettere la password esistente, ma per motivi di sicurezza è necessario scegliere una nuova password.
- Se IBM Marketing Software è configurato per utilizzare SSL, potrebbe essere richiesto di accettare un certificato di sicurezza digitale al primo accesso. Fare clic su **Sì** per accettare il certificato.

Se l'accesso ha esito positivo, IBM Marketing Software visualizza la pagina del dashboard predefinita.

Risultati

Con le autorizzazioni predefinite assegnate agli account amministratore di Marketing Platform, è possibile gestire gli account utente e la sicurezza utilizzando le opzioni elencate nel menu **Impostazioni**. Per eseguire le attività di gestione di più alto livello per i dashboard IBM Marketing Software, è necessario eseguire l'accesso come **platform_admin**.

Capitolo 2. Configurazione degli utenti

Interact richiede la configurazione di due insiemi di utenti, gli utenti dell'ambiente di runtime e gli utenti dell'ambiente di progettazione.

- Gli **utenti di runtime** sono creati nell'istanza di Marketing Platform e configurati per utilizzare i server di runtime.
- Gli **utenti della fase di progettazione** sono utenti Campaign. Configurare la sicurezza per i vari membri del team di progettazione come per Campaign.

Configurazione dell'utente dell'ambiente di runtime

Dopo aver installato Interact, è necessario configurare almeno un utente Interact, l'utente dell'ambiente di runtime. Gli utenti di runtime sono creati in Marketing Platform.

Informazioni su questa attività

L'utente dell'ambiente di runtime fornisce l'accesso alle tabelle di runtime. Per utente dell'ambiente di runtime si intende il nome utente e la password utilizzati per distribuire canali interattivi. Il server di runtime utilizza l'autenticazione JDBC del server delle applicazioni Web per le credenziali del database. Non è necessario aggiungere origini dati dell'ambiente di runtime all'utente dell'ambiente di runtime.

Un utente LDAP e qualsiasi utente Platform possono distribuire un canale interattivo. Non è necessario InteractAdminRole per distribuire il canale interattivo.

Quando si creano utenti di runtime:

- Se si hanno istanze separate di Marketing Platform per ogni server di runtime, è necessario creare lo stesso utente e la stessa password in ogni istanza. Tutti i server di runtime, che appartengono allo stesso gruppo di server, devono condividere le credenziali utente.
- Se si utilizza un programma di utilità per il caricamento del database, è necessario definire le tabelle di runtime come origine dati, con le credenziali di accesso per l'ambiente di runtime nelle proprietà di configurazione in `Interact > general > systemTablesDataSource`.
- Se si abilita la sicurezza per il monitoraggio JMX con il protocollo JMXMP, potrebbe essere necessario configurare un utente separato per la sicurezza del monitoraggio JMX.

Consultare la documentazione di Marketing Platform per conoscere la procedura di creazione degli utenti di runtime.

Configurazione degli utenti dell'ambiente di progettazione

Gli utenti dell'ambiente di progettazione sono utenti Campaign. Si configurano gli utenti dell'ambiente di progettazione in modo analogo a come si configurano le autorizzazioni di ruolo di Campaign.

Informazioni su questa attività

Alcuni utenti dell'ambiente di progettazione richiedono anche qualche autorizzazione Campaign, ad esempio Macro personalizzate.

Quando si creano utenti dell'ambiente di progettazione:

- Se sono presenti utenti Campaign con l'autorizzazione a modificare diagrammi di flusso interattivi, fornire ad essi l'accesso all'origine dati delle tabelle di esecuzione di test.
- Se è stato installato e configurato Interact, sono disponibili le seguenti opzioni supplementari per la politica globale predefinita e per nuove politiche.
-

Categoria	Autorizzazioni
Campagne	• Visualizza strategie di interazione della campagna - Possibilità di visualizzare, ma non modificare, le schede della strategia di interazione in una campagna. • Modifica strategie di interazione della campagna - Possibilità di apportare modifiche alle schede della strategia di interazione, incluse le regole di trattamento. • Cancella strategie di interazione della campagna - Possibilità di rimuovere le schede della strategia di interazione dalle campagne. La cancellazione di una scheda della strategia di interazione è limitata se la strategia di interazione è stata inclusa in una distribuzione del canale interattivo. • Aggiungi strategie di interazione della campagna - Possibilità di creare nuove schede della della strategia di interazione in una campagna. • Inizia distribuzioni della strategia di interazione della campagna - Possibilità di contrassegnare una scheda della strategia di interazione per la distribuzione o per la rimozione della distribuzione.
Canali interattivi	• Distribuisci canali interattivi - Possibilità di distribuire un canale interattivo per gli ambienti di runtime Interact. • Modifica canali interattivi - Possibilità di apportare modifiche alla scheda riepilogo dei canali interattivi. • Cancella canali interattivi - Possibilità di rimuovere i canali interattivi. La cancellazione dei canali interattivi è limitata se il canale interattivo è stato distribuito. • Visualizza canali interattivi - Possibilità di visualizzare, ma non modificare, i canali interattivi. • Aggiungi canali interattivi - Possibilità di creare nuovi canali interattivi. • Visualizza report del canale interattivo - Possibilità di visualizzare la scheda di analisi del canale interattivo. • Aggiungi oggetti figlio del canale interattivo - Possibilità di aggiungere punti di interazione, zone, eventi e categorie.

Categoria	Autorizzazioni
Sessioni	• Visualizza diagrammi di flusso interattivi - Possibilità di visualizzare un diagramma di flusso interattivo in una sessione. • Aggiungi diagrammi di flusso interattivi - Possibilità di creare nuovi diagrammi di flusso interattivi in una sessione. • Modifica diagrammi di flusso interattivi - Possibilità di apportare modifiche ai diagrammi di flusso interattivi. • Cancella diagrammi di flusso interattivi - Possibilità di rimuovere i diagrammi di flusso interattivi. La cancellazione dei diagrammi di flusso interattivi è limitata, se il canale interattivo, a cui il flusso interattivo è assegnato, è stato distribuito. • Copia diagrammi di flusso interattivi - Possibilità di copiare diagrammi di flusso interattivi. • Esegui test di diagrammi di flusso interattivi - Possibilità di iniziare un'esecuzione di test di un diagramma di flusso interattivo. • Revisiona diagrammi di flusso interattivi - Possibilità di visualizzare un diagramma di flusso interattivo e aprire processi per visualizzare le impostazioni, ma non è possibile apportare modifiche. • Distribuisci diagrammi di flusso interattivi - Possibilità di contrassegnare diagrammi di flusso interattivi per la distribuzione o per la rimozione della distribuzione.

Autorizzazioni dell'ambiente di progettazione di esempio

In questo esempio vengono elencate le autorizzazioni concesse a due ruoli differenti, uno per gli utenti che creano diagrammi di flusso interattivi e uno per quelli che definiscono strategie di interazione.

Ruolo per il diagramma di flusso interattivo

Questa tabella riporta le autorizzazioni concesse al ruolo per il diagramma di flusso interattivo:

Categoria	Autorizzazione
Macro personalizzata	Al ruolo utente sono concesse queste autorizzazioni: • Aggiungi macro personalizzate • Modifica macro personalizzate • Utilizza macro personalizzate
Campo derivato	Al ruolo utente sono concesse queste autorizzazioni: • Aggiungi campi derivati • Modifica campi derivati • Utilizza campi derivati
Modello di diagramma di flusso	Al ruolo utente sono concesse queste autorizzazioni: • Incolla modelli

Categoria	Autorizzazione
Modello del segmento	Al ruolo utente sono concesse queste autorizzazioni: • Aggiungi segmenti • Modifica segmenti
Sessione	Al ruolo utente sono concesse queste autorizzazioni: • Visualizza riepilogo sessioni • Visualizza diagrammi di flusso interattivi • Aggiungi diagrammi di flusso interattivi • Modifica diagrammi di flusso interattivi • Copia diagrammi di flusso interattivi • Esegui test di diagrammi di flusso interattivi • Distribuisci diagrammi di flusso interattivi

Ruolo per la strategia di interazione

Questa tabella riporta le autorizzazioni concesse al ruolo per la strategia di interazione:

Categoria	Autorizzazione
Campagna	Al ruolo utente sono concesse queste autorizzazioni: • Visualizza riepilogo della campagna • Gestisci celle obiettivo della campagna • Visualizza strategie di interazione della campagna • Modifica strategie di interazione della campagna • Aggiungi strategie di interazione della campagna • Inizia distribuzioni strategia di interazione della campagna
Offerta	Al ruolo utente sono concesse queste autorizzazioni: • Visualizza riepilogo delle offerte
Modello del segmento	Al ruolo utente sono concesse queste autorizzazioni: • Visualizza riepilogo dei segmenti
Sessione	Al ruolo utente sono concesse queste autorizzazioni: • Revisiona diagrammi di flusso interattivi

Capitolo 3. Gestione delle origini dati di Interact

Interact richiede che diverse origini dati funzionino correttamente. Alcune origini dati contengono le informazioni che Interact richiede per poter funzionare, altre origini dati contengono i dati dell'utente.

Le seguenti sezioni descrivono le origini dati di Interact incluse le informazioni necessarie per configurarle correttamente, ed alcuni suggerimenti per gestirle.

Origini dati Interact

Interact richiede per il funzionamento varie serie di dati. Le serie di dati sono archiviate e recuperate in origini dati e le origini dati che si impostano, dipenderanno dalle funzioni di Interact che si stanno abilitando.

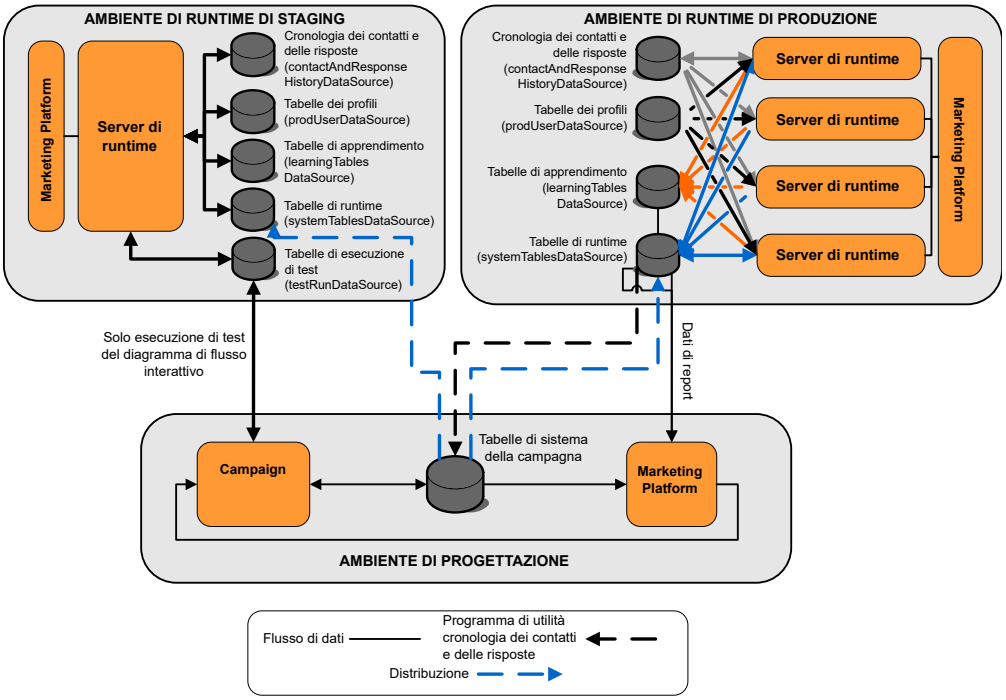
- **Tabelle di sistema di Campaign**. Oltre a tutti dati per Campaign, le tabelle di sistema di Campaign contengono dati per i componenti Interact che si creano nell'ambiente di progettazione, ad esempio le regole di trattamento e i canali interattivi. L'ambiente di progettazione e le tabelle di sistema di Campaign utilizzano lo stesso schema e database fisico.
- **Tabelle di runtime**(`systemTablesDataSource`). Questa origine dati contiene i dati sulla distribuzione dall'ambiente di progettazione, tabelle di staging per la cronologia dei contatti e delle risposte e statistiche sul runtime.
- **Tabelle profili** (`prodUserDataSource`). Questa origine dati contiene tutti i dati del cliente, oltre le informazioni raccolte in tempo reale, richieste dai diagrammi di flusso interattivi per inserire correttamente i visitatori in segmenti smart. Se si fa riferimento interamente a dati in tempo reale, non sono necessarie le tabelle profili. Se si stanno utilizzando tabelle profili, è necessario disporre almeno di una tabella profili per livello destinatario di cui fa uso il canale interattivo.

  Le tabelle profili possono contenere anche le tabelle utilizzate per ampliare la presentazione dell'offerta, incluse tabelle per la soppressione dell'offerta, la sovrascrittura del punteggio e l'assegnazione di offerte globali e individuali.
- **Tabelle di esecuzione di test** (`testRunDataSource`). Questa origine dati contiene un esempio di tutti i dati richiesti dai diagrammi di flusso interattivi per inserire i visitatori in segmenti smart, inclusi dati simulano quelli che vengono raccolti in tempo reale, durante un'interazione. Queste tabelle sono necessarie solo per il gruppo di server designato come gruppo di server di esecuzione di test per l'ambiente di progettazione.
- **Tabelle di apprendimento** (`learningTablesDataSource`). Questa origine dati contiene tutti i dati che vengono raccolti dal programma di utilità di apprendimento integrato. Queste tabelle possono includere una tabella che definisce attributi dinamici. Se non si sta utilizzando l'apprendimento o si sta utilizzando un programma di utilità di apprendimento esterno creato dall'utente, non sono necessarie tabelle di apprendimento.
- **Cronologia dei contatti e delle risposte per la risposta delle sessioni incrociate** (`contactAndResponseHistoryDataSource`). Questa origine dati contiene le tabella della cronologia dei contatti di Campaign oppure una copia di tali tabelle. Se non si sta utilizzando la funzione di risposta delle sessioni incrociate, non sarà necessario configurare queste tabelle della cronologia dei contatti.

Database e applicazioni

Le origini dati che si creano per essere utilizzate da Interact potrebbero anche essere utilizzate per scambiare o condividere i dati con altre applicazioni di IBM Marketing Software.

Il diagramma seguente mostra le origini dati di Interact e in che modo si correlano alle applicazioni IBM Marketing Software.



- Sia Campaign che il gruppo di server per l'esecuzione di test accedono alle tabelle di esecuzione di test.
- Le tabelle di esecuzione di test vengono utilizzate esclusivamente per le esecuzioni di test di diagrammi di flusso interattivi.
- Quando si sta utilizzando un server di runtime per il test di una distribuzione, che include l'API Interact, il server di runtime fa riferimento alle tabelle profili per i dati.
- Se si configura il modulo della cronologia dei contatti e delle risposte, il modulo utilizza un processo ETL (Extract, Transform, Load) in background per spostare i dati dalle tabelle di staging di runtime alle tabelle della cronologia dei contatti e delle risposte di Campaign.
- La funzione di reporting esegue query dei dati dalle tabelle di apprendimento, dalle tabelle di runtime e dalle tabelle di sistema di Campaign per presentare report in Campaign.

Si dovrebbero configurare gli ambienti di runtime di test per utilizzare una serie di tabelle differente rispetto agli ambienti di runtime di produzione. Mantenendo separate le tabelle di staging e produzione, è possibile tenere i risultati di test separati dai risultati reali. Si tenga presente che il modulo della cronologia dei contatti e delle risposte inserisce sempre dati nelle tabelle effettive della cronologia dei contatti e delle risposte di Campaign (Campaign non dispone di tabelle della cronologia dei contatti e delle risposte di test). Se esistono tabelle di apprendimento separate per l'ambiente di runtime di test e si desidera visualizzare

i risultati in report, è necessaria un'istanza separata di IBM Cognos BI, che esegua i report di apprendimento per l'ambiente di test.

Tabelle di sistema Campaign

Quando si installa l'ambiente di progettazione di Interact, si creano anche nuove tabelle, specifiche per Interact, nelle tabelle di sistema di Campaign. Le tabelle che si creano dipendono dalle funzioni che si stanno abilitando per Interact.

Se si abilita il modulo della cronologia dei contatti e delle risposte, il modulo copia la cronologia dei contatti e delle risposte dalle tabelle di staging, che si trovano nelle tabelle di runtime, alle tabelle della cronologia dei contatti e delle risposte, che si trovano nelle tabelle di sistema di Campaign. Le tabelle predefinite sono `UA_ContactHistory`, `UA_DtlContactHist` e `UA_ResponseHistory`, ma il modulo della cronologia dei contatti e delle risposte utilizzerà qualsiasi tabella associata in Campaign per le tabelle della cronologia dei contatti e delle risposte.

Se si utilizzano le tabelle delle offerte globali e quelle di sovrascrittura del punteggio per l'assegnazione di offerte, è possibile che sia necessario popolare la tabella `UACI_ICBatchOffers`, nelle tabelle di sistema di Campaign, se si stanno gestendo offerte non contenute nelle regole di trattamento per il canale interattivo.

Tabelle di runtime

Se si dispone di più di un livello destinatario, è necessario creare le tabelle di staging per i dati della cronologia dei contatti e delle risposte per ciascun livello destinatario.

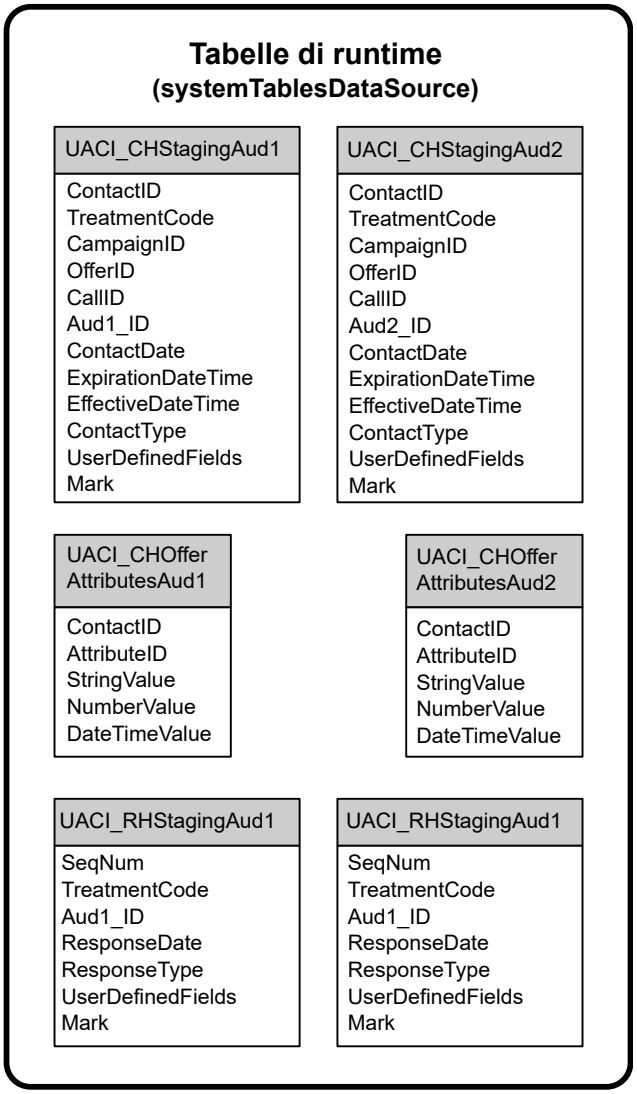
Gli script SQL creano le seguenti tabelle per il livello destinatario predefinito:

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

È necessario creare copie di queste tabelle per ognuno dei livelli destinatario nelle tabelle di runtime.

Se le tabelle della cronologia dei contatti e delle risposte di Campaign contengono campi definiti dall'utente, è necessario creare gli stessi nomi e tipi di campo nelle tabelle `UACI_CHStaging` e `UACI_RHStaging`. È possibile popolare tali campi durante il runtime creando coppie nome-valore dello stesso nome nei dati di sessione. Ad esempio, le tabelle della cronologia dei contatti e delle risposte contengono il campo `catalogID`. È necessario aggiungere il campo `catalogID` a entrambe le tabelle `UACI_CHStaging` e `UACI_RHStaging`. In seguito, l'API Interact popola questo campo definendo un parametro evento come coppia nome-valore denominata `catalogID`. I dati di sessione possono essere forniti dalla tabella profili, dai dati temporali, dall'apprendimento o dall'API Interact.

Il seguente diagramma mostra tabelle di esempio per i destinatari Aud1 e Aud2. Questo diagramma non include tutte le tabelle nel database di runtime.



Tutti i campi nelle tabelle sono obbligatori. È possibile modificare il `CustomerID` e i `UserDefinedFields` in modo che corrispondano alle tabelle della cronologia dei contatti e delle risposte di Campaign.

Tabelle di esecuzione di test

Le tabelle di esecuzione di test vengono utilizzate solo per esecuzioni di test di diagrammi di flusso interattivi. Le esecuzioni di test dei diagrammi di flusso interattivi devono testare la logica di segmentazione. È necessario configurare un solo database di esecuzioni di test per l'installazione di Interact. Non è necessario che le tabelle di esecuzione di test siano in un database autonomo. Ad esempio, si potrebbero utilizzare le tabelle di dati del cliente per Campaign.

L'utente del database associato alle tabelle di esecuzione di test deve disporre del privilegio CREATE per aggiungere le tabelle di risultato dell'esecuzione di test.

Il database di esecuzioni di test deve contenere tutte le tabelle associate nel canale interattivo.

Queste tabelle devono contenere i dati per eseguire gli scenari che si desidera testare nei diagrammi di flusso interattivi. Ad esempio, se i diagrammi di flusso interattivi dispongono della logica per ordinare le persone in segmenti in base alla scelta selezionata in un sistema di caselle vocali, è necessario avere almeno una riga per ogni possibile selezione. Se si sta creando un'interazione che funziona con un modulo sul sito web, includere le righe che rappresentano i dati mancanti o in formato non corretto, ad esempio, utilizzare `name@domaincom` per il valore di un indirizzo email.

Ogni tabella di esecuzione di test deve contenere almeno un elenco di ID per il livello destinatario appropriato, e una colonna che rappresenta i dati in tempo reale che si prevede di utilizzare. Poiché le esecuzioni di test non hanno accesso ai dati in tempo reale, è necessario fornire dati campione per ogni porzione di dati in tempo reale previsti. Ad esempio, se si desidera utilizzare i dati che è possibile raccogliere in tempo reale, come il nome dell'ultima pagina web visitata, memorizzata nell'attributo `lastPageVisited`, o il numero di elementi presenti in un carrello degli acquisti, memorizzato nell'attributo `shoppingCartItemCount`, è necessario creare le colonne con gli stessi nomi e popolarle con gli stessi dati. Questo consente di eseguire i test dei rami della logica del diagramma di flusso che sono di natura comportamentale o contestuale.

Le esecuzioni di test dei diagrammi di flusso interattivi non sono ottimizzate per il funzionamento con dataset di grosse dimensioni. È possibile limitare il numero di righe utilizzare per l'esecuzione di test nel processo di interazione. Tuttavia, questo comporta sempre la selezione della prima serie di righe. Per assicurarsi che siano selezionate serie di righe diverse, utilizzare viste diverse delle tabelle di esecuzione di test.

Per testare le prestazioni della produttività dei diagrammi di flusso interattivi nel runtime, è necessario creare un ambiente di runtime di test, incluso una tabella profili per l'ambiente di test.

In pratica, potrebbero essere necessarie tre serie di tabelle per il test, una tabella di esecuzione di test per le esecuzioni di test dei diagrammi di flusso interattivi, tabelle profili di test per il gruppo di server di test, e una serie di tabelle profili di produzione.

Sovrascrittura dei tipi di dati predefiniti utilizzati per le tabelle create dinamicamente

L'ambiente di runtime Interact crea dinamicamente le tabelle in due scenari: durante un'esecuzione di test di un diagramma di flusso e durante l'esecuzione di un processo Snapshot che scrive in una tabella che non esiste già. Per creare queste tabelle, Interact si basa su tipi di dati hardcoded per ogni tipo di database supportato.

È possibile sovrascrivere i tipi di dati predefiniti creando una tabella di tipi di dati alternativi, denominata `uaci_column_types`, in `testRunDataSource` o `prodUserDataSource.` Questa tabella aggiuntiva consente ad Interact di soddisfare i rari casi che non sono coperti dai tipi di dati hardcoded.

Quando la tabella `uaci_column_types` è definita, Interact utilizza i metadati per le colonne come tipi di dati da utilizzare per qualsiasi generazione di tabella. Se la tabella `uaci_column_types` non è definita o se vengono rilevate eccezioni nel tentativo di leggere la tabella, vengono utilizzati i tipi di dati predefiniti.

All'avvio, il sistema di runtime inizialmente cerca in testRunDataSource la tabella uaci_column_types. Se la tabella uaci_column_types non esiste in testRunDataSource, o se prodUserDataSource è di un tipo di database diverso, Interact cerca in prodUserDataSource la la tabella.

Sovrascrittura dei tipi di dati predefiniti

Utilizzare questa procedura per sovrascrivere i tipi di dati predefiniti per le tabelle create dinamicamente.

Informazioni su questa attività

È necessario riavviare il server di runtime ogni volta che si modifica la tabella uaci_column_types. Pianificare le modifiche in modo che il riavvio del server abbia un impatto minimo sulle operazioni.

Procedura

1. Creare una tabella in TestRunDataSource o ProdUserDataSource con le seguenti proprietà:

   Nome tabella: uaci_column_types

   Nomi colonna:

   - uaci_float
   - uaci_number
   - uaci_datetime
   - uaci_string

   Utilizzare il tipo di dati appropriato supportato dal database per definire ciascuna colonna.
2. Riavviare il server di runtime per consentire a Interact il riconoscimento della nuova tabella.

Tipi di dati predefiniti per tabelle create dinamicamente

Per ogni database supportato, utilizzato dal sistema di runtime Interact, sono previsti tipi di dati hardcoded, adottati per impostazione predefinita per colonne float, number, date/time e string.

Tabella 1. Tipi di dati predefiniti per tabelle create dinamicamente

Database	Tipi di dati predefiniti
DB2	• float • bigint • timestamp • varchar
Informix	• float • int8 • DATETIME YEAR TO FRACTION • char2
Oracle	• float • number(19) • timestamp • varchar2

Tabella 1. Tipi di dati predefiniti per tabelle create dinamicamente (Continua)

Database	Tipi di dati predefiniti
SQL Server	• `float` • `bigint` • `datetime` • `nvarchar`

Database dei profili

Il contenuto del database dei profili dipende interamente dai dati necessari per la configurazione dei diagrammi di flusso interattivi e l'API Interact. Interact richiede o consiglia che ogni database contenga determinate tabelle e dati.

Il database dei profili deve contenere quanto segue:

- Tutte le tabelle associate nel canale interattivo.

  Tali tabelle devono contenere tutti i dati richiesti per l'esecuzione dei diagrammi di flusso interattivi in produzione. Queste tabelle devono essere rese bidimensionali, semplificate e indicizzate correttamente. Poiché vi è un costo di prestazione per l'accesso ai dati dimensionali, utilizzare uno schema denormalizzato, quando possibile. Come minimo, indicizzare la tabella profili sui campi ID livello destinatario. Se vengono richiamati altri campi dalle tabelle dimensionali, questi devono essere indicizzati in modo appropriato per ridurre il tempo di richiamo dal database. Gli ID destinatario delle tabelle profili devono corrispondere agli ID destinatario definiti in Campaign.
- Se si imposta la proprietà di configurazione `enableScoreOverrideLookup` su true, è necessario includere una tabella di sovrascrittura dei punteggi per almeno un livello destinatario. Per definire i nomi di tabella di sovrascrittura dei punteggi utilizzare la proprietà `scoreOverrideTable`.

  La tabella di sovrascrittura dei punteggi può contenere associazioni cliente-offerta individuali. È possibile creare una tabella di sovrascrittura dei punteggi di esempio, `UACI_ScoreOverride` eseguendo lo script SQL `aci_usrtab` sul database dei profili. È necessario inoltre indicizzare questa tabella nella colonna ID destinatario.

  Se si imposta la proprietà `enableScoreOverrideLookup` su false, non è necessario includere una tabella di sovrascrittura dei punteggi.
- Se si imposta la proprietà di configurazione `enableDefaultOfferLookup` su true, è necessario includere la tabella di offerte globali (`UACI_DefaultOffers`). È possibile creare la tabella di offerte globali eseguendo lo script SQL `aci_usrtab` sul database dei profili.

  La tabella di offerte globali può contenere le associazioni destinatario-offerta.
- Se si imposta la proprietà `enableOfferSuppressionLookup` su true, è necessario includere una tabella delle soppressioni di offerta per almeno un livello destinatario. Definire i nomi della tabella delle soppressioni di offerta con la proprietà `offerSuppressionTable`.

  La tabella delle soppressioni di offerta può contenere una riga per ogni offerta soppressa per un membro destinatario, anche se non è richiesta una voce per tutti i membri destinatari. È possibile creare una tabella delle soppressioni dell'offerta di esempio, `UACI_BlackList` eseguendo lo script SQL `aci_usrtab` per il database dei profili.

  Se si imposta la proprietà `enableOfferSuppressionLookup` su false, non è necessario includere una tabella delle soppressioni di offerta.

Una quantità elevata di dati in una di queste tabelle può ostacolare le prestazioni. Per risultati ottimali, inserire gli indici appropriati nelle colonne del livello destinatario per le tabelle utilizzate al runtime che hanno una quantità elevata di dati.

Tutte le proprietà di configurazione a cui si è fatto rifermento in precedenza sono presenti nella categoria **Interact > profilo** o **Interact > profilo > Livelli destinatario > *AudienceLevel***. Lo script SQL `aci_usrtab` si trova nella directory `ddl` della directory di installazione dell'ambiente di runtime.

I diagrammi riportati di seguito mostrano esempi di tabelle per l'esecuzione di test e i database dei profili per i livelli destinatario Aud1 e Aud2.

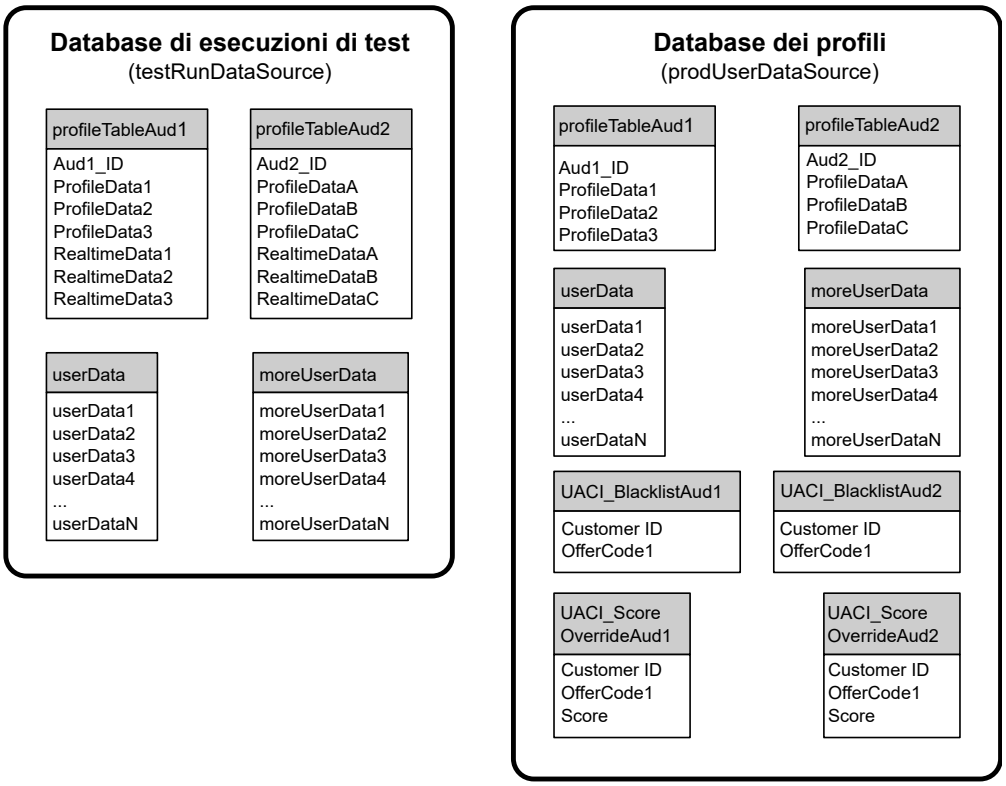


Tabelle di apprendimento

Se si utilizza l'apprendimento integrato di Interact, è necessario configurare le tabelle di apprendimento. Tali tabelle contengono tutti i dati su cui opera la funzione di apprendimento integrata.

Se si utilizzano gli attributi di apprendimento dinamici, è necessario popolare la tabella `UACI_AttributeList`.

L'apprendimento implica la scrittura in tabelle di staging intermedie e l'aggregazione delle informazioni delle tabelle di staging nelle tabelle di apprendimento. Le proprietà di configurazione `insertRawStatsIntervalInMinutes` e `aggregateStatsIntervalInMinutes` nella categoria `Interact > offerserving > Built-in Learning Config` determinano la frequenza con cui vengono popolate le tabelle di apprendimento.

L'attributo `insertRawStatsIntervalInMinutes` determina la frequenza con cui le informazioni di contatto e di accettazione di ciascun cliente e offerta vengono spostate dalla memoria alle tabelle di staging, `UACI_OfferStatsTX` e `UACI_OfferTxAll`. Le informazioni memorizzare nelle tabelle di staging vengono aggregate e spostate nelle tabelle `UACI_OfferStats` e `UACI_OfferStatsAll` ad intervalli regolari determinati dalla proprietà di configurazione `aggregateStatsIntervalInMinutes`.

L'apprendimento integrato di Interact utilizza questi dati per calcolare i punteggi finali per le offerte.

Cronologia dei contatti per il tracciamento della risposta delle sessioni incrociate

Se si abilita la funzione di risposta delle sessioni incrociate, l'ambiente di runtime necessita dell'accesso in sola lettura alle tabelle della cronologia dei contatti di Campaign. È possibile configurare l'ambiente di runtime per visualizzare le tabelle di sistema di Campaign, oppure creare una copia delle tabelle della cronologia dei contatti di Campaign. Se si crea una copia delle tabella, è necessario gestire il processo che tiene aggiornata la copia. Il modulo della cronologia dei contatti e delle risposte non aggiornerà la copia delle tabelle della cronologia dei contatti.

È necessario eseguire lo script SQL `aci_crhtab` sulle tabelle della cronologia dei contatti per aggiungere le tabelle richieste per la funzione di tracciamento della risposta delle sessioni incrociate.

Esecuzione di script del database per abilitare le funzioni

Per utilizzare le funzioni facoltative disponibili in , eseguire gli script di database per il database per creare tabelle o aggiornare quelle esistenti.

L'installazione di , sia l'ambiente della fase di progettazione che l'ambiente di runtime, include script **`ddl`** di funzioni. Gli script **`ddl`** aggiungono le colonne richieste alle tabelle.

Per abilitare le funzioni facoltative, eseguire lo script appropriato per il database o la tabella indicati.

`dbType` è il tipo di database, ad esempio `sqlsvr` per Microsoft SQL Server, `ora` per Oracle o `db2` per IBM DB2.

Utilizzare la seguente tabella per eseguire gli script di database per il database per creare tabelle o aggiornare quelle esistenti:

Tabella 2. Script di database

Nome funzione	Script funzione	Esegui su	Modifica
Offerte globali, soppressione offerte e sovrascrittura punteggi	**`aci_usrtab`**_*dbType*`.sql` in *`Interact_Home`*`\ddl\acifeatures\` (directory di installazione dell'ambiente di runtime)	Database dei profili (userProdDataSource)	Crea le tabelle UACI_DefaultOffers, UACI_BlackList e UACI_ScoreOverride.

Tabella 2. Script di database (Continua)

Nome funzione	Script funzione	Esegui su	Modifica
Punteggio	**aci_scoringfeature**_*dbType*.**sql** in *Interact_Home*\ddl\acifeatures\ (directory di installazione dell'ambiente di runtime)	Tabelle di sovrascrittura dei punteggi nel database dei profili (userProdDataSource)	Aggiunge le colonne LikelihoodScore e AdjExploreScore.
Apprendimento	**aci_lrnfeature**_*dbType*.**sql** in *Interact_Home*\interactDT\ddl\acifeatures\ (directory di installazione dell'ambiente della fase di progettazione)	Database Campaign che contiene le tabelle della cronologia dei contatti	Aggiunge le colonne RTSelectionMethod, RTLearningMode e RTLearningModelID alla tabella UA_DtlContactHist. Inoltre aggiunge le colonne RTLearningMode e RTLearningModelID alla tabella UA_ResponseHistory. Questo script è richiesto anche dalle funzioni di reporting fornite dal Reports Pack facoltativo.

Informazioni sul tracciamento della cronologia dei contatti e delle risposte

È possibile configurare l'ambiente di runtime in modo da registrare la cronologia dei contatti e delle risposte nelle tabelle della cronologia dei contatti e delle risposte di Campaign. I server di runtime memorizzano la cronologia dei contatti e delle risposte in tabelle di staging. Il modulo della cronologia dei contatti e delle risposte copia questi dati dalle tabelle di staging nelle tabelle della cronologia dei contatti e delle risposte di Campaign.

Il modulo della cronologia dei contatti e delle risposte funziona solo se si impostano le proprietà `Campaign > partitions > partition1 > Interact > interactInstalled` e `contactAndResponseHistTracking > isEnabled` nella pagina Configurazione per l'ambiente di progettazione su `yes`.

Se si sta utilizzando il modulo di tracciamento della risposta delle sessioni incrociate, il modulo della cronologia dei contatti e delle risposte dovrà essere considerato un'entità separata.

Tipi di contatto e risposta

È possibile registrare un tipo di contatto e due tipi di risposta con Interact. È possibile inoltre registrare ulteriori tipi di risposta personalizzati con il metodo `postEvent`.

Proprietà nella tabella di contactAndResponseHistTracking

In questa tabella sono elencate le proprietà trovate nella categoria `contactAndResponseHistTracking`:

Evento	Tipo di contatto/risposta	Proprietà di configurazione
Registra contatto offerta	Contatto	`contattato`
Registra accettazione offerta	Risposta	`accetta`

Evento	Tipo di contatto/risposta	Proprietà di configurazione
Registra rifiuto offerta	Risposta	rifiuta

Proprietà nella tabella di UA_UsrResponseType

Assicurarsi che la colonna CountsAsResponse della tabella UA_UsrResponseType nelle tabelle di sistema di Campaign sia configurata in modo appropriato. Nella tabella UA_UsrResponseType, devono esistere tutti questi tipi di risposta.

Perché una voce sia valida nella tabella UA_UsrResponseType, è necessario definire un valore per tutte le colonne nella tabella, inclusa CountsAsResponse. Valori validi per CountsAsResponse sono:

- 0 - nessuna risposta
- 1 - una risposta
- 2 - un rifiuto
-

Queste risposte sono utilizzate per la creazione di report.

Tipi di risposta aggiuntivi

In Interact, è possibile utilizzare il metodo postEvent, contenuto nell'API Interact, per attivare un evento che registri un'azione "accetta" o "rifiuta" per un'offerta. È anche possibile ampliare il sistema per consentire alla chiamata postEvent di registrare ulteriori tipi di risposta, ad esempio Esamina, Valuta, Prendi l'impegno o Porta a compimento.

Nella tabella UA_UsrResponseType, che fa parte delle tabelle di sistema di Campaign, devono esistere tutti questi tipi di risposta. Utilizzando parametri evento specifici per il metodo postEvent, è possibile registrare ulteriori tipi di risposta e definire se nell'apprendimento dovrebbe essere inclusa un'accettazione.

Per registrare ulteriori tipi di risposta, è necessario aggiungere i seguenti parametri evento:

- **UACIResponseTypeCode** - una stringa che rappresenta un codice di tipo di risposta. Il valore deve essere una voce valida nella tabella UA_UsrResponseType.

  Perché una voce sia valida in UA_UsrResponseType, è necessario definire tutte le colonne nella tabella, inclusa CountsAsResponse. Valori validi per CountsAsResponse sono 0, 1, o 2. 0 indica nessuna risposta, 1 indica una risposta e 2 indica un rifiuto. Queste risposte sono utilizzate per la creazione di report.
- **UACILogToLearning** - Un numero con valore 1 o 0. 1 indica che Interact dovrebbe registrare l'evento come accettazione per il sistema di apprendimento o abilitare la soppressione dell'offerta in una sessione. 0 indica che Interact non dovrebbe registrare l'evento per il sistema di apprendimento o abilitare la soppressione dell'offerta in una sessione. Questo parametro consente di creare vari metodi postEvent, che registrano differenti tipi di risposta, senza influire sull'apprendimento. Se non si definisce UACILogToLearning, Interact presuppone il valore predefinito 0.

Se viene fornito un responseTypeCode durante l'invio di un evento di accettazione, l'offerta non viene soppressa al momento dell'accettazione. Indipendentemente dal valore di ResponseTypeCode (ad esempio, 0, 1, 2), se logToLearningAsAccept ha valore 0, l'offerta non dovrebbe mai venire soppressa. Per sopprimere l'offerta,

postEvent non dovrebbe specificare il parametro UACIResponseTypeCode. Se viene fornito il parametro UACIResponseTypeCode, il valore di UACILogToLearning dovrebbe essere 1 se si desidera che l'offerta venga soppressa.

È possibile che si vogliano creare diversi eventi con l'azione Registra accettazione offerta, uno per ogni tipo di risposta che si desidera registrare, oppure un singolo evento con l'azione Registra accettazione offerta, da utilizzare per ogni chiamata postEvent, utilizzata per registrare tipi di risposta differenti.

Ad esempio, creare un evento con l'azione Registra accettazione offerta per ogni tipo di risposta. È possibile definire le seguenti risposte personalizzate nella tabella UA_UsrResponseType [as Name (codice)]: Explore (EXP), Consider (CON) e Commit (CMT). È quindi possibile creare tre eventi e denominarli LogAccept_Explore, LogAccept_Consider e LogAccept_Commit. Tutti e tre gli eventi sono esattamente uguali (prevedono l'azione Registra accettazione offerta), ma i nomi sono differenti, in modo che la persona che utilizza l'API li possa distinguere.

Oppure, è possibile creare un singolo evento con l'azione Registra accettazione offerta da utilizzare per tutti i tipi di risposta personalizzati. Ad esempio, denominarlo LogCustomResponse.

Quando si utilizza l'API, non esiste alcuna differenza funzionale tra gli eventi, ma le convenzioni di denominazione possono contribuire a rendere più comprensibile il codice. Inoltre, se si fornisce un nome diverso a ciascuna risposta del cliente, il report di riepilogo attività eventi canale visualizza informazioni più accurate.

In primo luogo, impostare tutte le coppie nome-valore

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

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

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

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

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

In questo primo esempio viene illustrato l'utilizzo dei singoli eventi.

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

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

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

In questo secondo esempio viene mostrato l'utilizzo di un solo evento.

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

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

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

Entrambi gli esempi eseguono esattamente le stesse azioni, tuttavia, una versione è possibile che sia più facile da leggere rispetto all'altra.

Tabelle di staging dell'ambiente di runtime per il mapping delle tabelle di cronologia Campaign

Le tabelle di staging di cronologia dei contatti Interact si associano alle tabelle di cronologia Campaign. È necessario disporre di una delle tabelle di staging dell'ambiente di runtime per ciascun livello destinatario.

Mapping della tabella di staging di cronologia dei contatti UACI_CHStaging

Questa tabella mostra il modo in cui la tabella di staging dell'ambiente di runtime `UACI_CHStaging` si associa alla tabella della cronologia dei contatti Campaign. I nomi tabella visualizzati sono le tabelle di esempio create per il destinatario predefinito nelle tabelle di runtime e nelle tabelle di sistema Campaign.

Tabella 3. Cronologia dei contatti

UACI_CHStaging **Nome della colonna della tabella di staging di cronologia dei contatti Interact**	**Tabella della cronologia dei contatti Campaign**	**Nome colonna tabella**
ContactID	N/D	N/D
TreatmentCode	UA_Treatment	TreatmentCode
CampaignID	UA_Treatment	CampaignID
OfferID	UA_Treatment	OfferID
CellID	UA_Treatment	CellID
CustomerID	UA_DtlContactHist	CustomerID
ContactDate	UA_DtlContactHist	ContactDateTime

Tabella 3. Cronologia dei contatti (Continua)

UACI_CHStaging **Nome della colonna della tabella di staging di cronologia dei contatti Interact**	**Tabella della cronologia dei contatti Campaign**	**Nome colonna tabella**
ExpirationDateTime	UA_Treatment	ExpirationDateTime
EffectiveDateTime	UA_Treatment	EffectiveDateTime
ContactType	UA_DtlContactHist	ContactStatusID
UserDefinedFields	UA_DtlContactHist	UserDefinedFields

`ContactID` è una chiave per unire la tabella `UACI_CHOfferAtrib` con la tabella `UACI_CHStaging`. La colonna `userDefinedFields` può contenere qualsiasi dato si desideri.

Mapping della tabella di staging di cronologia dei contatti UACI_CHOfferAttrib

Questa tabella mostra il modo in cui la tabella di staging dell'ambiente di runtime `UACI_CHOfferAttrib` si associa alla tabella della cronologia dei contatti Campaign. I nomi tabella visualizzati sono le tabelle di esempio create per il destinatario predefinito nelle tabelle di runtime e nelle tabelle di sistema Campaign.

Tabella 4. Attributi dell'offerta

UACI_CHOfferAttrib **Nome della colonna della tabella di staging di cronologia dei contatti Interact**	**Tabella della cronologia dei contatti Campaign**	**Nome colonna tabella**
ContactID	N/D	N/D
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

Mapping della tabella di staging della cronologia delle risposte dei contatti UACI_RHStaging

Questa tabella mostra il modo in cui la tabella di staging dell'ambiente di runtime `UACI_RHStaging` si associa alla tabella della cronologia delle risposte Campaign. I nomi tabella visualizzati sono le tabelle di esempio create per il destinatario predefinito nelle tabelle di runtime e nelle tabelle di sistema Campaign.

Tabella 5. Cronologia delle risposte

UACI_RHStaging **Nome della colonna della tabella di staging di cronologia delle risposte Interact**	**Tabella della cronologia delle risposte Campaign**	**Nome colonna tabella**
SeqNum	N/D	N/D
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime

Tabella 5. Cronologia delle risposte (Continua)

UACI_RHStaging **Nome della colonna della tabella di staging di cronologia delle risposte Interact**	**Tabella della cronologia delle risposte Campaign**	**Nome colonna tabella**
ResponseType	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

`SeqNum` è una chiave che viene utilizzata dal modulo della cronologia dei contatti e delle risposte per identificare i dati, ma non viene registrata nelle tabelle di risposte Campaign. La colonna `userDefinedFields` può contenere qualsiasi dato si desideri.

Colonne aggiuntive nelle tabelle di staging

Se si aggiungono colonne alle tabelle di staging, il modulo della cronologia dei contatti e delle risposte le scrive nelle tabelle `UA_DtlContactHist` o `UA_ResponseHistory` in colonne con lo stesso nome.

Ad esempio, se si aggiunge la colonna `linkFrom` alla tabella `UACI_CHStaging`, il modulo della cronologia dei contatti e delle risposte copia tali dati nella colonna `linkFrom` nella tabella `UA_DtlContactHist`.

Colonne aggiuntive nelle tabelle della cronologia dei contatti e delle risposte Campaign

Se si dispone di colonne aggiuntive nelle proprie tabelle della cronologia dei contatti e delle risposte Campaign, aggiungere colonne corrispondenti alle tabelle di staging prima di eseguire il modulo della cronologia dei contatti e delle risposte.

Per popolare le colonne supplementari nelle tabelle di staging, creare colonne con gli stessi nomi delle coppie nome-valore nei dati della sessione di runtime.

Ad esempio, creare coppie nome-valore `NumberItemsInWishList` e `NumberItemsInShoppingCart` e aggiungerle alla tabella `UACI_RHStaging`. Quando si verifica un evento Registra accettazione offerta o Registra rifiuto offerta, l'ambiente di runtime popola tali campi. L'ambiente di runtime popola la tabella `UACI_CHStaging` quando si verifica un evento Registra contatto offerta.

Utilizzo delle tabelle per includere un punteggio per un'offerta

È possibile utilizzare i campi definiti dall'utente per includere il punteggio che viene utilizzato per presentare un'offerta. Aggiungere una colonna denominata `FinalScore` sia alla tabella `UACI_CHStaging` nelle tabelle di runtime, sia alla tabella `UA_DtlContactHist` nelle tabelle di sistema Campaign. Interact popola automaticamente la colonna `FinalScore` con il punteggio finale utilizzato per l'offerta se si sta utilizzando l'apprendimento integrato.

Se si sta creando un modulo di apprendimento personalizzato, è possibile utilizzare il metodo `setActualValueUsed` dell'interfaccia `ITreatment` e il metodo `logEvent` dell'interfaccia `ILearning`.

Se non si utilizza l'apprendimento, aggiungere una colonna denominata `Score` sia alla tabella `UACI_CHStaging` nelle tabelle di runtime, sia alla tabella

UA_DtlContactHist nelle tabelle di sistema Campaign. Interact popola automaticamente la colonna Score con il punteggio utilizzato per l'offerta.

Creazione di nuove tabelle della cronologia in Campaign e di tabelle di staging in Interact

Se si sta utilizzando un livello destinatario diverso da Cliente, sarà necessario creare nuove tabelle della cronologia in Campaign, e nuove tabelle di staging in Interact.

Ad esempio, lo script di esempio riportato di seguito viene utilizzato nel database DB2 della fase di progettazione di IBM per creare le tabelle della cronologia in Campaign per un livello destinatario di tipo Account.

```
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;
```

Lo script di esempio riportato di seguito viene utilizzato nel database DB2 della fase di runtime di IBM per creare le tabelle di staging della cronologia in Interact per un livello destinatario di tipo Account.

```
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);
```

Configurazione del monitoraggio JMX per il modulo della cronologia dei contatti e delle risposte

Attenersi a questa procedura per configurare il monitoraggio JMX per il modulo della cronologia dei contatti e delle risposte. Sono supportati i protocolli JMXMP e RMI. La configurazione del monitoraggio JMX non abilita la sicurezza per il modulo della cronologia dei contatti e delle risposte. Si utilizza Marketing Platform per l'ambiente di progettazione per configurare il monitoraggio JMX.

Informazioni su questa attività

Per utilizzare lo strumento di monitoraggio JMX per il modulo della cronologia dei contatti e delle risposte, l'indirizzo predefinito da specificare per:

- Il protocollo JMXMP è `service:jmx:jmxmp://CampaignServer:port/campaign`.
- Il protocollo RMI è `service:jmx:rmi:///jndi/rmi://CampaignServer:port/campaign`.

Quando si visualizzano i dati nel proprio strumento di monitoraggio JMX, gli attributi dei risultati vengono organizzati prima per partizione e successivamente per livello destinatario.

Procedura

In Marketing Platform per l'ambiente di progettazione, modificare le seguenti proprietà di configurazione nella categoria `Campaign > monitoring`.

Proprietà di configurazione	Impostazione
`monitorEnabledForInteract`	**True**
port	Il numero porta per il servizio JMX
protocol	Il protocollo da utilizzare: • **JMXMP** • **RMI** La sicurezza per il modulo della cronologia dei contatti e delle risposte non è abilitata, anche se si seleziona il protocollo JMXMP.

Informazioni sul tracciamento della risposta delle sessioni incrociate

È possibile che i visitatori non riescano a completare una transazione in una sola visita al touchpoint. Un cliente può aggiungere un articolo al carrello degli acquisti sul sito Web e non completa la vendita fino a due giorni dopo. Mantenere la sessione di runtime attiva a tempo indeterminato non è fattibile. È possibile abilitare il tracciamento della risposta delle sessioni incrociate per tenere traccia di una presentazione dell'offerta in una sessione e metterla in corrispondenza con una risposta in un'altra sessione.

Il tracciamento della risposta delle sessioni incrociate di Interact può mettere in corrispondenza in base ai codici trattamento o ai codici offerta per impostazione predefinita. È anche possibile configurarlo per corrispondere a qualsiasi codice personalizzato di propria scelta. La risposta delle sessioni incrociate mette in corrispondenza in base ai dati disponibili. Ad esempio, il sito Web include un'offerta con un codice promozionale generato al momento della visualizzazione per un buono sconto per una settimana. Un utente può aggiungere articoli al

carrello degli acquisti, ma non completare l'acquisto fino a tre giorni dopo. Quando si utilizza la chiamata `postEvent` per registrare un evento di accettazione, è possibile includere solo il codice promozionale. Dal momento che il runtime non può trovare un codice trattamento o offerta di corrispondenza nella sessione corrente, il runtime pone l'evento di accettazione con le informazioni disponibili in una tabella di staging di risposte delle sessioni incrociate (XSessResponse). Il servizio CrossSessionResponse legge periodicamente la tabella XSessResponse e tenta di far corrispondere i record con i dati della cronologia dei contatti disponibili. Il servizio CrossSessionResponse mette in corrispondenza il codice promozionale con la cronologia dei contatti e raccoglie tutti i dati richiesti per registrare una risposta appropriata. Il servizio CrossSessionResponse, quindi, scrive la risposta nelle tabelle di staging di risposta, e se l'apprendimento è abilitato, nelle tabelle di apprendimento. Il modulo della cronologia dei contatti e delle risposte, quindi, scrive la risposta nelle tabelle della cronologia dei contatti e delle risposte di Campaign. L'elaborazione corretta della risposta delle sessioni incrociate dipende dai record della cronologia dei contatti originali che sono stati migrati sul database di Campaign dall'ETL della cronologia dei contatti.

Configurazione dell'origine dati del tracciamento della risposta delle sessioni incrociate

Il tracciamento della risposta delle sessioni incrociate Interact mette in corrispondenza i dati sessione dall'ambiente di runtime con la cronologia dei contatti e delle risposte Campaign. Per impostazione predefinita, il tracciamento della risposta delle sessioni incrociate mette in corrispondenza in base al codice trattamento o al codice offerta. È possibile configurare l'ambiente di runtime per la corrispondenza in base a un codice alternativo personalizzato.

- Se si sceglie di mettere in corrispondenza in base al codice alternativo, è necessario definire tale codice nella tabella `UACI_TrackingType` nelle tabelle di runtime Interact.
- L'ambiente di runtime deve disporre dell'accesso alle tabelle della cronologia dei contatti Campaign. Si può effettuare ciò configurando l'ambiente di runtime per avere accesso alle tabelle della cronologia dei contatti Campaign o creando una copia delle tabelle della cronologia dei contatti nell'ambiente di runtime.

  Questo accesso è di sola lettura ed è separato dal programma di utilità della cronologia dei contatti e delle risposte.

  Se si crea una copia delle tabelle, è responsabilità dell'utente verificare che i dati nella copia della cronologia dei contatti siano accurati. È possibile configurare il tempo per cui il servizio CrossSessionResponse conserva le risposte non corrispondenti in modo che corrisponda alla frequenza di aggiornamento dei dati nella copia delle tabelle della cronologia dei contatti con la proprietà `purgeOrphanResponseThresholdInMinutes`. Se si sta utilizzando il modulo della cronologia dei contatti e delle risposte, è necessario coordinare gli aggiornamenti ETL per assicurarsi di avere i dati più recenti.

Configurazione delle tabelle della cronologia dei contatti e delle risposte per il tracciamento della risposta delle sessioni incrociate

Sia che si crei una copia delle tabelle della cronologia dei contatti o che si utilizzino le tabelle nelle tabelle di sistema Campaign, è necessario effettuare la seguente procedura per configurare le tabelle della cronologia dei contatti e delle risposte.

Prima di iniziare

Le tabelle della cronologia dei contatti e delle risposte devono essere associate correttamente in Campaign prima di poter eseguire questi step.

Procedura

1. Eseguire lo script SQL `aci_lrnfeature` nella directory `interactDT/ddl/acifeatures` nella directory di installazione dell'ambiente di progettazione di Interact sulle tabelle `UA_DtlContactHist` e `UA_ResponseHistory` nelle tabelle di sistema Campaign.

   Questa azione aggiunge la colonna `RTSelectionMethod` alle tabelle `UA_DtlContactHist` e `UA_ResponseHistory`. Eseguire lo script `aci_lrnfeature` su queste tabelle per ciascuno dei livelli destinatario. Modificare lo script come necessario per utilizzare la tabella corretta per ciascuno dei livelli destinatario.
2. Se si desidera copiare le tabelle della cronologia dei contatti nell'ambiente di runtime, farlo ora.

   Se si sta creando una copia delle tabelle della cronologia dei contatti Campaign accessibile dall'ambiente di runtime per il supporto del tracciamento della risposta delle sessioni incrociate, utilizzare le seguenti linee guida:
   - Il tracciamento della risposta delle sessioni incrociate richiede l'accesso di sola lettura a tali tabelle.
   - Il tracciamento della risposta delle sessioni incrociate richiede le seguenti tabelle della cronologia dei contatti Campaign.
     - `UA_DtlContactHist` (per ciascun livello destinatario)
     - `UA_Treatment`

     È necessario aggiornare i dati in queste tabelle su base regolare per garantire un accurato tracciamento delle risposte.
3. Eseguire lo script SQL `aci_crhtab` nella directory `ddl` nella directory di installazione dell'ambiente di runtime Interact sull'origine dati della cronologia dei contatti e delle risposte.

   Questo script crea le tabelle `UACI_XsessResponse` e `UACI_CRHTAB_Ver`.
4. Creare una versione della tabella `UACI_XsessResponse` per ciascun livello destinatario.

Risultati

Per migliorare le prestazioni del tracciamento della risposta delle sessioni incrociate, è possibile limitare la quantità di dati della cronologia dei contatti o tramite la modalità per cui si copiano i dati della cronologia dei contatti o configurando una vista nelle tabelle della cronologia dei contatti Campaign. Ad esempio, se si segue una pratica di business per cui nessuna offerta è valida per più di 30 giorni, è necessario limitare i dati della cronologia dei contatti agli ultimi 30 giorni. Per modificare il numero di giorni per i quali conservare i dati della cronologia dei contatti, aprire la proprietà di configurazione **Campaign | partitions | partition*n*| Interact | contactAndResponseHistTracking** e impostare il valore **daysBackInHistoryToLookupContact**.

Non sarà possibile vedere i risultati del tracciamento della risposta delle sessioni incrociate finché non si esegue il modulo della cronologia dei contatti e delle risposte. Ad esempio, l'impostazione `processSleepIntervalInMinutes` predefinita è 60 minuti. Quindi, potrebbe essere necessaria almeno un'ora prima che le risposte delle sessioni incrociate vengano visualizzate nella cronologia delle risposte Campaign.

Tabella UACI_TrackingType

La tabella UACI_TrackingType fa parte delle tabelle dell'ambiente di runtime. Questa tabella definisce i codici di tracciamento utilizzati con il tracciamento della risposta delle sessioni incrociate. Il codice di tracciamento definisce il metodo utilizzato dall'ambiente di runtime per mettere in corrispondenza l'offerta corrente in una sessione di runtime con la cronologia dei contatti e delle risposte.

Colonna	Tipo	Descrizione
TrackingCodeType	int	Un numero che rappresenta il tipo di codice di tracciamento. A questo numero fanno riferimento i comandi SQL utilizzati per mettere in corrispondenza le informazioni dai dati di sessione con le tabelle della cronologia dei contatti e delle risposte.
Nome	varchar(64)	Il nome per il tipo di codice di tracciamento. Viene trasferito nei dati di sessione utilizzando il parametro riservato UACI_TrackingCodeType con il metodo postEvent.
Descrizione	varchar(512)	Una breve descrizione del tipo di codice di tracciamento. Questo campo è facoltativo.

Per impostazione predefinita, l'ambiente di runtime ha due tipi di codice di tracciamento definiti, come mostrato nella seguente tabella. Per qualsiasi altro codice, è necessario definire un TrackingCodeType univoco.

TrackingCodeType	Nome	Descrizione
1	Codice trattamento	Codice trattamento generato da UACI
2	Codice offerta	Codice offerta UAC Campaign

UACI_XSessResponse

La tabella UACI_XSessResponse fa parte delle tabelle dell'ambiente di runtime. Questa tabella viene utilizzata per il tracciamento della risposta delle sessioni incrociate.

Deve esistere una istanza di questa tabella per ciascun livello destinatario nell'origine dati della cronologia delle risposte e dei contatti disponibile per il tracciamento della risposta delle sessioni incrociate di Interact.

Colonna	Tipo	Descrizione
SeqNumber	bigint	Identificativo per la riga di dati. Il servizio CrossSessionResponse elabora tutti i record nell'ordine SeqNumber.
ICID	bigint	ID canale interattivo
AudienceID	bigint	L'ID del destinatario per questo livello destinatario. Il nome di questa colonna deve corrispondere all'ID del destinatario definito in Campaign. La tabella di esempio contiene la colonna CustomerID.
TrackingCode	varchar(64)	Il valore che viene trasferito dal parametro UACIOfferTrackingCode del metodo postEvent.
TrackingCodeType	int	La rappresentazione numerica del codice di tracciamento. Il valore deve essere una voce valida nella tabella UACI_TrackingType.

Colonna	Tipo	Descrizione
OfferID	bigint	L'ID offerta come definito in Campaign.
ResponseType	int	Il tipo di risposta per questo record. Il valore deve essere una voce valida nella tabella UA_UsrResponseType.
ResponseTypeCode	varchar(64)	Il codice del tipo di risposta per questo record. Il valore deve essere una voce valida nella tabella UA_UsrResponseType.
ResponseDate	datetime	La data della risposta.
Mark	bigint	Il valore di questo campo identifica lo stato del record. • 1 - In corso • 2 - Riuscito • NULL - Riprova • -1 - Il record è stato nel database per più di purgeOrphanResponseThresholdInMinutes minuti. Come parte della manutenzione di questa tabella da parte dell'amministratore del database, è possibile selezionare questo campo per i record che non corrispondono, ossia, tutti i record con un valore di -1. Tutti i record con il valore 2 vengono rimossi automaticamente dal servizio CrossSessionResponse.
UsrDefinedFields	char(18)	I campi personalizzati che si desidera includere quando si mettono in corrispondenza le risposte alle offerte alla cronologia dei contatti e delle risposte. Ad esempio, se si desidera eseguire la corrispondere con un codice promozionale, includere un campo definito dall'utente del codice promozionale.

Abilitazione del tracciamento della risposta delle sessioni incrociate

Utilizzare questa procedura per abilitare il tracciamento della risposta delle sessioni incrociate.

Prima di iniziare

È necessario configurare il modulo della cronologia dei contatti e delle risposte per usufruire a pieno dei vantaggi del tracciamento della risposta delle sessioni incrociate.

Per utilizzare il tracciamento della risposta delle sessioni incrociate, è necessario configurare l'ambiente di runtime per avere l'accesso in lettura alle tabelle della cronologia dei contatti e delle risposte di Campaign. È possibile leggere dalle effettive tabelle della cronologia dei contatti e delle risposte di Campaign nell'ambiente di progettazione o da una copia delle tabelle nelle origini dati dell'ambiente di runtime. L'attività di configurazione dell'ambiente di runtime per avere l'accesso in lettura alla tabella della cronologia dei contatti e delle risposte è separata dalla configurazione di qualsiasi modulo della cronologia dei contatti e delle risposte.

Se si sta mettendo in corrispondenza in base a un elemento diverso dal codice trattamento o offerta, è necessario aggiungerlo alla tabella UACI_TrackingType.

Procedura

1. Creare le tabelle XSessResponse nelle tabelle della cronologia dei contatti e delle risposte accessibili per l'ambiente di runtime.
2. Definire le proprietà nella categoria `contactAndResponseHistoryDataSource` per l'ambiente di runtime.
3. Definire la proprietà `crossSessionResponseTable` per ciascun livello destinatario.
4. Creare una categoria `OverridePerAudience` per ciascun livello destinatario.

Corrispondenza offerta/risposta delle sessioni incrociate

Per impostazione predefinita, il tracciamento della risposta delle sessioni incrociate mette in corrispondenza in base ai codici trattamento o ai codici offerta. Il servizio crossSessionResponse utilizza i comandi SQL per mettere in corrispondenza i codici trattamento, i codici offerta o un codice personalizzato dai dati di sessione alle tabelle della cronologia dei contatti e delle risposte di Campaign. È possibile modificare questi comandi SQL per la corrispondenza con qualsiasi personalizzazione effettuata ai codici di tracciamento, i codici offerta o i codici personalizzati.

Corrispondenza per codice trattamento

L'SQL per la corrispondenza in base al codice trattamento deve restituire tutte le colonne nella tabella XSessResponse per questo livello destinatario più una colonna denominata `OfferIDMatch`. Il valore nella colonna `OfferIDMatch` deve essere la `offerId` che va con il codice trattamento nel record XSessResponse.

Di seguito viene riportato un esempio del comando SQL generato per impostazione predefinita che mette in corrispondenza i codici trattamento. Interact genera l'SQL per utilizzare i nomi tabella corretti per il livello destinatario. Questo SQL è utilizzato se la proprietà `Interact > services > crossSessionResponse > OverridePerAudience > AudienceLevel > TrackingCodes > byTreatmentCode > SQL` è impostata su **Use System Generated SQL**.

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

I valori `UACI_XsessResponse`, `UA_DtlContactHist`, `CustomerID` e `UA_ContactHistory` sono definiti dalle impostazioni in Interact. Ad esempio, `UACI_XsessResponse` è definito dalla proprietà di configurazione `Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable`.

Se sono state personalizzate le tabelle della cronologia dei contatti e delle risposte, è possibile che sia necessario rivedere questo SQL per utilizzare le tabelle. Si definiscono le sovrascritture SQL nella proprietà `Interact > services > crossSessionResponse > OverridePerAudience > (AudienceLevel) > TrackingCodes > byTreatmentCode > OverrideSQL`. Se si fornisce SQL di sovrascrittura, è necessario anche modificare la proprietà `SQL` in **Override SQL**.

Corrispondenza per codice offerta

L'SQL per la corrispondenza in base al codice offerta deve restituire tutte le colonne nella tabella XSessResponse per questo livello destinatario più una colonna denominata `TreatmentCodeMatch`. Il valore nella colonna `TreatmentCodeMatch` è il codice trattamento che va con l'ID offerta (e codice offerta) nel record XSessResponse.

Di seguito viene riportato un esempio del comando SQL generato per impostazione predefinita che mette in corrispondenza i codici offerta. Interact genera l'SQL per utilizzare i nomi tabella corretti per il livello destinatario. Questo SQL è utilizzato se la proprietà `Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byOfferCode > SQL` è impostata su **Use System Generated SQL**.

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

I valori UACI_XsessResponse, UA_DtlContactHist, CustomerID e UA_ContactHistory sono definiti dalle impostazioni in Interact. Ad esempio, UACI_XsessResponse è definito dalla proprietà di configurazione Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable.

Se sono state personalizzate le tabelle della cronologia dei contatti e delle risposte, è possibile che sia necessario rivedere questo SQL per utilizzare le tabelle. Si definiscono le sovrascritture SQL nella proprietà Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byOfferCode > OverrideSQL. Se si fornisce SQL di sovrascrittura, è necessario anche modificare la proprietà SQL in **Override SQL**.

Corrispondenza per codice alternativo

È possibile definire un comando SQL per effettuare la corrispondenza in base a un codice alternativo di propria scelta. Ad esempio, è possibile avere i codici promozionali o codici prodotto separati dai codici offerta o trattamento.

È necessario definire il codice alternativo nella tabella UACI_TrackingType nelle tabelle dell'ambiente di runtime di Interact.

È necessario fornire l'SQL o una procedura memorizzata nella proprietà Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byAlternateCode > OverrideSQL che restituisce tutte le colonne nella tabella XSessResponse per questo livello destinatario oltre alle colonne TreatmentCodeMatch e OfferIDMatch. Facoltativamente è possibile restituire offerCode al posto di OfferIDMatch (nel formato offerCode1, offerCode2, ... offerCode*N* per N codici offerta). I valori nella colonna TreatmentCodeMatch e nella colonna OfferIDMatch (o nelle colonne codice offerta) devono corrispondere a TrackingCode nel record XSessResponse.

Ad esempio, il seguente pseudo codice SQL mette in corrispondenza con la colonna AlternateCode nella tabella 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>
```

Dove <*x*> è il codice di tracciamento definito nella tabella UACI_TrackingType.

Utilizzo del programma di utilità per il caricamento del database con l'ambiente di runtime

Per impostazione predefinita, l'ambiente di runtime scrive i dati della cronologia dei contatti e delle risposte dai dati di sessione nelle tabelle di staging. Su un sistema di produzione molto attivo, tuttavia, la quantità di memoria richiesta per inserire nella cache tutti i dati prima che il runtime possa scriverli nelle tabelle di staging potrebbe essere proibitiva. È possibile configurare il runtime per utilizzare un programma di utilità per il caricamento del database per migliorare le prestazioni.

Quando si abilita il programma di utilità per il caricamento del database, invece di conservare in memoria tutta la cronologia dei contatti e delle risposte prima di scriverla nelle tabelle di staging, il runtime scrive i dati in un file di staging.

Definire l'ubicazione della directory che contiene i file di staging con la proprietà `externalLoaderStagingDirectory`. Questa directory contiene diverse directory secondaria. La prima directory secondaria è la directory delle istanze di runtime, che contiene le directory `contactHist` e `respHist`. Le directory `contactHist` e `respHist` contengono directory secondarie denominate in modo univoco nel formato *audienceLevelName.uniqueID.currentState*, che contengono i file di staging.

Stato corrente	Descrizione
CACHE	Il contenuto della directory attualmente è in corso di scrittura in un file.
READY	Il contenuto della directory è pronto per essere elaborato.
RUN	Il contenuto della directory attualmente è in corso di scrittura nel database.
PROCESSED	Il contenuto della directory è stato scritto nel database.
ERROR	Si è verificato un errore durante la scrittura del contenuto della directory nel database.
ATTN	Il contenuto della directory necessita di attenzione. Vale a dire, potrebbe essere necessario effettuare alcuni step manuali per completare la scrittura del contenuto di questa directory nel database.
RERUN	Il contenuto della directory è pronto per essere scritto nel database. Ridenominare una directory da `ATTN` o `ERROR` in `RERUN` dopo aver corretto il problema.

È possibile definire la directory delle istanze di runtime definendo la proprietà JVM interact.runtime.instance.name nello script di avvio del server delle applicazioni. Ad esempio, è possibile aggiungere `-Dinteract.runtime.instance.name=instance2` allo script di avvio del server delle applicazioni web. Se non è impostato, il nome predefinito è `DefaultInteractRuntimeInstance`.

La directory `samples` contiene file di esempio per assistere l'utente durante la scrittura di propri file di controllo del programma di utilità per il caricamento del database.

Abilitazione di un programma di utilità per il caricamento del database con l'ambiente di runtime

Utilizzare questa procedura per abilitare un programma di utilità per il caricamento del database con l'ambiente di runtime.

Prima di iniziare

È necessario definire eventuali file di comando o di controllo per il programma di utilità per il caricamento del database, prima di configurare l'ambiente di runtime per l'utilizzo di tali file. Questi file devono trovarsi nella stessa ubicazione su tutti i server di runtime nel medesimo gruppo di server.

Interact fornisce file di comando e di controllo di esempio nella directory `loaderService`, nell'installazione del server di runtime Interact.

Procedura

1. Confermare che l'utente dell'ambiente di runtime disponga delle credenziali di accesso per l'origine dati delle tabelle di runtime, definita in `Interact > general > systemTablesDataSource` nelle proprietà di configurazione.
2. Definire le proprietà di configurazione `Interact > general > systemTablesDataSource > loaderProperties`.
3. Definire la proprietà `Interact > services >externalLoaderStagingDirectory`.
4. Revisionare le proprietà di configurazione `Interact > services > responseHist > fileCache`, se necessario.
5. Revisionare le proprietà di configurazione `Interact > services > contactHist > fileCache`, se necessario.
6. Riavviare il server di runtime.

Processo ETL pattern di evento

Per elaborare grosse quantità di dati di pattern di evento IBM Interact e renderli disponibili per le query e a scopo di report, è possibile installare un processo ETL (Extract, Transform, Load) autonomo sui server supportati per ottenere prestazioni ottimali.

In Interact, tutti i dati del pattern di evento di un determinato ID destinatario vengono memorizzati come singola raccolta nelle tabelle database di runtime. Le informazioni sull'ID destinatario e lo stato del pattern vengono memorizzate come BLOB (Binary Large Object). Per eseguire query SQL o report basati su pattern di evento, questo nuovo processo ETL è necessario per suddividere l'oggetto in tabelle in un database di destinazione. A tale scopo, il processo ETL in modalità autonoma prende i dati dei pattern di evento dalle tabelle database di runtime di Interact, li elabora nella pianificazione specificata e li memorizza nel database di destinazione dove sono disponibili per le query SQL o per ulteriori report.

Oltre a spostare e trasformare i dati dei pattern di evento nel database di destinazione, il processo ETL in modalità autonoma sincronizza anche i dati nel database di destinazione con le informazioni più aggiornate del database di runtime di Interact. Ad esempio, se si elimina un pattern di evento nel runtime Interact, i dati elaborati di quel pattern di evento vengono rimossi dal database di destinazione la volta successiva in cui viene eseguito il processo ETL. Anche le informazioni sullo stato del pattern di evento vengono aggiornate. Quindi le informazioni sui pattern di evento memorizzate nel database di destinazione sono solo dati correnti, non informazioni cronologiche.

Esecuzione del processo ETL in modalità autonoma

Quando si avvia il processo ETL in modalità autonoma su un server, viene eseguito di continuo in background fino a quando non viene arrestato. Il processo segue le istruzioni contenute nelle proprietà di configurazione di Marketing Platform per determinare la frequenza, le connessioni al database ed altri dettagli durante la sua operatività.

Prima di iniziare

Prima di eseguire il processo ETL in modalità autonoma, completare le attività riportate di seguito:

- È necessario disporre dell'autorizzazione di un ruolo utente amministratore di Interact.

- È necessario aver installato il processo su un server e aver configurato entrambi i file sul server e in Marketing Platform in modo corretto per la configurazione.

Nota:

Se si sta eseguendo il processo ETL su Microsoft Windows per una lingua diversa dall'inglese, utilizzare `chcp` al prompt dei comandi per impostare la code page della lingua utilizzata. Ad esempio, si potrebbe utilizzare uno dei seguenti codici: ja_jp=932, zh_cn=936, ko_kr=949, ru_ru=1251 e per de_de, fr_fr, it_it, es_es, pt_br, utilizzare 1252. Per assicurare una corretta visualizzazione dei caratteri, utilizzare il comando `chcp` nel prompt dei comandi Windows prima di avviare il processo ETL.

Informazioni su questa attività

Dopo aver installato e configurato il processo ETL in modalità autonoma, si è pronti ad avviare il processo.

Procedura

1. Aprire un prompt dei comandi sul server in cui è installato il processo ETL.
2. Andare alla directory `<Interact_home>/PatternStateETL/bin` che contiene i file eseguibili per il processo ETL.
3. Eseguire il file `command.bat` (su Microsoft Windows) o il file `command.sh` (su sistemi operativi di tipo UNIX) con i seguenti parametri:
   - `-u <username>`. Questo valore deve essere un utente Marketing Platform valido e quell'utente deve essere stato configurato con accesso alle origini dati **TargetDS** e **RuntimeDS** che il processo ETL utilizzerà.
   - `-p <password>`. Sostituire *<password>* con la password corrispondente all'utente specificato. Se la password di questo utente è vuota, specificare due doppie virgolette (come in `-p ""`). La password è facoltativa quando si esegue il file del comando; se si omette la password con il comando, ne viene richiesta l'immissione quando viene eseguito il comando.
   - `-c <profileName>`. Sostituire *<profileName>* con il nome esatto specificato in Marketing Platform nella configurazione **Interact | PatternStateETL** creata.

     Il nome immesso qui deve corrispondere al valore specificato nel campo **Nuovo nome categoria** quando è stata creata la configurazione.
   - `start`. Il comando start è richiesto per avviare il processo.

   Il comando completo per avviare il processo pertanto avrebbe il seguente formato:

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

Risultati

Il processo ETL in modalità autonoma viene eseguito, e continua ad essere eseguito, in background fino a quando non viene arrestato o quando il server viene riavviato.

Nota:

La prima volta che viene eseguito il processo, l'esecuzione dei dati accumulati del pattern di evento potrebbe impiegare una considerevole quantità di tempo. Le

volte successive in cui viene eseguito il processo utilizzeranno solo la serie più recente di dati del pattern di evento e sarà necessario meno tempo per il completamento.

Tenere presente che è possibile anche fornire l'argomento `help` per il file `command.bat` o `command.sh` per visualizzare tutte le opzioni disponibili, come nel seguente esempio:

```
command.bat help
```

Arresto del processo ETL in modalità autonoma

Quando si avvia il processo ETL in modalità autonoma su un server, viene eseguito continuamente in background fino a quando non viene arrestato.

Informazioni su questa attività

Procedura

1. Aprire un prompt dei comandi sul server in cui è installato il processo ETL.
2. Andare alla directory `<Interact_home>/PatternStateETL/bin` che contiene i file eseguibili per il processo ETL.
3. Eseguire il file `command.bat` (su Microsoft Windows) o il file `command.sh` (su sistemi operativi di tipo UNIX) con i seguenti parametri:
   - `-u <username>`. Questo valore deve essere un utente Marketing Platform valido, e quell'utente deve essere stato configurato con accesso alle origini dati **TargetDS** e **RuntimeDS** che il processo ETL utilizzerà.
   - `-p <password>`. Sostituire *<password>* con la password corrispondente all'utente specificato. Se la password di questo utente è vuota, specificare due doppie virgolette (come in `-p ""`). La password è facoltativa quando si esegue il file del comando; se si omette la password con il comando, ne viene richiesta l'immissione quando viene eseguito il comando.
   - `-c <profileName>`. Sostituire *<profileName>* con il nome esatto specificato in Marketing Platform nella configurazione **Interact | PatternStateETL** creata.

     Il nome immesso qui deve corrispondere al valore specificato nel campo **Nuovo nome categoria** quando è stata creata la configurazione.
   - `stop`. Il comando stop è richiesto per arrestare il processo. Se si utilizza questo comando, qualsiasi operazione ETL in corso verrà completata prima che il processo venga chiuso.

     Per chiudere il processo ETL senza attendere il completamento delle operazioni in corso, utilizzare `forcestop` al posto di `stop`.

   Il comando completo per avviare il processo pertanto avrebbe il seguente formato:

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

Risultati

Il processo ETL in modalità autonoma viene arrestato.

Capitolo 4. Presentazione di offerte

È possibile configurare Interact in molti modi per migliorare il metodo con cui seleziona le offerte da presentare. Le seguenti sezioni descrivono in dettagli queste funzioni facoltative.

Idoneità dell'offerta

Lo scopo di Interact è di presentare offerte idonee. Semplicemente Interact presenta le offerte ottimali tra quelle idonee, in base al visitatore, al canale e alla situazione.

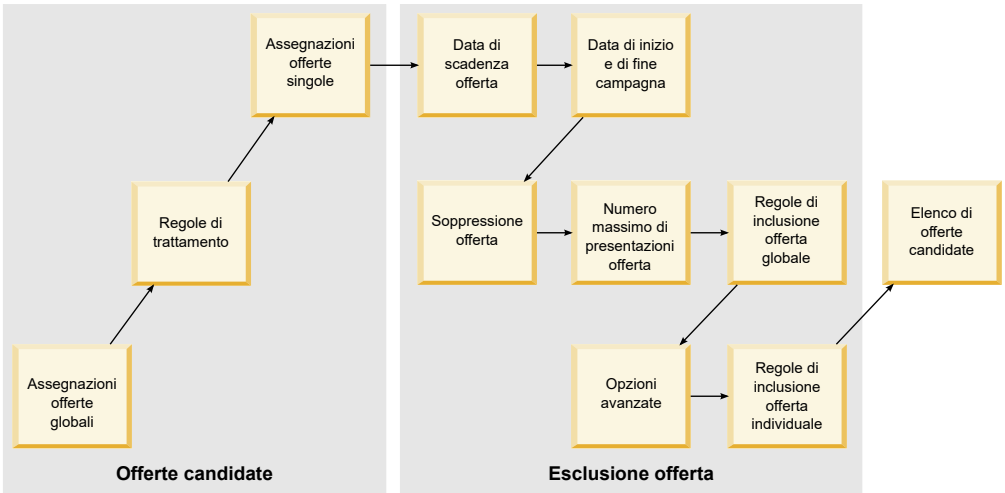
Le regole di trattamento sono solo l'inizio del modo in cui Interact determina quali offerte sono idonee per un cliente. Interact dispone di diverse funzioni facoltative che l'utente può implementare per migliorare la modalità con cui l'ambiente di runtime determina quali offerte presentare. Nessuna di queste funzioni garantisce che un'offerta venga presentata ad un cliente. Queste funzioni influenzano la probabilità che un'offerta sia idonea ad essere presentata ad un cliente. È possibile utilizzare tutte le funzioni che si ritengono necessarie per implementare la soluzione migliore per il proprio ambiente.

Sono disponibili tre aree principali in cui è possibile influenzare l'idoneità dell'offerta: la generazione dell'elenco di offerte candidate, la determinazione del punteggio di marketing e l'apprendimento.

Generazione di un elenco di offerte candidate

La generazione di un elenco di offerte candidate è costituita da due fasi principali. La prima fase genera un elenco di tutte le possibili offerte per le quali il cliente potrebbe essere idoneo. La seconda fase filtra le offerte per le quali il cliente non è più idoneo. Vi sono diversi punti in entrambe le fasi in cui è possibile influenzare la generazione dell'elenco di offerte candidate.

Questo diagramma mostra le fasi della generazione dell'elenco di offerte candidate. Le frecce indicano l'ordine di precedenza. Ad esempio, se un'offerta passa il filtro **N. massimo di volte in cui presentare un'offerta**, ma non passa il filtro **Regole di inclusione offerte globali**, l'ambiente di runtime esclude l'offerta.

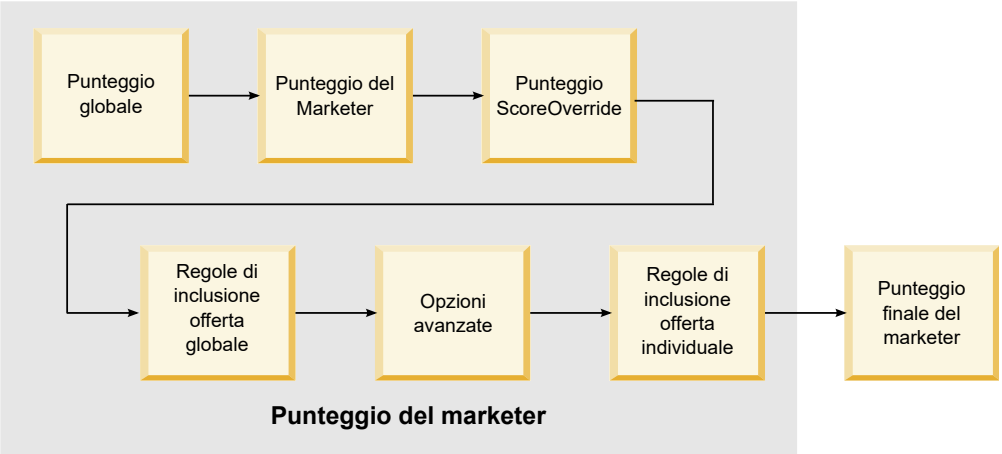


- **Assegnazioni di offerte globali** - È possibile definire le offerte globali in base al livello destinatario utilizzando la tabella delle offerte globali.
- **Regole di trattamento** - Il metodo di base per definire le offerte in base al segmento per punto di interazione utilizzando la scheda della strategia di interazione.
- **Assegnazioni di singole offerte** - È possibile definire assegnazioni di offerte specifiche per cliente utilizzando la tabella di sovrascrittura dei punteggi.
- **Data di scadenza dell'offerta** - Quando si crea un'offerta in Campaign, è possibile definire una data di scadenza. Se la data di scadenza di un'offerta è passata, l'ambiente di runtime esclude l'offerta.
- **Data di inizio e di fine della campagna** - Quando si crea una campagna in Campaign, è possibile definire una data di inizio e una data di fine della campagna. Se la data di inizio della campagna non si è verificata o la data di fine è passata, l'ambiente di runtime esclude l'offerta.
- **Soppressione dell'offerta** - È possibile definire la soppressione dell'offerta per membri specifici destinatari utilizzando la tabella delle soppressioni di offerte.
- **N. massimo di volte in cui presentare un'offerta** - Quando si definisce un canale interattivo, viene definito il numero massimo di volte in cui presentare un'offerta ad un cliente per sessione. Se l'offerta è già stata presentata per questo numero di volte, l'ambiente di runtime esclude l'offerta.
- **Regole di inclusione offerte globali** - È possibile definire un'espressione booleana per filtrare le offerte su un livello destinatario utilizzando la tabella di offerte globali. Se il risultato è false, l'ambiente di runtime esclude l'offerta.
- **Opzioni avanzate** - È possibile utilizzare l'opzione avanzata **Considera questa regola idonea se la seguente espressione è true** in una regola di trattamento per filtrare le offerte su un livello segmento. Se il risultato è false, l'ambiente di runtime esclude l'offerta.
- **Regole di inclusione offerte singole** - È possibile definire un'espressione booleana per filtrare le offerte su un livello cliente utilizzando la tabella di sovrascrittura dei punteggi. Se il risultato è false, l'ambiente di runtime esclude l'offerta.

Calcolo del punteggio di marketing

Vi sono molti modi per influenzare (utilizzando un calcolo) o sovrascrivere il punteggio di marketing.

Questo diagramma mostra le varie fasi in cui è possibile influenzare o sovrascrivere il punteggio di marketing.



Le frecce indicano l'ordine di precedenza. Ad esempio, se si definisce un'espressione per determinare il punteggio di marketing nelle opzioni avanzate per una regola di trattamento e si definisce un'espressione nella tabella di sovrascrittura del punteggio, l'espressione nella tabella di sovrascrittura del punteggio ha la precedenza.

- **Punteggio globale** - È possibile definire un punteggio per livello destinatario utilizzando la tabella delle offerte globali.
- **Punteggio marketer** - È possibile definire un punteggio per segmento utilizzando l'indicatore in una regola di trattamento.
- **Punteggio sovrascrittura del punteggio** - È possibile definire un punteggio per cliente utilizzando la tabella di sovrascrittura del punteggio.
- **Regole di inclusione delle offerte globali** - È possibile definire un'espressione che calcola un punteggio per livello destinatario utilizzando la tabella delle offerte globali.
- **Opzioni avanzate** - È possibile definire un'espressione che calcola un punteggio per segmento utilizzando l'opzione avanzata **Utilizza la seguente espressione come punteggio di marketing** in una regola di trattamento.
- **Regole di inclusione offerta di sovrascrittura punteggio** - È possibile definire un'espressione che calcola un punteggio per cliente utilizzando la tabella di sovrascrittura del punteggio.

Influenzare l'apprendimento

Se si utilizza il modulo di apprendimento integrato Interact, è possibile influenzare l'output dell'apprendimento oltre le configurazioni di apprendimento standard, come l'elenco di attributi di apprendimento o il livello di confidenza. È possibile sovrascrivere i componenti dell'algoritmo di apprendimento pur utilizzando i restanti componenti.

È possibile sovrascrivere l'apprendimento utilizzando le colonne `LikelihoodScore` e `AdjExploreScore` delle offerte predefinite e le tabelle di ovrascrittura dei punteggi. È possibile aggiungere queste colonne alle offerte predefinite e alle tabelle di sovrascrittura dei punteggi utilizzando lo script delle funzione `aci_scoringfeature`. Per utilizzare inmodo corretto le sovrascritture, è necessaria una profonda conoscenza dell'apprendimento integrato di Interact.

Il modulo di apprendimento utilizza l'elenco delle offerte candidate e il punteggio di marketing per offerta candidata nei calcoli finali. L'elenco di offerte viene utilizzato con gli attributi di apprendimento per calcolare la probabilità (probabilità di accettazione) che il cliente accetterà l'offerta. Utilizzando queste probabilità e il numero cronologico delle presentazioni per creare un equilibrio tra esplorazione e utilizzo, l'algoritmo di apprendimento determina il peso dell'offerta. Infine, l'apprendimento integrato prende il peso dell'offerta, lo moltiplica per il punteggio di marketing finale e restituisce un punteggio finale. Le offerte vengono ordinate in base a questo punteggio finale.

Soppressione di offerte

È possibile configurare l'ambiente di runtime per la soppressione di offerte.

Sono disponibili diversi modi in cui l'ambiente di runtime sopprime un'offerta:

- L'elemento **N. massimo di visualizzazioni di un'offerta durante una singola visita** di un canale interattivo.

N. massimo di visualizzazioni di un'offerta durante una singola visita viene definito quando si crea o si modifica un canale interattivo.

- L'utilizzo di una tabella delle soppressioni dell'offerta.

  La tabella delle soppressioni dell'offerta viene creata nel database dei profili.
- Offerte la cui data di scadenza è trascorsa.
- Offerte di campagne scadute
- Offerte escluse perché non passano una regola di inclusione offerta (opzione avanzata della regola di trattamento).
- Offerte già accettate o rifiutate esplicitamente in una sessione Interact. Se un cliente accetta o rifiuta esplicitamente un'offerta, quell'offerta viene soppressa durante la sessione.

Abilitazione della soppressione dell'offerta

Utilizzare questa procedura per abilitare la soppressione dell'offerta.

Informazioni su questa attività

È possibile configurare Interact per far riferimento ad un elenco di offerte soppresse.

Procedura

1. Creare una `offerSuppressionTable`, una nuova tabella per ogni destinatario che contiene l'ID destinatario e l'ID offerta.
2. Impostare la proprietà `enableOfferSuppressionLookup` su **true**.
3. Impostare la proprietà `Interact > profile > offerSuppressionTable` sul nome della tabella delle soppressioni dell'offerta per il destinatario appropriato.

Tabella delle soppressioni dell'offerta

La tabella delle soppressioni dell'offerta consente di sopprimere un'offerta per un ID destinatario specifico. Ad esempio, se il destinatario è Customer, è possibile sopprimere un'offerta per il cliente John Smith. Deve esistere una versione di questa tabella per almeno un livello destinatario nel database dei profili di produzione. È possibile creare una tabella delle soppressioni di offerta, `UACI_Blacklist` eseguendo lo script SQL `aci_usrtab` sul database dei profili. Lo script SQL `aci_usrtab` si trova nella directory `ddl` della directory di installazione dell'ambiente di runtime.

È necessario definire i campi `AudienceID` e `OfferCode1` per ciascuna riga. È possibile aggiungere ulteriori colonne se l'ID destinatario o il codice offerta è costituto da più colonne. Tali colonne devono corrispondere ai nomi colonna definiti in Campaign. Ad esempio, se si definisce il destinatario `Customer` per i campi `HHold_ID` e `MemberNum`, è necessario aggiungere `HHold_ID` e `MemberNum` alla tabella delle soppressioni dell'offerta.

Nome	Descrizione
AudienceID	(Obbligatorio) Il nome di questa colonna deve corrispondere al nome della colonna che definisce l'ID destinatario in Campaign. Se l'ID destinatario è formato da più colonne, è possibile aggiungerle a questa tabella. Ogni riga deve contenere l'ID destinatario a cui viene assegnata l'offerta predefinita, ad esempio, `customer1`.

Nome	Descrizione
OfferCode1	(Obbligatorio) Il codice offerta dell'offerta che si sta sovrascrivendo. Se i codici offerta sono costituiti da più campi, è possibile aggiungere ulteriori colonne, ad esempio `OfferCode2`, e così via.

Offerte globali e singole assegnazioni

È possibile configurare l'ambiente di runtime per assegnare specifiche offerte oltre alle regole di trattamento configurate nella scheda Strategia di interazione. È possibile definire offerte globali per qualsiasi membro di un livello destinatario e singole assegnazioni per membri destinatario specifici. Ad esempio, è possibile definire un'offerta globale per tutti i nuclei familiari per visualizzare quando non ne sono disponibili altre, e quindi creare una singola assegnazione di offerta per lo specifico nucleo familiare Smith.

È possibile limitare sia le offerte globali che le singole assegnazioni per zona, cella e regole di inclusione dell'offerta. Sia le offerte globali che le singole assegnazioni vengono configurate aggiungendo dati a tabelle specifiche nel database dei profili di produzione.

Perché le offerte globali e le singole assegnazioni funzionino correttamente, tutte i codici cella e offerta di riferimento devono esistere nella distribuzione. Per assicurarsi che i dati richiesti siano disponibili, è necessario configurare i codici cella predefiniti e la tabella `UACI_ICBatchOffers`.

Definizione dei codici cella predefiniti

Se si utilizzano le offerte predefinite o le tabelle di sovrascrittura del punteggio per le assegnazioni di offerte globali o singole, è necessario definire i codici cella predefiniti. `DefaultCellCode` viene utilizzato quando non è stato definito un codice cella in una particolare riga nelle offerte predefinite o nelle tabelle di sovrascrittura del punteggio. Il reporting utilizza questo codice cella predefinito.

Informazioni su questa attività

`DefaultCellCode` deve corrispondere al formato del codice cella definito in Campaign. Questo codice cella viene utilizzato per tutte le assegnazioni visualizzate nel reporting.

Se si definiscono codici cella predefiniti univoci, è possibile identificare facilmente le offerte assegnate dalle offerte predefinite o dalle tabelle di sovrascrittura del punteggio.

Procedura

Definire la proprietà `DefaultCellCode` per ogni livello destinatario e tipo di tabella nella categoria `IndividualTreatment`.

Definizione di offerte non utilizzate in una regola di trattamento

Se si utilizzano le offerte predefinite o le tabelle di sovrascrittura del punteggio, è necessario assicurarsi che tutti i codici offerta esistano nella distribuzione. Se è noto che tutte le offerte utilizzate nelle offerte predefinite o nelle tabelle di sovrascrittura del punteggio sono utilizzate nelle regole di trattamento, le offerte esistono nella

distribuzione. Tuttavia, qualsiasi offerta non utilizzata in una regola di trattamento deve essere definita nella tabella UACI_ICBatchOffers.

Informazioni su questa attività

La tabella UACI_ICBatchOffers esiste nelle tabelle di sistema di Campaign.

Procedura

Popolare la tabella UACI_ICBatchOffers con codici offerta che vengono utilizzati nelle tabelle di offerte predefinite o di sovrascrittura del punteggio. La tabella ha il seguente formato:

Nome colonna	Tipo	Descrizione
ICName	varchar(64)	Il nome del canale interattivo a cui è associata l'offerta. Se si utilizza la stessa offerta con due canali interattivi differenti, è necessario fornire una riga per ogni canale interattivo.
OfferCode1	varchar(64)	La prima parte del codice offerta.
OfferCode2	varchar(64)	La seconda parte del codice offerta.
OfferCode3	varchar(64)	La terza parte del codice offerta.
OfferCode4	varchar(64)	La quarta parte del codice offerta.
OfferCode5	varchar(64)	La quinta parte del codice offerta.

Informazioni sulla tabella delle offerte globali

La tabella delle offerte globali consente di definire i trattamenti a livello destinatario. Ad esempio, è possibile definire un'offerta globale per ogni membro del nucleo familiare destinatario.

È possibile definire le impostazioni globali per i seguenti elementi della presentazione dell'offerta Interact.

- Assegnazione offerta globale
- Punteggio globale del marketer, per un numero o per un'espressione
- Espressione booleana per filtrare le offerte
- Probabilità di apprendimento e peso, se si sta utilizzando l'apprendimento integrato Interact
- Sovrascrittura apprendimento globale

Assegnazione di offerte globali

Utilizzare questa procedura per configurare l'ambiente di runtime per assegnare offerte globali per un livello destinatario, oltre a tutto ciò che è definito nelle regole di trattamento.

Procedura

1. Creare una tabella denominata UACI_DefaultOffers nel database dei profili.

   Per creare la tabella UACI_DefaultOffers con le colonne corrette, utilizzare il file ddl aci_usrtab.
2. Impostare la proprietà Interact > profile > enableDefaultOfferLookup su **true**.

Tabella offerte globali

La tabella offerte globali deve esistere nel database dei profili. È possibile creare la tabella offerta globale, `UACI_DefaultOffers` eseguendo lo script SQL `aci_usrtab` per il proprio database dei profili.

Lo script SQL `aci_usrtab` si trova nella directory `ddl` della directory di installazione dell'ambiente di runtime.

È necessario definire i campi `AudienceLevel` e `OfferCode1` per ciascuna riga. Gli altri campi sono facoltativi per limitare ulteriormente le assegnazioni di offerte o influenzare l'apprendimento integrato al livello destinatario.

Per prestazioni ottimali, creare un indice su questa tabella nella colonna del livello destinatario.

Nome	Tipo	Descrizione
AudienceLevel	varchar(64)	(Obbligatorio) Il nome del livello destinatario a cui viene assegnata l'offerta predefinita, ad esempio `cliente` o `nucleo familiare`. Questo nome deve corrispondere al livello destinatario definito in Campaign.
OfferCode1	varchar(64)	(Obbligatorio) Il codice offerta per l'offerta predefinita. Se i codici offerta sono costituiti da più campi, è possibile aggiungere ulteriori colonne, ad esempio `OfferCode2` e così via. Se si sta aggiungendo questa offerta per fornire un'assegnazione di offerta globale, è necessario aggiungere questa offerta alla tabella `UACI_ICBatchOffers`.
Score	float	Un numero per definire il punteggio di marketing per questa assegnazione di offerta.
OverrideTypeID	int	Se impostato su `1`, se l'offerta non esiste nell'elenco di offerte candidate, aggiungere questa offerta all'elenco e utilizzare qualsiasi dato di punteggio per l'offerta. In generale, utilizzare `1` per fornire assegnazioni di offerte globali. Se impostato su `0`, *null* o qualsiasi altro valore diverso da 1, utilizzare i dati per l'offerta solo se l'offerta esiste nell'elenco di offerte candidate. Nella maggior parte dei casi, una regola di trattamento o una singola assegnazione sovrascriveranno questa impostazione.

Nome	Tipo	Descrizione
Predicate	varchar(4000)	È possibile immettere le espressioni in questa colonna come per le opzioni avanzate per le regole di trattamento. È possibile utilizzare le stesse variabili e macro disponibili per l'utente quando si scrivono le opzioni avanzate per le regole di trattamento. Il comportamento di questa colonna dipende dal valore presente nella colonna `EnableStateID`. • Se `EnableStateID` è `2`, questa colonna agisce come l'opzione **Considera questa regola idonea se la seguente espressione è true** nelle opzioni avanzate per le regole di trattamento per limitare questa assegnazione di offerta. Questa colonna deve contenere un'espressione booleana e deve risolversi in true per includere questa offerta. Se accidentalmente si definisce un'espressione che si risolve in un numero, qualsiasi numero diverso da zero viene considerato true e zero false. • Se `EnableStateID` è `3`, questa colonna agisce come l'opzione **Utilizza la seguente espressione come punteggio di marketing** nelle opzioni avanzate per le regole di trattamento per limitare questa offerta. Questa colonna deve contenere un'espressione che si risolve in un numero. • Se `EnableStateID` è `1`, Interact ignora qualsiasi valore presente in questa colonna.
FinalScore	float	Un numero per sovrascrivere il punteggio finale utilizzato per ordinare l'elenco finale delle offerte restituite. Questa colonna viene utilizzata se è abilitato il modulo di apprendimento integrato. È possibile implementare un proprio apprendimento per utilizzare questa colonna.
CellCode	varchar(64)	Il codice cella per un segmento interattivo distribuito a cui si desidera assegnare questa offerta predefinita. Se i codici cella sono costituiti da più campi, è possibile aggiungere le ulteriori colonne. È necessario fornire un codice cella se `OverrideTypeID` è `0` o null. Se non si include un codice cella, l'ambiente di runtime ignora questa riga di dati. Se `OverrideTypeID` è `1`, non è necessario fornire un codice cella in questa colonna. Se non si fornisce un codice cella, l'ambiente di runtime utilizza il codice cella definito nella proprietà `DefaultCellCode` per questo livello destinatario e tabella a scopo di report.
Zone	varchar(64)	Il nome della zona a cui si desidera applicare questa assegnazione di offerta. Se NULL, viene applicata a tutte le zone.

Nome	Tipo	Descrizione
EnableStateID	int	Il valore in questa colonna definisce il comportamento della colonna `Predicate`. • **1** - Non utilizzare la colonna `Predicate`. • **2** - Utilizzare `Predicate` come valore booleano per filtrare l'offerta. Segue le stesse regole dell'opzione avanzata **Considera questa regola idonea se la seguente espressione è true** advanced in una regola di trattamento. • **3** - Utilizzare `Predicate` per definire il punteggio del marketer. Segue le stesse regole dell'opzione avanzata **Utilizza la seguente espressione come punteggio di marketing** in una regola di trattamento. Qualsiasi riga in cui questa colonna è Null o qualsiasi valore diverso da 2 o 3 ignora la colonna `Predicate`.
LikelihoodScore	float	Questa colonna viene utilizzata solo per influenzare l'apprendimento integrato. È possibile aggiungere questa colonna con il ddl `aci_scoringfeature`.
AdjExploreScore	float	Questa colonna viene utilizzata solo per influenzare l'apprendimento integrato. È possibile aggiungere questa colonna con il ddl `aci_scoringfeature`.

Informazioni sulla tabella di sovrascrittura del punteggio

La tabella di sovrascrittura del punteggio consente di definire i trattamenti su un ID destinatario o un singolo livello. Ad esempio, se il livello destinatario è Visitatore, è possibile creare sovrascritture per specifici visitatori.

È possibile definire le sovrascritture per i seguenti elementi della presentazione dell'offerta Interact.

- Assegnazione singola dell'offerta
- Punteggio singolo del marketer, per un numero o per un'espressione
- Espressione booleana per filtrare le offerte
- Probabilità di apprendimento e peso, se si sta utilizzando l'apprendimento integrato
- Sovrascrittura apprendimento singola

Configurazione delle sovrascritture di punteggio

È possibile configurare Interact per utilizzare un punteggio generato da un'applicazione di modeling invece del punteggio di marketing.

Procedura

1. Creare una tabella di sovrascrittura dei punteggi per ogni livello destinatario per il quale si desidera fornire le sovrascritture.

   Per creare una tabella di sovrascrittura dei punteggi di esempio con le colonne corrette, utilizzare il file ddl `aci_usrtab`.
2. Impostare la proprietà `Interact > Profile > enableScoreOverrideLookup` su **true**.

3. Impostare la proprietà `scoreOverrideTable` sul nome della tabella di sovrascrittura per ogni livello destinatario per il quale si desidera fornire le sovrascritture.

   Non è necessario fornire una tabella di sovrascrittura dei punteggi per ogni livello destinatario.

Tabella di sovrascrittura del punteggio

La tabella di sovrascrittura del punteggio deve esistere nel database dei profili di produzione. È possibile creare una tabella di sovrascrittura dei punteggi di esempio, `UACI_ScoreOverride` eseguendo lo script SQL `aci_usrtab` sul database dei profili.

Lo script SQL `aci_usrtab` si trova nella directory `ddl` della directory di installazione dell'ambiente di runtime.

È necessario definire i campi *`AudienceID`*, `OfferCode1` e `Score` per ciascuna riga. I valori negli altri campi sono facoltativi per limitare ulteriormente le singole assegnazioni di offerte o fornire informazioni sulla sovrascrittura del punteggio per l'apprendimento integrato.

Nome	Tipo	Descrizione
AudienceID	varchar(64)	(Obbligatorio) Il nome di questa colonna deve corrispondere al nome della colonna che definisce l'ID destinatario in Campaign. La tabella di esempio creata dal file ddl `aci_usrtab` crea questa colonna come colonna `CustomerID`. Se l'ID destinatario è formato da più colonne, è possibile aggiungerle a questa tabella. Ogni colonna deve contenere l'ID destinatario a cui viene assegnata la singola offerta, ad esempio, `customer1`. Per prestazioni ottimali, creare un indice su questa colonna.
OfferCode1	varchar(64)	(Obbligatorio) Il codice offerta dell'offerta. Se i codici offerta sono costituiti da più campi, è possibile aggiungere ulteriori colonne, ad esempio `OfferCode2` e così via. Se si sta aggiungendo questa offerta per fornire un'assegnazione di singola offerta, è necessario aggiungere questa offerta alla tabella `UACI_ICBatchOffers`.
Score	float	Un numero per definire il punteggio di marketing per questa assegnazione di offerta.
OverrideTypeID	int	Se impostato su `0` o *null* (o qualsiasi altro valore diverso da 1), utilizzare i dati per l'offerta solo se l'offerta esiste nell'elenco di offerte candidate. In generale, utilizzare `0` per fornire sovrascritture di punteggio. È necessario fornire un codice cella. Se impostato su `1`, se l'offerta non esiste nell'elenco di offerte candidate, aggiungere questa offerta all'elenco e utilizzare qualsiasi dato di punteggio per l'offerta. In generale, utilizzare `1` per fornire assegnazioni di singole offerte.

Nome	Tipo	Descrizione
Predicate	varchar(4000)	È possibile immettere le espressioni in questa colonna come per le opzioni avanzate per le regole di trattamento. È possibile utilizzare le stesse variabili e macro disponibili per l'utente quando si scrivono le opzioni avanzate per le regole di trattamento. Il comportamento di questa colonna dipende dal valore presente nella colonna `EnableStateID`. • Se `EnableStateID` è `2`, questa colonna agisce come l'opzione **Considera questa regola idonea se la seguente espressione è true** nelle opzioni avanzate per le regole di trattamento per limitare questa assegnazione di offerta. Questa colonna deve contenere un'espressione booleana e deve risolversi in true per includere questa offerta. Se accidentalmente si definisce un'espressione che si risolve in un numero, qualsiasi numero diverso da zero viene considerato true e zero false. • Se `EnableStateID` è `3`, questa colonna agisce come l'opzione **Utilizza la seguente espressione come punteggio di marketing** nelle opzioni avanzate per le regole di trattamento per limitare questa offerta. Questa colonna deve contenere un'espressione che si risolve in un numero. • Se `EnableStateID` è `1`, Interact ignora qualsiasi valore presente in questa colonna.
FinalScore	float	Un numero per sovrascrivere il punteggio finale utilizzato per ordinare l'elenco finale delle offerte restituite. Questa colonna viene utilizzata se è abilitato il modulo di apprendimento integrato. È possibile implementare un proprio apprendimento per utilizzare questa colonna.
CellCode	varchar(64)	Il codice cella per un segmento interattivo a cui si desidera assegnare questa offerta. Se i codici cella sono costituiti da più campi, è possibile aggiungere le ulteriori colonne. È necessario fornire un codice cella se `OverrideTypeID` è 0 o null. Se non si include un codice cella, l'ambiente di runtime ignora questa riga di dati. Se `OverrideTypeID` è `1`, non è necessario fornire un codice cella in questa colonna. Se non si fornisce un codice cella, l'ambiente di runtime utilizza il codice cella definito nella proprietà `DefaultCellCode` per questo livello destinatario e tabella a scopo di report.
Zone	varchar(64)	Il nome della zona a cui si desidera applicare questa assegnazione di offerta. Se NULL, viene applicata a tutte le zone.

Nome	Tipo	Descrizione
EnableStateID	int	Il valore in questa colonna definisce il comportamento della colonna `Predicate`. • **1** - Non utilizzare la colonna `Predicate`. • **2** - Utilizzare `Predicate` come valore booleano per filtrare l'offerta. Segue le stesse regole dell'opzione avanzata **Considera questa regola idonea se la seguente espressione è true** advanced in una regola di trattamento. • **3** - Utilizzare `Predicate` per definire il punteggio del marketer. Segue le stesse regole dell'opzione avanzata **Utilizza la seguente espressione come punteggio di marketing** in una regola di trattamento. Qualsiasi riga in cui questa colonna è Null o qualsiasi valore diverso da 2 o 3 ignora la colonna `Predicate`.
LikelihoodScore	float	Questa colonna viene utilizzata solo per influenzare l'apprendimento integrato. È possibile aggiungere questa colonna con il ddl `aci_scoringfeature`.
AdjExploreScore	float	Questa colonna viene utilizzata solo per influenzare l'apprendimento integrato. È possibile aggiungere questa colonna con il ddl `aci_scoringfeature`.

Panoramica sull'apprendimento integrato di Interact

Pur facendo tutto il possibile per assicurarsi di proporre le giuste offerte ai giusti segmenti, è sempre possibile apprendere qualcosa dalle effettive selezioni dei visitatori. Il comportamento effettivo dei visitatori deve influenzare la strategia. È possibile utilizzare la cronologia delle risposte ed eseguirla in uno strumento di modeling per ottenere un punteggio che può essere incluso nei diagrammi di flusso interattivi.

Tuttavia, questi dati non sono in tempo reale.

Interact fornisce due opzioni per apprendere dalle azioni dei visitatori in tempo reale:

- Modulo di apprendimento integrato - L'ambiente di runtime ha un modulo di apprendimento basato su Naive-Bayesiano. Questo modulo monitora gli attributi cliente di propria scelta e utilizza quei dati per selezionare quali offerte presentare.
- API di apprendimento - L'ambiente di runtime dispone anche di un'API di apprendimento con cui poter scrivere un proprio modulo di apprendimento.

Non è necessario utilizzare l'apprendimento. Per impostazione predefinita l'apprendimento è disabilitato.

Modulo di apprendimento di Interact

Il modulo di apprendimento di Interact monitora le risposte del visitatore alle offerte e gli attributi visitatore.

Modalità del modulo di apprendimento

Il modulo di apprendimento ha due modalità generali:

- Esplorazione - il modulo di apprendimento presenta le offerte in ordine in modo da poter raccogliere dati di risposta sufficienti per ottimizzare la stima utilizzata durante la modalità utilizzo. Le offerte presentate durante l'esplorazione non riflettono necessariamente la scelta ottimale.
- Utilizzo - dopo aver raccolto dati sufficienti dalla fase di esplorazione, il modulo di apprendimento utilizza le probabilità per facilitare la selezione delle offerte da presentare.

Il modulo di apprendimento utilizza due proprietà per alternare la modalità esplorazione e la modalità utilizzo. Le due proprietà sono:

- un livello di confidenza che l'utente può configurare con la proprietà `confidenceLevel`.
- una probabilità che il modulo di apprendimento presenti un offerta casuale che l'utente può configurare con la proprietà `percentRandomSelection`.

Proprietà del livello di confidenza

`confidenceLevel` viene impostato su una percentuale che rappresenta il livello di sicurezza (o confidenza) che il modulo di apprendimento deve avere prima che i relativi punteggi per un'offerta vengano utilizzati nell'arbitraggio. Inizialmente, quando il modulo di apprendimento non dispone di dati con cui lavorare, si affida interamente al punteggio di marketing. Dopo che l'offerta è stata presentata per il numero di volte definito da `minPresentCountThreshold`, il modulo di apprendimento entra in modalità esplorazione. Con una scarsa quantità di dati con cui lavorare, il modulo di apprendimento non è sicuro che le percentuali calcolate siano corrette. Pertanto resta in modalità esplorazione.

Il modulo di apprendimento assegna i pesi a ciascuna offerta. Per calcolare i pesi, il modulo di apprendimento utilizza una formula che acquisisce come input il livello di confidenza configurato, i dati cronologici di accettazione e i dati correnti della sessione. La formula effettua intrinsecamente un bilanciamento tra esplorazione e utilizzo, e restituisce il peso appropriato.

Proprietà della selezione casuale

Per assicurarsi che il sistema non tenda verso le offerte che hanno avuto le migliori prestazioni nella prima fase, Interact presenta un'offerta casuale per una percentuale di tempo `percentRandomSelection`. Questa percentuale di offerta casuale forza il modulo di apprendimento a consigliare offerte diverse da quelle con maggior successo per determinare se altre offerte potrebbero essere più valide se avessero una maggiore esposizione. Ad esempio, se si configura `percentRandomSelection` su 5, il 5% del tempo in cui il modulo di apprendimento presenta un'offerta casuale ed aggiunge i dati di risposta ai calcoli.

È possibile impostare **% casuale** per specificare la possibilità che l'offerta restituita venga selezionata a caso, senza considerare i punteggi, per ogni zona nella scheda Punti di interazione della finestra Canale interattivo.

In che modo il modulo di apprendimento determina le offerte

Il modulo di apprendimento determina quali offerte vengono presentate nel modo riportato di seguito.

1. Calcola la probabilità che un visitatore selezioni un offerta.
2. Calcola il peso dell'offerta utilizzando la probabilità dello step 1 e determina se essere in modalità esplorazione o utilizzo.
3. Calcola un punteggio finale per ciascuna offerta utilizzando il punteggio di marketing e il peso dell'offerta dello step 2.
4. Ordina le offerte in base ai punteggi determinati allo step 3 e restituisce il numero richiesto di offerte migliori.

Ad esempio, il modulo di apprendimento determina che un visitatore ha il 30% di probabilità di accettare l'offerta A e il 70% di accettare l'offerta B e stabilisce di sfruttare queste informazioni. Dalle regole di trattamento, il punteggio di marketing per l'offerta A è 75 e 55 per l'offerta B. Tuttavia, i calcoli dello step 3 rendono il punteggio finale dell'offerta B superiore all'offerta A, pertanto l'ambiente di runtime consiglia l'offerta B.

Proprietà del fattore peso

L'apprendimento si basa anche sulle proprietà `recencyWeightingFactor` e `recencyWeightingPeriod`. Tali proprietà consentono di aggiungere un maggior peso ai dati più recenti rispetto a quelli meno recenti. `recencyWeightingFactor` è la percentuale di peso da assegnare ai dati recenti. `recencyWeightingPeriod` è il periodo di tempo considerato recente. Ad esempio, configurare il `recencyWeightingFactor` su 0.30 e `recencyWeightingPeriod` su 24. Queste impostazioni indicano che le precedenti 24 ore di dati sono il 30% di tutti i dati considerati. Per un periodo di una settimana di dati, la media di tutti i dati nei primi sei giorni è il 70% dei dati e l'ultimo giorno è 30% dei dati.

Dati scritti nella tabella di staging

Ogni sessione scrive i seguenti dati in una tabella di staging di apprendimento:

- Contatto offerta
- Accettazione offerta
- Attributi di apprendimento

Ad intervalli configurabili, un programma di aggregazione legge i dati dalla tabella di staging, li compila e li scrive in una tabella. Il modulo di apprendimento legge quei dati aggregati e li utilizza nei suoi calcoli.

Abilitazione del modulo di apprendimento

Tutti i server di runtime dispongono di un modulo di apprendimento integrato. Per impostazione predefinita il modulo di apprendimento è disabilitato. Per abilitare il modulo di apprendimento modificare una proprietà di configurazione.

Procedura

In Marketing Platform per l'ambiente di runtime, modificare le seguenti proprietà di configurazione nella categoria `Interact > offerserving`.

Proprietà di configurazione	Impostazione
`optimizationType`	**BuiltInLearning**

Attributi di apprendimento

Il modulo di apprendimento apprende utilizzando gli attributi visitatore e i dati di accettazione dell'offerta. È possibile selezionare quali attributi visitatore monitorare. Questi attributi visitatore possono essere qualsiasi elemento presente in un profilo cliente, inclusi alcuni parametri evento raccolti in tempo reale.

Gli attributi delle tabelle dimensionali non sono supportati nell'apprendimento.

Sebbene sia possibile configurare qualsiasi numero di attributi da monitorare, IBM consiglia di non configurare più di dieci attributi di apprendimento tra quelli statici e quelli dinamici, oltre a seguire queste linee guida.

- Selezionare attributi indipendenti.

  Non selezionare attributi simili. Ad esempio, se si crea un attributo denominato HighValue, e quell'attributo è definito da un calcolo che si basa sullo stipendio, non selezionare sia HighValue che Salary. Attributi simili non giovano all'algoritmo di apprendimento.
- Selezionare attributi con valori discreti.

  Se un attributo ha intervalli di valori, è necessario selezionare un valore esatto. Ad esempio, se si desidera utilizzare lo stipendio come attributo, assegnare un valore specifico ad ogni intervallo di stipendio, ad esempio l'intervallo 20.000-30.000 deve essere A, 30.001-40.000 B, e così via.
- Limitare il numero di attributi da tracciare in modo da non ostacolare le prestazioni.

  Il numero di attributi che è possibile tracciare dipende dai requisiti delle prestazioni e dall'installazione di Interact. Se possibile, utilizzare un altro strumento di modeling (ad esempio PredictiveInsight) per determinare i primi dieci attributi predittivi. È possibile configurare il modulo di apprendimento per eliminare automaticamente gli attributi non di previsione, ma che hanno anche un costo di prestazione.

È possibile gestire le prestazioni definendo sia il numero di attributi monitorati che il numero di valori per attributo monitorato. La proprietà `Campaign > partitions > partition1 > Interact > learning > maxAttributeNames` definisce il numero massimo di attributi visitatore tracciati. La proprietà `maxAttributeValues` definisce il numero massimo di valori tracciati per attributo. Tutti gli altri valori vengono assegnati a una categoria definita dal valore della proprietà `otherAttributeValue`. Tuttavia, il motore di apprendimento tiene traccia solo dei primi valori incontrati. Ad esempio, si tiene traccia dell'attributo visitatore colore degli occhi. Si è interessati solo ai valori azzurro, castano e verde, quindi si imposta `maxAttributeValues` su 3. Tuttavia, i primi tre visitatori hanno i valori azzurro, castano e nocciola. Questo significa che a tutti i visitatori con gli occhi verdi viene assegnato `otherAttributeValue`.

È inoltre possibile utilizzare gli attributi di apprendimento dinamici che consentono di definire i criteri di apprendimento in modo più specifico. Gli attributi di apprendimento dinamici consentono di apprendere in base alla combinazione di due attributi come singola voce. Ad esempio, considerare le informazioni sul profilo riportate di seguito.

ID visitatore	Tipo di carta	Saldo carta
1	Carta Oro	$1.000
2	Carta Oro	$9.000
3	Carta Bronzo	$1.000

ID visitatore	Tipo di carta	Saldo carta
4	Carta Bronzo	$9.000

Se si utilizzano gli attributi di apprendimento standard, è possibile apprendere sulla base del tipo di carta e del saldo singolarmente. I visitatori 1 e 2 verranno raggruppati in base allo stesso tipo di carta e i visitatori 2 e 4 in base al saldo della carta. Questo potrebbe non essere una metodo di previsione accurato del comportamento di accettazione dell'offerta. Se i possessori della Carta Oro tendono ad avere saldi più elevati, il comportamento del visitatore 2 potrebbe essere radicalmente diverso dal visitatore 4, che distorcerebbe gli attributi di apprendimento standard. Tuttavia, se si utilizzano gli attributi di apprendimento dinamici, l'apprendimento su ognuno di questi visitatori viene effettuato singolarmente e le previsioni saranno più accurate.

Se si utilizzano attributi di apprendimento dinamici, e il visitatore ha due valori validi per un attributo, il modulo di apprendimento seleziona il primo valore individuato.

Se si imposta la proprietà `enablePruning` su `yes`, il modulo di apprendimento determina algoritmicamente quali attributi non sono di previsione e smette di prenderli in considerazione durante il calcolo dei pesi. Ad esempio, se si sta tenendo traccia di un attributo che rappresenta il colore dei capelli e il modulo di apprendimento determina che non esiste un pattern per accettare un'offerta basata sul colore dei capelli del visitatore, il modulo di apprendimento smette di prendere in considerazione l'attributo del colore di capelli. Gli attributi vengono valutati nuovamente ogni volta che viene eseguito il processo di aggregazione di apprendimento (definito dalla proprietà `aggregateStatsIntervalInMinutes`). Anche gli attributi di apprendimento dinamici vengono eliminati.

Definizione di un attributo di apprendimento

Utilizzare questa procedura per definire un attributo di apprendimento.

Informazioni su questa attività

È possibile configurare fino al numero di `maxAttributeNames` di attributi visitatore.

`(learningAttributes)` è un modello per creare nuovi attributi di apprendimento. È necessario immettere un nuovo nome per ogni attributo. Non è possibile creare due categorie con lo stesso nome

Procedura

In Marketing Platform per l'ambiente di progettazione, modificare le seguenti proprietà di configurazione nella categoria `Campaign > partitions > partition`*`n`* `> Interact > learning`.

Proprietà di configurazione	Impostazione
`attributeName`	`attributeName` deve corrispondere al nome della coppia nome-valore nei dati del profilo. Questo nome non è sensibile al maiuscolo/minuscolo.

Definizione degli attributi di apprendimento dinamici

Per definire gli attributi di apprendimento dinamici, è necessario popolare la tabella `UACI_AttributeList` nell'origine dati di apprendimento.

Tutte le colonne in questa tabella hanno il tipo varchar(64).

Colonna	Descrizione
`AttributeName`	Il nome dell'attributo dinamico di cui si desidera apprendere. Questo valore deve essere un valore effettivo possibile in `AttributeNameCol`.
`AttributeNameCol`	Il nome colonna completo (struttura gerarchica, a partire dalla tabella profili) dove è possibile trovare `AttributeName`. Questo nome colonna non deve essere un attributo di apprendimento standard.
`AttributeValueCol`	Il nome colonna completo (struttura gerarchica, a partire dalla tabella profili) dove è possibile trovare il valore associato per `AttributeName`.

Ad esempio, prendere in considerazione la seguente tabella profili e la relativa tabella di dimensioni associata.

Tabella 6. MyProfileTable

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

Tabella 7. MyDimensionTable

KeyField	CardType	CardBalance
Key1	Carta Oro	1000
Key2	Carta Oro	9000
Key3	Carta Bronzo	1000
Key4	Carta Bronzo	9000

Di seguito viene riportato un esempio di tabella `UACI_AttributeList` corrispondente al tipo di carta e al saldo.

Tabella 8. UACI_AttributeList

AttributeName	AttributeNameCol	AttributeValueCol
Carta Oro	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance
Carta Bronzo	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance

Configurazione dell'ambiente di runtime per riconoscere moduli di apprendimento esterno

È possibile utilizzare l'API di apprendimento Java™ per scrivere un proprio modulo di apprendimento. È necessario configurare l'ambiente di runtime per riconoscere il proprio programma di utilità di apprendimento in Marketing Platform.

Informazioni su questa attività

È necessario riavviare il server di runtime di Interact perché queste modifiche diventino effettive.

Procedura

1. In Marketing Platform per l'ambiente di runtime, modificare le seguenti proprietà di configurazione nella categoria `Interact > offerserving`. Le proprietà di configurazione per l'API optimizer di apprendimento sono presenti nella categoria `Interact > offerserving > External Learning Config`.

Proprietà di configurazione	**Impostazione**
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Il nome classe dell'apprendimento esterno
`externalLearningClassPath`	Il percorso dei file di classe o JAR sul server di runtime per l'apprendimento esterno. Se si utilizza un gruppo di server e tutti i server di runtime fanno riferimento alla stessa istanza di Marketing Platform, ogni server deve avere una copia dei file di classe o JAR nella stessa ubicazione.

2. Riavviare il server di runtime Interact perché queste modifiche diventino effettive.

Capitolo 5. Nozioni relative all'API di Interact

Interact presenta in modo dinamico offerte per un'ampia gamma di touchpoint. Ad esempio, è possibile configurare l'ambiente di runtime e il proprio touchpoint per l'invio di messaggi ai dipendenti addetti al call center, per informarli delle migliori possibilità di up-sell o cross-sell per un cliente che ha chiamato ponendo una richiesta di servizio di tipo specifico. È anche possibile configurare l'ambiente di runtime e il touchpoint per fornire offerte personalizzate ad un cliente (visitatore), che è entrato in una particolare area del sito Web.

L'API (application programming interface) di Interact consente di configurare il proprio touchpoint e un server di runtime in modo che funzionino congiuntamente per presentare le migliori offerte possibili. Mediante l'API, il touchpoint può richiedere informazioni al server di runtime, per assegnare il visitatore ad un gruppo (segmento) e presentare offerte mirate per quel segmento. È, inoltre, possibile registrare i dati per un'analisi successiva per affinare le strategie di presentazione dell'offerta.

L'API di Interact consente anche la comunicazione da client utente finale a server tramite JavaScript.

Al fine di garantire la massima flessibilità possibile nell'integrazione di Interact con i propri ambienti, IBM fornisce un servizio Web, a cui si può accedere mediante l'API di Interact.

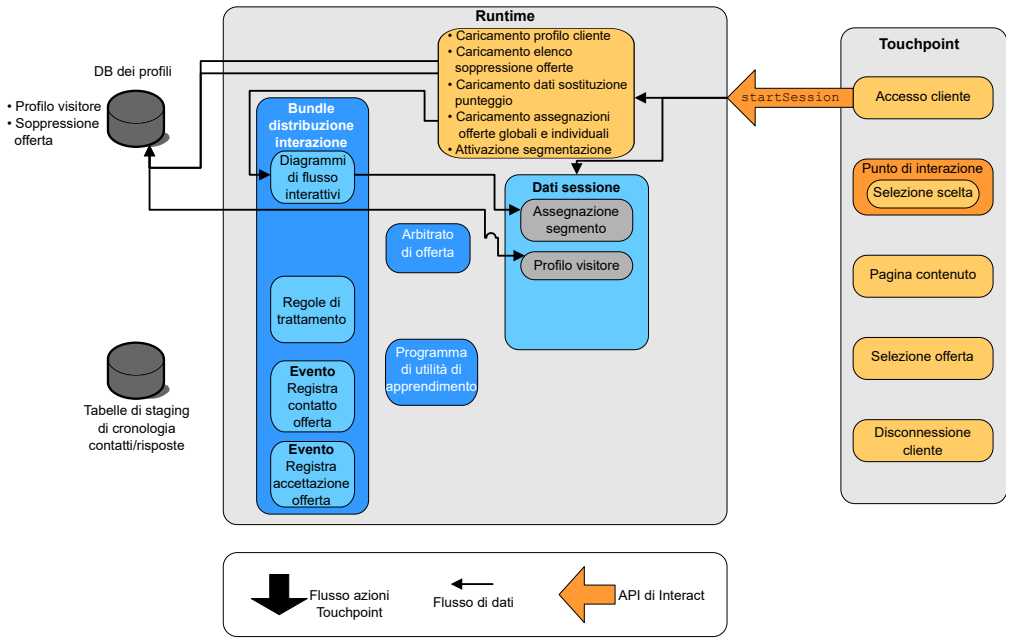
Flusso di dati dell'API di Interact

In questo esempio viene illustrato il funzionamento dell'API tra il touchpoint e l'ambiente di runtime. Il visitatore deve effettuare solo quattro azioni - accedere, andare alla pagina in cui vengono presentate le offerte, selezionare un'offerta e disconnettersi. È possibile progettare il livello di complessità desiderato per la propria integrazione, nei limiti imposti dai requisiti delle prestazioni.

Questo diagramma mostra un'implementazione semplice dell'API di Interact.

Un visitatore accede ad un sito Web e naviga fino ad una pagina che presenta delle offerte. Il visitatore seleziona un'offerta e si disconnette. Anche se l'interazione risulta semplice, si verificano vari eventi sia nel touchpoint che nel server runtime:

1. Avvio di una sessione
2. Navigazione ad una pagina
3. Selezione di un'offerta
4. Chiusura della sessione



Avvio della sessione

Quando il visitatore accede, attiva un metodo `startSession`.

Il metodo `startSession` esegue quattro operazioni:

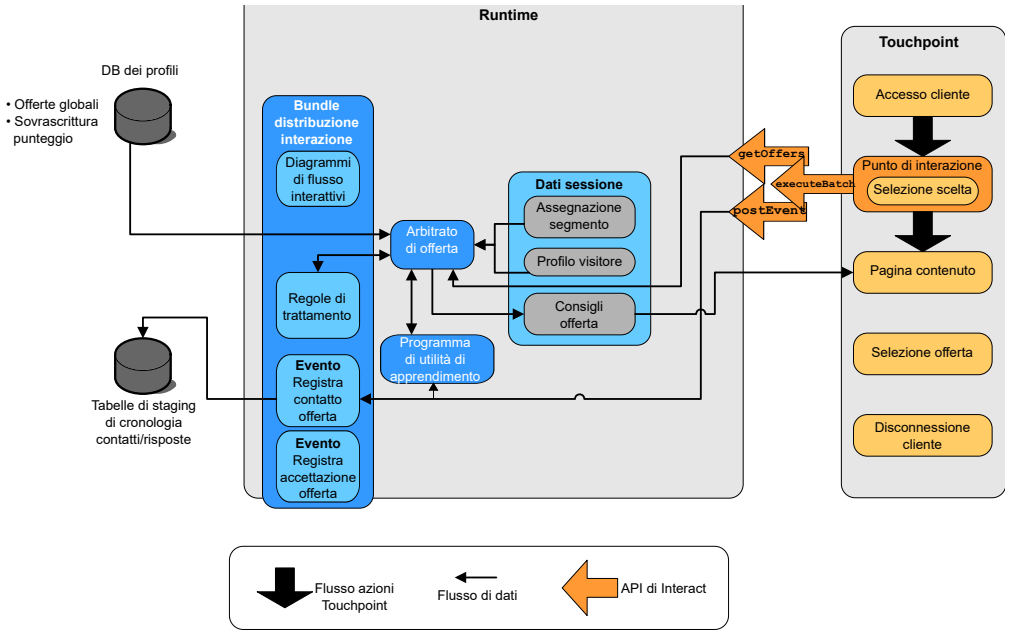
1. Crea una nuova sessione di runtime
2. Invia una richiesta per il caricamento dei dati del profilo cliente nella sessione
3. Invia una richiesta per utilizzare i dati del profilo ed avviare un diagramma di flusso interattivo per inserire il cliente in segmenti. Questa esecuzione del diagramma di flusso si svolge in modalità asincrona.
4. Il server di runtime carica eventuali informazioni relative alla soppressione dell'offerta e al trattamento di offerte globali e individuali nella sessione. I dati della sessione vengono conservati in memoria durante la sessione.

Navigazione ad una pagina

Il visitatore naviga nel sito fino a quando raggiunge un punto di interazione predefinito. Nella figura proposta, il secondo punto di interazione (Selezione scelta) è l'ubicazione di un link su cui il visitatore fa clic e che presenta una serie di offerte. Il manager del touchpoint ha configurato il link in modo che attivi un metodo `executeBatch` per la selezione di un'offerta.

Selezione di un'offerta

Questo diagramma mostra la chiamata API che attiva il metodo `executeBatch`.

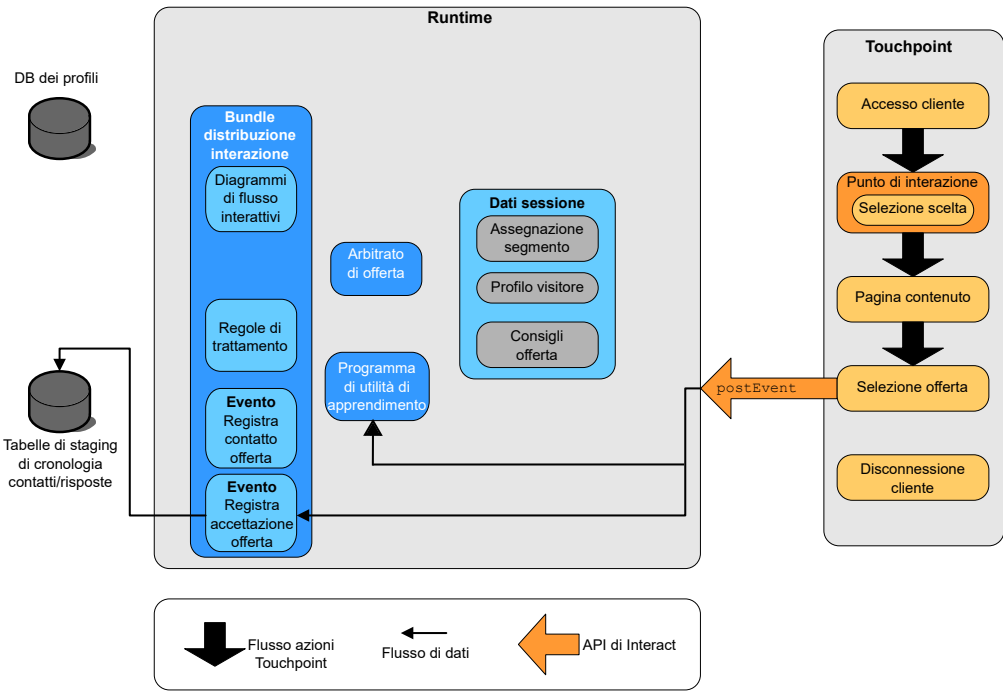


Il metodo `executeBatch` consente di richiamare più di un metodo in una singola chiamata al server di runtime. Questo metodo executeBatch in particolare richiama altri due metodi, `getOffers` e `postEvent`. Il metodo `getOffers` richiede un elenco di offerte. Il server di runtime utilizza i dati della segmentazione, l'elenco delle soppressioni dell'offerta, le regole di trattamento e il modulo di apprendimento per proporre una serie di offerte. Il server runtime restituisce una serie di offerte che vengono visualizzate nella pagina di contenuto.

Il metodo `postEvent` attiva uno degli eventi definiti nell'ambiente di progettazione. In questo caso particolare, l'evento invia una richiesta per la registrazione delle offerte presentate nella cronologia dei contatti.

Il visitatore seleziona una delle offerte (Selezione offerta).

Questo diagramma mostra il funzionamento del metodo `postEvent` .

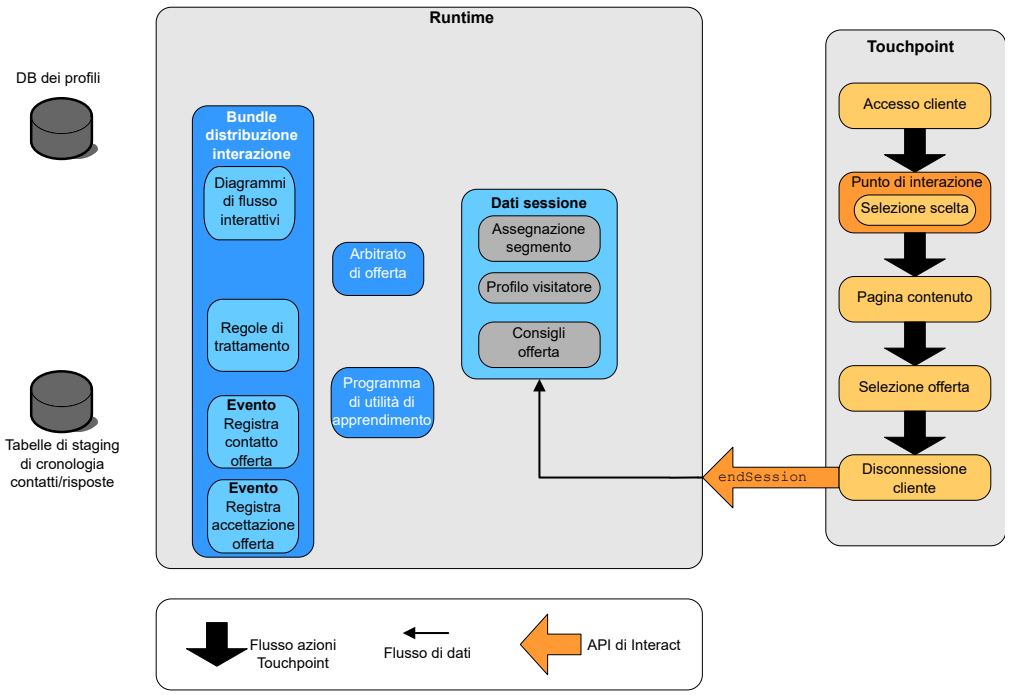


Il controllo dell'interfaccia utente associato alla selezione dell'offerta è configurato per l'invio di un altro metodo `postEvent`. Questo evento invia una richiesta per la registrazione dell'accettazione dell'offerta nella cronologia delle risposte.

Chiusura della sessione

Una volta che il visitatore seleziona l'offerta, termina la visita al sito Web e si disconnette. Il comando per la disconnessione è collegato al metodo `endSession`.

Questo diagramma mostra il funzionamento del metodo `endSession`.



Il metodo `endSession` chiude la sessione. Se il visitatore dimentica di disconnettersi, è possibile configurare un timeout di sessione, per garantire che alla fine tutte le sessioni si chiudano. Se si vogliono conservare dei dati trasmessi alla sessione, ad esempio informazioni incluse in parametri, nei metodi `startSession` o `setAudience`, collaborare con la persona che crea i diagrammi di flusso interattivi. La persona che crea un diagramma di flusso interattivo può utilizzare il processo Snapshot per scrivere tali dati in un database, prima che la sessione termini e che i dati in questione vengano persi. È possibile, quindi, utilizzare il metodo `postEvent` per richiamare il diagramma di flusso interattivo che contiene il processo Snapshot.

Esempio di pianificazione di interazione semplice

In questo esempio, si sta progettando un'interazione per un sito Web di una società di telefonia mobile. Si creano tre offerte differenti, si imposta la registrazione per le offerte, si assegnano i codici trattamento all'offerta e si presenta una serie di immagini che si collegano alle offerte.

Processo di progettazione

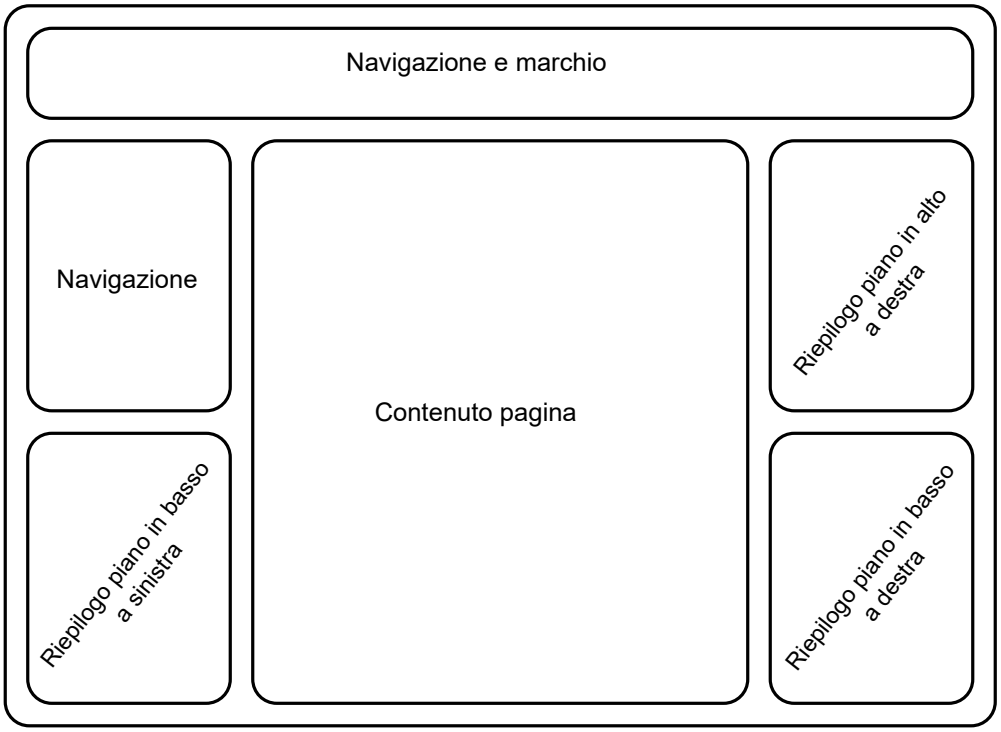
Per progettare un'interazione per questo cliente, è possibile:

1. Identificare i requisiti per la pagina di riepilogo del cliente
2. Creare i punti di interazione per i requisiti dell'offerta
3. Configurare la registrazione per le offerte
4. Creare codici trattamento
5. Collegare una serie di immagini a rotazione per le offerte

Si tratta di un esempio base e non presenta il modo migliore di scrivere codice per l'integrazione. A conferma di ciò, in nessuno di questi esempi è incluso un controllo degli errori che utilizzi la classe risposta.

Identificare i requisiti per la pagina di riepilogo del piano per la telefonia mobile

Il seguente diagramma mostra il layout per la pagina di riepilogo del piano per la telefonia mobile.



Si definiscono i seguenti elementi in modo che soddisfino i requisiti relativi alla pagina di riepilogo del piano per la telefonia mobile:

Requisito	Implementazione
Un'offerta da visualizzare in una zona dedicata alle offerte sugli aggiornamenti È necessario definire l'area nella pagina in cui viene presentata l'offerta di aggiornamento. Inoltre, dopo che Interact seleziona un'offerta da visualizzare, è necessario che le informazioni vengano registrate.	• Punto di interazione: `ip_planSummaryBottomRight` • Evento: `evt_logOffer`
Due offerte per aggiornamenti di telefonia È necessario definire ciascuna area nella pagina in cui vengono presentati gli aggiornamenti di telefonia.	• Punto di interazione: `ip_planSummaryTopRight` • Punto di interazione: `ip_planSummaryBottomLeft`
Ai fini dell'analisi, è necessario registrare quali offerte sono accettate e quali offerte vengono respinte.	• Evento: `evt_offerAccept` • Evento: `evt_offerReject`
Inoltre, si sa che è necessario trasmettere il codice trattamento di un'offerta, ogni volta che si registra un contatto, un accettazione o un rifiuto in relazione ad un'offerta.	`NameValuePair`

Requisito	Implementazione
Visualizzare sulla pagine tre immagini a rotazione. Collegare le immagini alle offerte.	

Creare punti di interazione

Ora è possibile chiedere all'utente dell'ambiente di progettazione di creare i punti di interazione e gli eventi, mentre si inizia a scrivere il codice per l'integrazione con il proprio touchpoint.

Per ciascun punto di interazione in cui viene presentata un'offerta, è necessario innanzitutto disporre di un'offerta, quindi, estrarre le informazioni necessarie per visualizzare l'offerta. Ad esempio, richiedere un'offerta per l'area in basso a destra della pagina Web (`planSummaryBottomRight`)

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

Questa chiamata di risposta restituisce un oggetto risposta che include una risposta `OfferList`. Tuttavia, la pagina Web non può utilizzare un oggetto `OfferList`. È necessario un file di immagine per l'offerta, che si sa essere uno degli attributi dell'offerta (`offerImg`). Si deve estrarre l'attributo dell'offerta di cui si ha bisogno da `OfferList`.

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

Configurare la registrazione

Ora che si sta visualizzando l'offerta, la si vuole registrare come contatto.

```
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)
```

Invece di effettuare chiamate singole ad ognuno di questi metodi, è possibile utilizzare il metodo `executeBatch`, come indicato nel seguente esempio per la parte `planSummaryBottomLeft` della pagina Web.

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

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

/** Build command array */
```

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

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

Non è necessario definire il parametro UACIOfferTrackingCode, in questo esempio. Il server di runtime Interact registra automaticamente l'ultimo elenco consigliato di trattamenti come contatti, se non viene specificato UACIOfferTrackingCode.

Creare codici trattamento

Ove necessario, si crea una voce NameValuePair che contenga il codice trattamento, come nel seguente esempio.

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

Collegare immagini alle offerte

Per la seconda area nella pagina che presenta un aggiornamento di telefonia, è stata configurata la modifica dell'immagine visualizzata ogni 30 secondi. Si decide la rotazione tra tre immagini e si utilizza il seguente codice per richiamare la serie di offerte da memorizzare nella cache per l'utilizzo nel proprio codice per la rotazione delle immagini.

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

È necessario scrivere il proprio codice cliente, estrarlo dalla cache locale e registrarlo nei contatti una sola volta per ogni offerta, dopo la visualizzazione della relativa immagine. Per registrare il contatto, è necessario inviare il parametro UACITrackingCode, come in precedenza. Ogni offerta ha un codice di tracciamento differente.

```
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);
  }
 }
}
```

Per ogni offerta, se l'offerta viene selezionata, si registra l'offerta accettata e quelle che sono state rifiutate. (In questo scenario, le offerte non esplicitamente selezionate vengono considerate rifiutate.) Quello che segue è un esempio del caso in cui venga selezionata l'offerta `ip_planSummaryTopRight`:

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

In pratica, la scelta migliore sarebbe quella di inviare queste tre chiamate a `postEvent` mediante il metodo `executeBatch`.

Progettazione dell'integrazione dell'API di Interact

La creazione dell'integrazione dell'API di Interact con il touchpoint richiede una qualche attività di progettazione, prima di poter iniziare l'implementazione. È necessario collaborare con il team del marketing per decidere in quale ubicazione nel touchpoint si desidera che l'ambiente di runtime presenti le offerte (definire i propri punti di interazione) e quale altro tipo di tracciamento o funzionalità interattiva si intende utilizzare (definire i propri eventi).

Nella fase di progettazione, queste possono essere considerate semplici indicazioni generali. Ad esempio, per un sito Web di telecomunicazioni, la pagina di riepilogo del piano del cliente dovrebbe presentare un'offerta relativa all'aggiornamento del piano e due offerte per gli aggiornamenti di telefonia.

Una volta che l'azienda ha stabilito dove e come interagire con i clienti, è necessario utilizzare Interact per definire i dettagli. Un autore di diagrammi di flusso deve progettare i diagrammi di flusso interattivi che verranno utilizzati quando si verificano eventi di risegmentazione. È necessario stabilire il numero e i nomi dei punti di interazione e degli eventi, oltre ai dati che devono essere

trasmessi perché si svolgano in modo appropriato la segmentazione, l'invio di eventi e il richiamo di offerte. L'utente dell'ambiente di progettazione definisce i punti di interazione e gli eventi per il canale interattivo. Si utilizzeranno, quindi, tali nomi quando si codifica l'integrazione con il touchpoint nell'ambiente di runtime. Si dovrebbe, inoltre, stabilire quali informazioni di metrica sono richieste, per definire quando è necessario registrare i contatti e le risposte per l'offerta.

Punti da tenere in considerazione

Quando si progetta un'interazione, tenere a mente le conseguenze la mancanza di un'offerta idonea, un server di runtime non raggiungibile, la tempistica del processo hanno sull'interazione. Essere specifici quando si definiscono i rifiuti dell'offerta. Prendere in considerazione le funzioni facoltative del prodotto che sono in grado di migliorare l'interazione.

Quando si progetta la propria interazione:

Creare contenuto predefinito di riempimento
Creare contenuto predefinito di riempimento, un messaggio di promozione del marchio o spazio senza testo, per ogni punto di interazione in cui si possono presentare offerte. Tale contenuto di riempimento viene utilizzato quando non ci sono offerte idonee da presentare per il visitatore corrente, nella situazione attuale. Si assegna questo contenuto di riempimento come stringa predefinita per il punto di interazione.

Includere un metodo alternativo di presentazione del contenuto
Includere un metodo di presentazione del contenuto da utilizzare nel caso in cui il touchpoint non riesca a raggiungere il gruppo dei server di runtime per qualche ragione imprevedibile.

Considerare il tempo impiegato per l'esecuzione dei diagrammi di flusso
Quando si attivano eventi che implicano la risegmentazione del visitatore, inclusi `postEvent` e `setAudience`, tenere a mente che l'esecuzione di diagrammi di flusso richiede un certo tempo. L'esecuzione del metodo `getOffers` attende il completamento della segmentazione prima di avviarsi. Una risegmentazione troppo frequente può penalizzare le prestazioni di risposta della chiamata a `getOffers`.

Decidere cosa si intende per "rifiuto dell'offerta"
Diversi report, come ad esempio il report di riepilogo delle prestazioni delle offerte del canale, presenta il numero di volte che un'offerta è respinta. Questo report mostra il numero di volte in cui un metodo `postEvent` ha attivato un'azione Registra rifiuto offerta. È necessario stabilire se l'azione Registra rifiuto offerta è conseguente ad un rifiuto effettivo, ad esempio fare clic su un link con etichetta **No, grazie**. Oppure se l'azione Registra rifiuto offerta è relativa ad un'offerta ignorata, ad esempio una page che presenta tre differenti banner pubblicitari, nessuno dei quali viene selezionato.

Decidere quali funzioni di selezione dell'offerta utilizzare
Esistono diverse funzioni facoltative che è possibile utilizzare per ottimizzare la selezione di offerte di Interact. Queste funzioni includono:

- Apprendimento
- Soppressione dell'offerta
- Assegnazioni di offerte individuali
- Altri elementi di presentazione dell'offerta

È necessario determinare quante, se non nessuna, di queste funzioni facoltative potrebbero ottimizzare le proprie interazioni.

Capitolo 6. Gestione dell'API di IBM Interact

Ogni volta che si utilizza il metodo `startSession`, si crea una sessione di runtime Interact sul server di runtime. Si possono utilizzare proprietà di configurazione per gestire le sessioni su un server di runtime.

È possibile che sia necessario configurare queste impostazioni quando si implementa la propria integrazione di Interact con il touchpoint.

Queste proprietà di configurazione fanno parte della categoria `sessionManagement`.

La locale e l'API di Interact

È possibile utilizzare Interact per touchpoint in lingua diversa dall'Inglese. Il touchpoint e tutte le stringhe nell'API utilizzano la locale definita per l'utente dell'ambiente di runtime.

È possibile selezionare una sola locale per gruppo di server.

Ad esempio, nell'ambiente di runtime, è possibile creare due utenti, asm_admin_en con la locale utente impostata su Inglese e asm_admin_fr con la locale utente impostata su Francese. Se il touchpoint è progettato per utenti di lingua francese, definire la proprietà `asmUserForDefaultLocale` per l'ambiente di runtime come asm_admin_fr.

Informazioni sul monitoraggio JMX

Interact fornisce il servizio di monitoraggio JMX (Java Management Extensions), a cui si può accedere mediante una qualsiasi applicazione di monitoraggio JMX. Questo monitoraggio JMX consente di monitorare e gestire i server di runtime.

Gli attributi JMX offrono una quantità di informazioni dettagliate sui server di runtime. Ad esempio, l'attributo JMX `ErrorCount` fornisce il numero dei messaggi di errore registrati a partire dall'ultima reimpostazione o dall'ultimo riavvio del sistema. È possibile fare riferimento a questa informazione, per capire con quale frequenza si verificano errori nel proprio sistema. Se il proprio sito Web è stato codificato in modo da richiamare un'istruzione di fine sessione solo se un utente completa una transazione, si potrebbero anche mettere a confronto i valori di `startSessionCount` e `endSessionCount` per sapere quante transazioni risultano non complete.

Interact supporta i protocolli RMI e JMXMP, secondo la definizione di JSR 160. È possibile connettersi al servizio di monitoraggio JMX con qualsiasi client JMX compatibile con JSR160.

I diagrammi di flusso interattivi possono essere monitorati solo mediante il monitoraggio JMX. Informazioni sui diagrammi di flusso interattivi non vengono visualizzate nel monitoraggio Campaign.

Nota: se si sta utilizzando IBM WebSphere con un gestore nodi, sarà necessario definire l'Argomento JVM generico per abilitare il monitoraggio JMX.

Configurazione di Interact per l'utilizzo del monitoraggio JMX con il protocollo RMI

Utilizzare questa procedura per configurare Interact per utilizzare il monitoraggio JMX con il protocollo RMI.

Informazioni su questa attività

L'indirizzo predefinito del monitoraggio per il protocollo RMI è `service:jmx:rmi:///jndi/rmi://RuntimeServer:port/interact`.

Procedura

In Marketing Platform per l'ambiente di runtime, modificare le seguenti proprietà di configurazione nella categoria `Interact > monitoring`.

Proprietà di configurazione	Impostazione
`protocol`	**RMI**
`port`	Il numero porta per il servizio JMX
`enableSecurity`	**False** L'implementazione Interact del protocollo RMI non supporta la sicurezza.

Configurazione di Interact per l'utilizzo del monitoraggio JMX con il protocollo JMXMP

Utilizzare questa procedura per configurare Interact per utilizzare il monitoraggio JMX con il protocollo JMXMP.

Prima di iniziare

Il protocollo JMXMP richiede due librerie supplementari nel seguente ordine, nel percorso classi, `InteractJMX.jar` e `jmxremote_optional.jar`. Entrambi i file sono reperibili nella directory `lib` della propria installazione dell'ambiente di runtime.

Informazioni su questa attività

Se si abilita la sicurezza, il nome utente e la password devono corrispondere a un utente in Marketing Platform per l'ambiente di runtime. Non è possibile utilizzare una password vuota.

L'indirizzo predefinito di monitoraggio per il protocollo JMXMP è `service:jmx:jmxmp://RuntimeServer:port`.

Procedura

1. Verificare che le librerie `InteractJMX.jar` e `jmxremote_optional.jar` si trovino, nell'ordine, nel percorso classi. Se non si trovano nel percorso classi, aggiungerle a tale percorso.
2. In Marketing Platform per l'ambiente di runtime, modificare le seguenti proprietà di configurazione nella categoria `Interact > monitoring`.

Proprietà di configurazione	Impostazione
`protocol`	**JMXMP**

Proprietà di configurazione	Impostazione
port	Il numero porta per il servizio JMX
enableSecurity	**False** per disabilitare la sicurezza or **True** per abilitarla.

Configurazione di Interact per l'utilizzo degli script jconsole per il monitoraggio JMX

Se non si dispone di un'applicazione di monitoraggio JMX separata, è possibile utilizzare la jconsole installata con la JVM. È possibile avviare la jconsole con gli script di avvio nella directory Interact/tools.

Informazioni su questa attività

Lo script jconsole utilizza il protocollo JMXMP per il monitoraggio, per impostazione predefinita. Le impostazioni predefinite per jconsole.bat sono:

La connessione 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 connessione 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
```

Procedura

1. Aprire Interact\tools\jconsole.bat (Windows) oppure Interact/tools/jconsole.sh (UNIX) in un editor di testo.
2. Impostare INTERACT_LIB sul percorso completo per la directory *InteractInstallationDirectory*/lib.
3. Impostare HOST sul nome host del server di runtime che si desidera monitorare.
4. Impostare PORT sulla porta su cui JMX è stato configurato per restare in ascolto mediante la proprietà Interact > monitoring > port.
5. Opzionale: Se si sta utilizzando il protocollo RMI per il monitoraggio, aggiungere un commento prima della connessione JMXMP e rimuovere il commento prima della connessione RMI.

Attributi JMX

Sono disponibili più attributi per il monitoraggio JMX. Gli attributi dell'ambiente di progettazione includono il monitoraggio dell'ETL della cronologia dei contatti e delle risposte. Gli attributi dell'ambiente di runtime includono statistiche su pool di thread, programma di registrazione, locale, vari attributi differenti del diagramma di flusso ed eccezioni . Sono disponibili anche diversi attributi inerenti statistiche sui servizi. Tutti i dati forniti dal monitoraggio JMX risalgono all'ultima reimpostazione o all'ultimo avvio del sistema. Ad esempio, per conteggio si intende il numero di elementi a partire dall'ultima reimpostazione o dall'ultimo avvio del sistema, non dall'installazione.

Attributi di monitoraggio dell'ETL della cronologia dei contatti e delle risposte

Gli attributi di monitoraggio dell'ETL della cronologia dei contatti e delle risposte fanno parte dell'ambiente di progettazione. Tutti gli attributi seguenti sono parte dell'ambiente di runtime.

Tabella 9. Monitoraggio dell'ETL della cronologia dei contatti e delle risposte

Attributo	Descrizione
AvgCHExecutionTime	Il numero medio di millisecondi impiegati dal modulo della cronologia dei contatti e delle risposte per scrivere nella tabella della cronologia dei contatti. Questo valore medio viene calcolato solo per le operazioni che hanno avuto esito positivo e per cui è stato scritto almeno un record nella tabella della cronologia dei contatti.
AvgETLExecutionTime	Il numero medio di millisecondi impiegati dal modulo della cronologia dei contatti e delle risposte per leggere dati dall'ambiente di runtime. Nel valore medio è incluso il tempo per le operazioni con esito positivo e per quelle con esito negativo.
AvgRHExecutionTime	Il numero medio di millisecondi impiegati dal modulo della cronologia dei contatti e delle risposte per scrivere nella tabella della cronologia delle risposte. Questo valore medio viene calcolato solo per le operazioni che hanno avuto esito positivo e per cui è stato scritto almeno un record nella tabella della cronologia delle risposte.
ErrorCount	Il numero di messaggi di errore che sono stati registrati a partire dall'ultima reimpostazione o dall'ultimo riavvio del sistema, se presenti.
HighWaterMarkCHExecutionTime	Il numero massimo di millisecondi impiegati dal modulo della cronologia dei contatti e delle risposte per scrivere nella tabella della cronologia dei contatti. Questo valore viene calcolato solo per le operazioni che hanno avuto esito positivo e per cui è stato scritto almeno un record nella tabella della cronologia dei contatti.
HighWaterMarkETLExecutionTime	Il numero massimo di millisecondi impiegati dal modulo della cronologia dei contatti e delle risposte per leggere dati dall'ambiente di runtime. Nel calcolo sono incluse le operazioni con esito positivo e quelle con esito negativo.

Tabella 9. Monitoraggio dell'ETL della cronologia dei contatti e delle risposte (Continua)

Attributo	Descrizione
HighWaterMarkRHExecutionTime	Il numero massimo di millisecondi impiegati dal modulo della cronologia dei contatti e delle risposte per scrivere nella tabella della cronologia delle risposte. Questo valore viene calcolato solo per le operazioni che hanno avuto esito positivo e per cui è stato scritto almeno un record nella tabella della cronologia delle risposte.
LastExecutionDuration	Il numero di millisecondi impiegati dal modulo della cronologia dei contatti e delle risposte per l'esecuzione dell'ultima copia.
NumberOfExecutions	Il numero di volte in cui il modulo della cronologia dei contatti e della risposte è stato eseguito a partire dall'inizializzazione.
LastExecutionStart	La data/ora in cui si è avviata l'ultima esecuzione del modulo della cronologia dei contatti e delle risposte.
LastExecutionSuccessful	Se il valore di questo attributo è true, l'ultima esecuzione del modulo della cronologia dei contatti e delle risposte ha avuto esito positivo. Se il valore è false, si è verificato un errore.
NumberOfContactHistoryRecordsMarked	Il numero di record della cronologia dei contatti, nella tabella `UACI_CHStaging`, che vengono spostati durante l'esecuzione corrente del modulo della cronologia dei contatti e delle risposte. Questo valore è maggiore di zero solo se il modulo della cronologia dei contatti e delle risposte è in esecuzione.
NumberOfResponseHistoryRecordsMarked	Il numero di record della cronologia delle risposte, nella tabella `UACI_RHStaging`, che vengono spostati durante l'esecuzione corrente del modulo della cronologia dei contatti e delle risposte. Questo valore è maggiore di zero solo se il modulo della cronologia dei contatti e delle risposte è in esecuzione.

Attributi Eccezione

Gli attributi Eccezione fanno parte dell'ambiente di runtime.

Tabella 10. Eccezioni

Attributo	Descrizione
errorCount	Il numero di messaggi di errore che sono stati registrati a partire dall'ultima reimpostazione o dall'ultimo riavvio del sistema.
warningCount	Il numero di messaggi di avvertenza che sono stati registrati a partire dall'ultima reimpostazione o dall'ultimo riavvio del sistema.

Attributi Statistiche motore diagramma di flusso

Gli attributi Statistiche motore diagramma di flusso fanno parte dell'ambiente di runtime.

Tabella 11. Statistiche motore diagramma di flusso

Attributo	Descrizione
activeProcessBoxThreads	Il conteggio attivo dei thread del processo del diagramma di flusso (condivisi tra tutte le esecuzioni) attualmente in esecuzione.
activeSchedulerThreads	Il conteggio attivo dei thread dello scheduler del diagramma di flusso attualmente in esecuzione.
avgExecutionTimeMillis	Il tempo medio, in millisecondi, di esecuzione del diagramma di flusso.
CurrentJobsInProcessBoxQueue	Il numero di job in attesa di essere eseguiti dai thread del processo del diagramma di flusso.
CurrentJobsInSchedulerQueue	Il numero di job in attesa di essere eseguiti dai thread dello scheduler del diagramma di flusso.
maximumProcessBoxThreads	Il numero massimo di thread del processo del diagramma di flusso (condivisi tra tutte le esecuzioni) che è possibile eseguire.
maximumSchedulerThreads	Il numero massimo di thread dello scheduler del diagramma di flusso (un thread per esecuzione) che è possibile eseguire.
numExecutionsCompleted	Il numero totale di esecuzioni di diagrammi di flusso completate.
numExecutionsStarted	Il numero totale di esecuzioni di diagrammi di flusso avviate.

Attributi Diagrammi di flusso specifici per canale interattivo

Gli attributi Diagrammi di flusso specifici per canale interattivo fanno parte dell'ambiente di runtime.

Tabella 12. Diagrammi di flusso specifici per canale interattivo

Attributo	Descrizione
AvgExecutionTimeMillis	Il tempo di esecuzione medio, in millisecondi, per questo diagramma di flusso in questo canale interattivo.
HighWaterMarkForExecutionTime	Il tempo di esecuzione massimo, in millisecondi, per questo diagramma di flusso in questo canale interattivo.
LastCompletedExecutionTimeMillis	Il tempo di esecuzione in millisecondi per l'ultimo completamento di questo diagramma in questo canale interattivo.
NumExecutionsCompleted	Il numero totale di esecuzioni che sono state completate per questo diagramma di flusso in questo canale interattivo.
NumExecutionsStarted	Il numero totale di esecuzioni che sono state avviate per questo diagramma di flusso in questo canale interattivo.

Attributi Locale

Gli attributi Locale fanno parte dell'ambiente di runtime.

Tabella 13. Locale

Attributo	Descrizione
locale	Impostazione della locale per il client JMX.

Attributi Configurazione programma di registrazione

Gli attributi Configurazione programma di registrazione fanno parte dell'ambiente di runtime.

Tabella 14. Configurazione programma di registrazione

Attributo	Descrizione
category	Modificare la categoria di log su cui è possibile manipolare il livello di registrazione.

Attributi Statistiche pool di thread di servizi

Gli attributi Statistiche pool di thread di servizi fanno parte dell'ambiente di runtime.

Tabella 15. Statistiche pool di thread di servizi

Attributo	Descrizione
activeContactHistThreads	Il numero approssimativo di thread che stanno attivamente eseguendo delle attività per la cronologia dei contatti e per la cronologia delle risposte.
activeFlushCacheToDBThreads	Il numero approssimativo di thread che stanno attivamente eseguendo delle attività per cancellare le statistiche memorizzate nella cache, nell'archivio dati.
activeOtherStatsThreads	Il numero approssimativo di thread che stanno attivamente eseguendo delle attività per Stat. idoneità, Attività evento e Stat. valore predefinito.
CurrentHighWaterMarkInContactHistQueue	Il numero massimo di voci in coda per essere registrate dal servizio che raccoglie i dati della cronologia dei contatti e delle risposte.
CurrentHighWaterMark InFlushCachetoDBQueue	Il numero massimo di voci in coda per essere registrate dal servizio che scrive i dati contenuti nella cache, nelle tabelle database.
CurrentHighWaterMarkInOtherStatsQueue	Il numero massimo di voci in coda per essere registrate dal servizio che raccoglie le statistiche di idoneità dell'offerta, le statistiche di utilizzo della stringa predefinita, le statistiche di attività evento e il log personalizzato in dati tabella.
currentMsgsInContactHistQueue	Il numero di job nella coda per il pool di thread utilizzato per la cronologia dei contatti e per la cronologia delle risposte.
currentMsgsInFlushCacheToDBQueue	Il numero di job nella coda per il pool di thread utilizzato per cancellare le statistiche memorizzate nella cache, nell'archivio dati.
currentMsgsInOtherStatsQueue	Il numero di job nella coda per il pool di thread utilizzato per Stat. idoneità, Attività evento e Stat. valore predefinito.
maximumContactHistThreads	Il numero massimo di thread che sono sempre stati simultaneamente nel pool utilizzato per la cronologia dei contatti e per la cronologia delle risposte.
maximumFlushCacheToDBThreads	Il numero massimo di thread che sono sempre stati simultaneamente nel pool utilizzato per cancellare le statistiche memorizzate nella cache, nell'archivio dati.

Tabella 15. Statistiche pool di thread di servizi (Continua)

Attributo	Descrizione
maximumOtherStatsThreads	Il numero massimo di thread che sono sempre stati simultaneamente nel pool utilizzato per Stat. idoneità, Attività evento e Stat. valore predefinito.

Attributi Statistiche servizio

Le statistiche servizio sono composte da una serie di attributi per ogni servizio.

- ContactHistoryMemoryCacheStatistics - Il servizio che raccoglie i dati per le tabelle di staging della cronologia dei contatti.
- CustomLoggerStatistics - Il servizio che raccoglie i dati personalizzati da scrivere in una tabella (un evento che utilizza il parametro evento `UACICustomLoggerTableName`).
- Statistiche valore predefinito - Il servizio che raccoglie le statistiche relative al numero di volte in cui è stata utilizzata la stringa predefinita per il punto di interazione.
- Statistiche idoneità - il servizio che scrive le statistiche per le offerte idonee.
- Statistiche attività evento - Il servizio che raccoglie le statistiche sull'evento, sia eventi di sistema quali `getOffer` o `startSession` che eventi utente attivati da `postEvent`.
- Statistiche cache della memoria della cronologia delle risposte - Il servizio che scrive nelle tabelle di staging della cronologia delle risposte.
- Statistiche della risposta delle sessioni incrociate - Il servizio che raccoglie i dati del tracciamento della risposta delle sessioni incrociate.

Tabella 16. Statistiche Servizio

Attributo	Descrizione
Count	Il numero di messaggi elaborati.
ExecTimeInsideMutex	La quantità di tempo impiegata per l'elaborazione dei messaggi per questo servizio, escluso il tempo trascorso in attesa di altri thread, in millisecondi. Se si presenta una grande differenza tra il valore di ExecTimeInsidMutex e quello di ExecTimeMillis, potrebbe essere necessario modificare la dimensione del pool di thread per il servizio.
ExecTimeMillis	La quantità di tempo impiegata per l'elaborazione dei messaggi per questo servizio, incluso il tempo trascorso in attesa di altri thread, in millisecondi.
ExecTimeOfDBInsertOnly	La quantità di tempo, in millisecondi, impiegata per l'elaborazione esclusivamente della porzione di inserimento batch.
HighWaterMark	Il numero massimo di messaggi elaborati per questo servizio.
NumberOfDBInserts	Il numero totale di inserimenti batch eseguiti.

Tabella 16. Statistiche Servizio (Continua)

Attributo	Descrizione
TotalRowsInserted	Il numero totale di righe inserite nel database.

Attributi Statistiche servizio - Programma di utilità per il caricamento del database

Gli attributi Statistiche servizio - Programma di utilità per il caricamento del database fanno parte dell'ambiente di runtime.

Tabella 17. Statistiche servizio - Programma di utilità per il caricamento del database

Attributo	Descrizione
ExecTimeOfWriteToCache	La quantità di tempo impiegata per la scrittura nella cache dei file, che include la scrittura nei file e l'acquisizione della chiave primaria dal database, quando necessario.
ExecTimeOfLoaderDBAccessOnly	La quantità di tempo, in millisecondi, impiegata per l'esecuzione esclusivamente della porzione del programma di caricamento del database.
ExecTimeOfLoaderThreads	La quantità di tempo, in millisecondi, impiegata dai thread del programma di caricamento del database.
ExecTimeOfFlushCacheFiles	La quantità di tempo, in millisecondi, impiegata per cancellare la cache e ricrearne di nuove.
ExecTimeOfRetrievePKDBAccess	La quantità di tempo, in millisecondi, impiegata per recuperare l'accesso al database della chiave primaria.
NumberOfDBLoaderRuns	Il numero totale di esecuzioni del programma di caricamento del database.
NumberOfLoaderStagingDirCreated	Il numero totale di directory di staging create.
NumberOfLoaderStagingDirRemoved	Il numero totale di directory di staging rimosse.
NumberOfLoaderStaging DirMovedToAttention	Il numero totale di directory di staging ridenominate per attenzione.
NumberOfLoaderStagingDirMovedToError	Il numero totale di directory di staging ridenominate per errore.
NumberOfLoaderStagingDirRecovered	Il numero totale di directory di staging ripristinate, anche in fase di avvio e rieseguite dai thread in background.
NumberOfTimesRetrievePKFromDB	Il numero totale di volte in cui la chiave primaria è stata richiamata dal database.
NumberOfLoaderThreadsRuns	Il numero totale di esecuzioni dei thread del programma di caricamento del database.
NumberOfFlushCacheFiles	Il numero totale di volte in cui il contenuto della cache dei file è stato cancellato.

Attributi Statistiche API

Gli attributi Statistiche API fanno parte dell'ambiente di runtime.

Tabella 18. Statistiche API

Attributo	Descrizione
endSessionCount	Il numero di chiamate API `endSession` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
endSessionDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `endSession`.
executeBatchCount	Il numero di chiamate API `executeBatch` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
executeBatchDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `executeBatch`.
getOffersCount	Il numero di chiamate API `getOffers` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
getOffersDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `getOffer`.
getProfileCount	Il numero di chiamate API `getProfile` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
getProfileDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `getProfileDuration`.
getVersionCount	Il numero di chiamate API `getVersion` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
getVersionDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `getVersion`.
loadOfferSuppressionDuration	Il tempo trascorso dall'ultima chiamata API `loadOfferSuppression`.
LoadOffersBySQLCount	Il numero di chiamate API `LoadOffersBySQL` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
LoadOffersBySQLDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `LoadOffersBySQL`.
loadProfileDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `loadProfile`.
loadScoreOverrideDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `loadScoreOverride`.
postEventCount	Il numero di chiamate API `postEvent` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
postEventDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `postEvent`.
runSegmentationDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `runSegmentation`.

Tabella 18. Statistiche API (Continua)

Attributo	Descrizione
setAudienceCount	Il numero di chiamate API `setAudience` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
setAudienceDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `setAudience`.
setDebugCount	Il numero di chiamate API `setDebug` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
setDebugDuration	Il tempo trascorso, in millisecondi, dall'ultima chiamata API `setDebug`.
startSessionCount	Il numero di chiamate API `startSession` dall'ultima reimpostazione o dall'ultimo avvio del sistema.
startSessionAverage	Il tempo medio trascorso, in millisecondi, dall'ultima chiamata API `startSession`.
ActiveSessionCount	Il numero di sessioni attualmente attive nell'istanza del runtime Interact. **Nota:** L'attributo ActiveSessionCount nell'MBean JMX com.unicacorp.interact:type=api, group=Statistics non tiene in considerazione gli eventi scaduti e, pertanto, potrebbe mostrare un conteggio attivo non corretto.

Attributi Statistiche optimizer di apprendimento

Gli attributi Statistiche optimizer di apprendimento fanno parte dell'ambiente di runtime.

Tabella 19. Statistiche optimizer di apprendimento

Attributo	Descrizione
LearningOptimizerAcceptCalls	Il numero di eventi accettazione che vengono trasmessi al modulo di apprendimento.
LearningOptimizer AcceptTrackingDuration	Il numero totale di millisecondi impiegati per registrare gli eventi accettazione nel modulo di apprendimento.
LearningOptimizerContactCalls	Il numero di eventi contatto che vengono trasmessi al modulo di apprendimento.
LearningOptimizer ContactTrackingDuration	Il numero totale di millisecondi impiegati per registrare gli eventi contatto nel modulo di apprendimento.
LearningOptimizerLogOtherCalls	Il numero di eventi non di contatto e non di accettazione che vengono trasmessi al modulo di apprendimento.
LearningOptimizer LogOtherTrackingDuration	La durata, in millisecondi, del tempo impiegato per registrare altri eventi (non di contatto e non di accettazione) nel modulo di apprendimento.

Tabella 19. Statistiche optimizer di apprendimento (Continua)

Attributo	Descrizione
LearningOptimizer NonRandomCalls	Il numero di volte in cui è stata applicata l'implementazione di apprendimento configurata.
LearningOptimizer RandomCalls	Il numero di volte in cui è stata ignorata l'implementazione di apprendimento configurata ed è stata applicata una selezione casuale.
LearningOptimizer RecommendCalls	Il numero di richieste di consiglio che vengono trasmesse al modulo di apprendimento.
LearningOptimizer RecommendDuration	Il numero totale di millisecondi impiegati nella logica del consiglio di apprendimento.

Attributi Statistiche offerte predefinite

Gli attributi Statistiche offerte predefinite fanno parte dell'ambiente di runtime.

Tabella 20. Statistiche offerte predefinite

Attributo	Descrizione
LoadDefaultOffersDuration	Il tempo trascorso nel caricamento delle offerte predefinite.
DefaultOffersCalls	Il numero di volte in cui si caricano offerte predefinite.

Attributi Dispatcher messaggi attivati

Gli attributi Dispatcher messaggi attivati fanno parte dell'ambiente di runtime.

Tabella 21. Dispatcher messaggi attivati

Attributo	Descrizione
NumRequested	Il numero totale di offerte di cui è stata richiesta la distribuzione tramite questo dispatcher.
NumDispatched	Il numero totale di offerte distribuite correttamente da questo dispatcher.
AvgExecutionTime	Il tempo medio in millisecondi che questo dispatcher utilizza per la distribuzione di un'offerta. Solo le offerte che sono state distribuite correttamente ai gateway vengono conteggiate nel calcolo.
CurrentQueueSize	Il numero di offerte attualmente in attesa di essere distribuite.
GatewayInvocation	Il numero di offerte e il tempo medio di distribuzione in millisecondi distribuite a ciascun gateway da questo dispatcher. Il formato del suo valore è `{nome gateway=[numero di offerte, tempo medio distribuzione]}`.

Attributi Gateway messaggi attivati

Gli attributi Gateway messaggi attivati fanno parte dell'ambiente di runtime.

Tabella 22. Gateway messaggi attivati

Attributo	Descrizione
NumValidationRequested	Il numero totale di offerte per cui questo gateway ha richiesto la convalida.
NumValidated	Il numero totale di offerte convalidate correttamente da questo gateway.
AvgValidationTime	Il tempo medio in millisecondi che questo gateway utilizza per la convalida di un'offerta. Solo le offerte che sono state convalidate correttamente vengono conteggiate nel calcolo.
NumDeliveryRequested	Il numero totale di offerte per cui questo gateway ha richiesto la consegna.
NumDelivered	Il numero totale di offerte consegnate correttamente da questo gateway.
AvgDeliveryTime	Il tempo medio in millisecondi che questo gateway utilizza per la consegna di un'offerta. Solo le offerte che sono state consegnate correttamente vengono conteggiate nel calcolo.

Attributi Messaggi messaggio attivato

Gli attributi Messaggi messaggio attivato fanno parte dell'ambiente di runtime.

Tabella 23. Messaggi messaggio attivato

Attributo	Descrizione
ProcessSuccessCount	Il numero totale di volte in cui questo messaggio attivato è stato eseguito correttamente.
AvgSuccessProcessTime	Il tempo medio in millisecondi impiegato da questo messaggio attivato per ogni esecuzione riuscita.
ProcessErrorCount	Il numero totale di volte in cui questo messaggio attivato è stato eseguito non correttamente.
AvgErrorProcessTime	Il tempo medio in millisecondi impiegato da questo messaggio attivato per ogni esecuzione non riuscita.
SelectBranchCount	Il numero totale di volte in cui è stata eseguita la selezione del ramo durante l'elaborazione dei messaggi attivati.

Tabella 23. Messaggi messaggio attivato (Continua)

Attributo	Descrizione
AvgSelectBranchTime	Il tempo medio in millisecondi che l'esecuzione di selezione ramo utilizza durante l'elaborazione dei messaggi attivati.
SelectOfferCount	Il numero totale di volte in cui è stata eseguita la selezione dell'offerta durante l'elaborazione dei messaggi attivati.
AvgSelectOfferTime	Il tempo medio in millisecondi che l'esecuzione di selezione offerta utilizza durante l'elaborazione dei messaggi attivati.
SelectChannelCount	Il numero totale di volte in cui è stata eseguita la selezione del canale durante l'elaborazione dei messaggi attivati.
AvgSelectChannelTime	Il tempo medio in millisecondi che l'esecuzione di selezione canale utilizza durante l'elaborazione dei messaggi attivati.
FlowchartWaitCount	Il numero totale di volte in cui questo messaggio attivato ha atteso il completamento della segmentazione.
AvgFlowchartWaitTime	Il tempo medio in millisecondi per cui questo messaggio attivato ha atteso il completamento della segmentazione in ciascuna esecuzione.
WaitFlowchartTimeoutCount	Il numero totale di volte in cui questo messaggio attivato è scaduto durante l'attesa del completamento della segmentazione.

Operazioni JMX

Sono disponibili varie operazioni per il monitoraggio JMX.

La seguente tabella descrive le operazioni disponibili per il monitoraggio JMX.

Gruppo	Attributo	Descrizione
Configurazione programma di registrazione	activateDebug	Impostare il livello di registrazione per il file di log definito in `Interact/conf/interact_log4j.properties` su debug.
Configurazione programma di registrazione	activateError	Impostare il livello di registrazione per il file di log definito in `Interact/conf/interact_log4j.properties` su error.
Configurazione programma di registrazione	activateFatal	Impostare il livello di registrazione per il file di log definito in `Interact/conf/interact_log4j.properties` su fatal.
Configurazione programma di registrazione	activateInfo	Impostare il livello di registrazione per il file di log definito in `Interact/conf/interact_log4j.properties` su info.

Gruppo	Attributo	Descrizione
Configurazione programma di registrazione	activateTrace	Impostare il livello di registrazione per il file di log definito in `Interact/conf/interact_log4j.properties` su trace.
Configurazione programma di registrazione	activateWarn	Impostare il livello di registrazione per il file di log definito in `Interact/conf/interact_log4j.properties` su warn.
Locale	changeLocale	Modificare la locale del client JMX. Le locale supportate da Interact sono `de`, `en`, `es` e `fr`.
ContactResponseHistory ETLMonitor	reset	Reimpostare tutti i contatori.
Statistiche offerte predefinite	updatePollPeriod	Aggiorna defaultOfferUpdatePollPeriod. Questo valore, espresso in secondi, indica al sistema la durata dell'attesa prima che il sistema aggiorni le offerte predefinite nella cache. Se questa proprietà è impostata su `-1`, il sistema legge il numero delle offerte predefinite solo all'avvio.

Capitolo 7. Classi e metodi per l'API Java, SOAP e REST di IBM Interact

Le seguenti sezioni elencano requisiti e altri dettagli che sarebbe necessario conoscere prima di utilizzare l'API di Interact.

Nota: in questa sezione si presuppone esperienza con il proprio touchpoint, il linguaggio di programmazione Java e lavorare l'utilizzo di un'API Java.

L'API di Interact è dotata di un adattatore client Java che utilizza la serializzazione Java su HTTP. Inoltre, Interact fornisce un WSDL a supporto dei client SOAP. Il WSDL espone la stessa serie di funzioni dell'adattatore client Java, quindi, le seguenti sezioni, fatta eccezione per gli esempi, sono ancora valide.

Nota: non sono supportate più ricorrenze di un qualunque parametro in una singola chiamata all'API.

Classi dell'API di Interact

L'API di Interact si basa sulla classe `InteractAPI`.

Esistono 6 interfacce di supporto.

- `AdvisoryMessage`
- `BatchResponse`
- `NameValuePair`
- `Offerta`
- `OfferList`
- `Risposta`

Queste interfacce hanno 3 classi concrete di supporto. È necessario creare un'istanza delle due classi concrete seguenti e devono essere trasmesse come argomenti nei metodi dell'API di Interact:

- NameValuePairImpl
- CommandImpl

Una terza classe concreta, denominata `AdvisoryMessageCode` è disponibile per fornire le costanti utilizzate per distinguere i codici messaggio restituiti dal server, ove possibile.

Il resto di questa sezione descrive i metodi che compongono l'API di Interact.

Prerequisiti della serializzazione Java su HTTP

L'adattatore client Java utilizza la serializzazione Java su HTTP.

I prerequisiti per l'utilizzo dell'adattatore client Java per la serializzazione Java su HTTP sono:

1. Aggiungere il seguente file a `CLASSPATH`:
   `Interact_Home/lib/interact_client.jar`

2. Tutti gli oggetti trasmessi bidirezionalmente tra client e server sono reperibili nel package `com.unicacorp.interact.api`. Per ulteriori dettagli, consultare il Javadoc dell'API di Interact installato sul server di runtime in `Interact_Home/docs/apiJavaDoc`. È possibile visualizzare il Javadoc, aprendo il file `index.html` in tale ubicazione con qualsiasi browser Web.
3. Per ottenere un'istanza della classe `InteractAPI`, richiamare il metodo statico `getInstance`, specificando l'url del server di runtime Interact.

Prerequisiti SOAP

Prima di poter accedere al server runtime con SOAP, si devono eseguire diverse attività prerequisite per configurare l'ambiente.

Importante: il test delle prestazioni mostra che l'adattatore di serializzazione Java ha prestazioni di livello molto più elevato rispetto ad un client SOAP generato. Per ottenere prestazioni ottimali, utilizzare l'adattatore di serializzazione Java ogniqualvolta possibile.

Per accedere al server di runtime con SOAP, è necessario effettuare le seguenti operazioni:

1. Convertire il WSDL dell'API di Interact con il toolkit SOAP di propria scelta.

   Il WSDL dell'API di Interact è installato con Interact nella directory `Interact/conf`.

   Quando si configura SOAP utilizzando file XML WSDL, è necessario modificare gli URL per il nome host e la porta del server di runtime.

   Il testo del WSDL è disponibile alla fine del manuale Interact - Guida dell'amministratore.
2. Installare e configurare il server di runtime.

   Il server runtime deve essere in esecuzione per un test completo dell'integrazione.
3. Verificare che si stia utilizzando la versione SOAP corretta.

   Interact utilizza axis2 1.3 come infrastruttura SOAP sui server di runtime Interact. Per dettagli sulle versioni di SOAP supportate da axis2 1.3, consultare il seguente sito Web:

   Apache Axis2

   Il test di Interact è stato eseguito con i client axis2, XFire, JAX-WS-Ri, DotNet, SOAPUI e IBM RAD SOAP.

Prerequisiti REST

Un metodo di chiamata dell'API di Interact consiste nell'utilizzare chiamate in formato JSON (JavaScript Object Notation) su HTTP, a cui si fa riferimento in questa sede come API REST. L'API REST presenta il vantaggio di avere prestazioni migliori rispetto a SOAP, anche se l'adattatore di serializzazione Java rappresenta ancora il metodo più veloce per le chiamate all'API di Interact.

Prima di iniziare ad utilizzare l'API REST, tenere presente quanto segue:

- L'URL che supporta chiamate REST all'API di Interact è:

  `http://Interact_Runtime_Server:PORT/interact/servlet/RestServlet`, con la sostituzione dell'effettivo nome host o indirizzo IP del server di runtime Interact e della porta su cui è distribuito Interact.
- Sono disponibili due classi Interact specifiche per l'API REST: `RestClientConnector`, che funziona come supporto per la connessione ad

un'istanza runtime Interact mediante REST con il formato JSON e `RestFieldConstants`, che descrive il formato sottostante del messaggio JSON utilizzato per le richieste API e le risposte.

- Un client REST di esempio viene fornito in `Interact _Home/samples/javaApi/InteractRestClient.java`. Anche se il codice di esempio è appunto un semplice esempio, dovrebbe fornire un buon punto di partenza per dimostrare la modalità di utilizzo dell'API REST.
- Per una descrizione completa delle classi API REST insieme a tutte le altre informazioni relativa all'API di Interact, consultare il Javadoc installato sul server di runtime in `Interact_Home/docs/apiJavaDoc`.
- L'API REST restituisce gli ID sessione e i messaggi nel formato escape HTML e non in formato Unicode.

Oltre alle informazioni qui menzionate, l'API REST supporta tutti i metodi che sono supportati dagli altri protocolli, per utilizzare l'API di Interact.

JavaDoc dell'API

Oltre alla guida dell'amministratore di Interact, con il server di runtime, viene installato il Javadoc per l'API Interact. Il Javadoc viene installato per riferimento nella directory `Interact_Home/docs/apiJavaDoc`.

Esempi di API

Tutti gli esempi nella guida sono stati creati con l'adattatore di serializzazione Java su HTTP. Le classi generate da WSDL possono variare in base al toolkit SOAP e alle opzioni selezionate. Se si sta utilizzando SOAP, questi esempi potrebbero non funzionare allo stesso modo nel proprio ambiente.

Gestione dei dati di sessione

Quando si avvia una sessione con il metodo `startSession`, i dati di sessione vengono caricati nella memoria. Nel corso di tutta la sessione, è possibile leggere e scrivere i dati di sessione (che rappresentano una super serie dei dati del profilo statici).

La sessione contiene i seguenti dati:

- Dati del profilo statici
- Assegnazioni del segmento
- Dati in tempo reale
- Raccomandazioni di offerte

Tutti i dati di sessione rimangono disponibili fino a quando non viene richiamato il metodo `endSession` o non scade il tempo specificato per `sessionTimeout`. Una volta terminata la sessione, tutti i dati non salvati esplicitamente nella cronologia dei contatti o delle risposta oppure in qualche altra tabella database, verranno persi.

I dati vengono memorizzati come una serie di coppie nome-valore. Se i dati vengono letti da una tabella del database, il nome corrisponde alla colonna della tabella.

È possibile creare queste coppie nome-valore mentre si utilizza l'API di Interact. Non è necessario dichiarare tutte le coppie nome-valore in un'area globale. Se si impostano nuovi parametri evento come coppie nome-valore, l'ambiente di runtime aggiunge le coppie nome-valore ai dati di sessione. Ad esempio, se si

utilizzano parametri evento con il metodo postEvent, l'ambiente di runtime aggiunge i parametri evento ai dati di sessione, anche se i parametri evento non erano disponibili nei dati del profilo. Questi dati esistono solo nei dati di sessione.

È possibile sovrascrivere i dati di sessione in qualsiasi momento. Ad esempio, se parte del profilo cliente include creditScore, è possibile trasmettere un parametro evento utilizzando il tipo personalizzato NameValuePair. Nella classe NameValuePair, è possibile utilizzare i metodi setName e setValueAsNumeric per modificare il valore. Il nome deve corrispondere. Nei dati di sessione, il nome non è sensibile al maiuscolo/minuscolo. Pertanto, ambedue i nomi creditscore o CrEdItScOrE sovrascriverebbero creditScore.

Solo gli ultimi dati scritti nei dati di sessione vengono conservati. Ad esempio, startSession carica i dati del profilo per il valore relativo a lastOffer. Un metodo postEvent sovrascrive lastOffer. Quindi, un secondo metodo postEvent sovrascrive lastOffer. L'ambiente di runtime conserva solo i dati scritti dal secondo metodo postEvent nei dati di sessione.

Quando una sessione termina, i dati vengono persi, a meno che non vengano effettuate considerazioni speciali, quali ad esempio utilizzare un processo Snapshot nel diagramma di flusso interattivo per la scrittura dei dati in una tabella del database. Si si sta pianificando l'utilizzo di processi Snapshot, tenere a mente che i nomi devono conformarsi alle limitazioni previste dal proprio database. Ad esempio, se è consentito un massimo di 256 caratteri per il nome di una colonna, allora il nome della coppia nome-valore non dovrebbe superare i 256 caratteri.

Informazioni sulla classe InteractAPI

La classe InteractAPI contiene i metodi che è possibile utilizzare per integrare il touchpoint con il server di runtime. Tutte le altre classi e metodi nell'API di Interact supportano i metodi contenuti in questa classe.

È necessario compilare la propria implementazione facendo riferimento al file interact_client.jar, che si trova nella directory lib della propria installazione dell'ambiente di runtime Interact.

endSession

Il metodo endSession contrassegna la fine della sessione di runtime. Quando il server di runtime riceve questo metodo, il server di runtime registra nella cronologia, ripulisce la memoria e così via.

```
endSession(String sessionID)
```

- **sessionID** - Stringa univoca che identifica la sessione.

Se il metodo endSession non viene richiamato, le sessioni di runtime vanno in timeout. Il periodo di timeout è configurabile con la proprietà sessionTimeout.

Valore restituito

Il server di runtime risponde al metodo endSession con l'oggetto Response con i seguenti attributi popolati:

- SessionID
- ApiVersion
- StatusCode
- AdvisoryMessages

Esempio

Il seguente esempio mostra il metodo endSession e come l'utente può analizzare la risposta. sessionId è la stessa stringa che identifica la sessione utilizzata dalla chiamata startSession che ha avviato questa sessione.

```
response = api.endSession(sessionId);
    // controllare l'esito della risposta
    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");
    }
    // Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazior
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("endSession",
   response.getAdvisoryMessages());
```

executeBatch

Il metodo executeBatch consente di eseguire diversi metodi con un'unica richiesta al server di runtime.

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

- **sessionID** - Una stringa che identifica l'ID sessione. Questo ID sessione viene utilizzato per tutti i comandi eseguiti da questa chiamata di metodo.
- **commandImpl[]** - un array di oggetti CommandImpl, uno per ogni comando che si desidera eseguire.

Il risultato della chiamata a questo metodo corrisponde a chiamare esplicitamente ogni metodo contenuto nell'array di comando. Questo metodo riduce al minimo il numero di richieste effettive al server di runtime. Il server di runtime esegue ogni metodo in serie; per ogni chiamata, vengono catturati gli eventuali errori o avvisi nell'oggetto risposta che corrisponde a quella chiamata di metodo. Se viene rilevato un errore, executeBatch continua con il resto delle chiamate contenute nel batch. Se l'esecuzione di un metodo ha come risultato un errore, lo stato di livello principale per l'oggetto BatchResponse riflette quell'errore. Se non si sono verificati errori, lo stato di livello principale riflette gli eventuali avvisi che possono essersi verificati. Se non si sono verificati avvisi, lo stato di livello principale riflette un'esecuzione corretta del batch.

Valore restituito

Il server di runtime risponde al executeBatch con un oggetto BatchResponse.

Esempio

Il seguente esempio mostra come chiamare tutti i metodi getOffer e postEvent con un'unica chiamata executeBatch, e un suggerimento su come gestire la risposta.

```
/** Definire tutte le variabili per tutti i membri di executeBatch*/
String sessionId="MySessionID-123";
String interactionPoint = "Overview Page Banner 1";
int numberRequested=1;
String eventName = "logOffer";
```

```
/** creare il comando getOffers */
Command getOffersCommand = new CommandImpl();
getOffersCommand.setMethodIdentifier(Command.COMMAND_GETOFFERS);
getOffersCommand.setInteractionPoint(interactionPoint);
getOffersCommand.setNumberRequested(numberRequested);

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

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

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

/** Elaborare la risposta in modo appropriato */
// Il codice di stato di livello principale è una scorciatoia per determinare se vi
// sono esiti negativi nell'array di oggetti risposta
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");
}

// Scorrere l'array, e stampare il messaggio per eventuali esiti negativi
for(Response response : batchResponse.getResponses())
{
  if(response.getStatusCode()!=Response.STATUS_SUCCESS)
  {
  printDetailMessageOfWarningOrError("executeBatchCommand",
 response.getAdvisoryMessages());
  }
}
```

Scrittura di richieste XML executeBatch() per l'API SOAP Interact

Utilizzare questa procedura per scrivere richieste XML `executeBatch()` per l'API SOAP Interact.

Informazioni su questa attività

Le chiamate all'API SOAP di richiesta XML per una singola operazione (`startSession`, `getOffers`, `setAudience`, `endSession`, ecc) non devono essere direttamente copiate o incollate in una chiamata `executeBatch()` per operazioni multiple. I sottocomandi nelle chiamate `executeBatch()` hanno strutture di richiesta XML e WSDL leggermente diverse da quelle delle chiamate API per operazione singola. Le differenze strutturali causano errori nelle risposte dal server se gli elementi XML vengono copiati e incollati da richieste API per operazione singola in richieste `executeBatch` per operazioni multiple.

Esempi di errori nelle risposte:

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

Utilizzare questi step per scrivere una richiesta XML `executeBatch()`. È possibile fare riferimento alle richieste di chiamata API operazione singola per i valori dei parametri durante questa procedura, ma non copiare e incollare gli elementi XML.

Procedura

1. Utilizzare uno strumento di elaborazione WSDL (ad esempio, SoapUI) per creare una richiesta XML `executeBatch()` in formato corretto dal file WSDL Interact.
2. Aggiungere i sottocomandi alla richiesta dopo la definizione WSDL per gli elementi figlio `executeBatch()`.
3. Completare gli argomenti dei sottocomandi dopo la definizione WSDL per gli elementi figlio `executeBatch()`.

getInstance

Il metodo `getInstance` crea un'istanza deò'API Interact che comunica con il server di runtime specificato.

```
getInstance(String URL)
```

Importante: ogni applicazione scritta utilizzando l'API Interact deve richiamare `getInstance` per creare un'istanza dell'oggetto InteractAPI che viene associato a un server di runtime specificato dal parametro URL.

Per i gruppi di server, se si utilizza un bilanciamento del carico, utilizzare il nome host e la porta configurati con il bilanciamento del carico. Se non si dispone di un bilanciamento del carico, sarà necessario includere la logica per effettuare la rotazione tra i server di runtime disponibili.

Questo metodo è applicabile solo per la serializzazione Java su adattatore HTTP. Non esiste un metodo corrispondente definito nel SOAP WSDL. Ogni implementazione di client SOAP ha un proprio modo di stabilire l'URL di endpoint.

- **URL** - Una stringa che identifica l'URL per l'istanza di runtime. Ad esempio, `http://localhost:7001/Interact/servlet/InteractJSService`.

Valore restituito

Il server di runtime restituisce l'InteractAPI.

Esempio

Il seguente esempio mostra come creare un'istanza dell'oggetto InteractAPI che punti ad un'istanza di server di runtime sulla stessa macchina del touchpoint.

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

getOffers

Il metodo getOffers consente di richiedere offerte dal server di runtime.

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

- **sessionID** - una stringa che identifica la sessione corrente.
- **interactionPoint** - una stringa che identifica il nome del punto di interazione a cui questo metodo fa riferimento.

  Nota: questo nome deve corrispondere esattamente al nome del punto di interazione definito nel canale interattivo.
- **numberOfOffers** - un numero intero che identifica il numero di offerte richieste.

Il metodo getOffers attende per il numero di millisecondi definito nella proprietà segmentationMaxWaitTimeInMS che siano completate tutte le nuove segmentazioni prima di passare all'esecuzione. Pertanto, se si chiama un metodo postEvent che attiva una nuova segmentazione o un metodo setAudience immediatamente prima di una chiamata getOffers, potrebbe verificarsi un ritardo.

Valore restituito

Il server di runtime risponde a getOffers con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

Esempio

Questo esempio mostra la richiesta di una singola offerta per il punto di interazione Overview Page Banner 1 e un modo per gestire la risposta.

sessionId è la stessa stringa che identifica la sessione di runtime utilizzata dalla chiamata startSession che ha avviato questa sessione.

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

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

/** Elaborare la risposta in modo appropriato */
 // controllare l'esito della risposta
 if(response.getStatusCode() == Response.STATUS_SUCCESS)
 {
  System.out.println("getOffers call processed with no warnings or errors");

        /** Verificare se sono presenti eventuali offerte */
        OfferList offerList=response.getOfferList();

        if(offerList.getRecommendedOffers() != null)
        {
            for(Offer offer : offerList.getRecommendedOffers())
            {
                // stampare l'offerta
                System.out.println("Offer Name:"+offer.getOfferName());
            }
        }
```

```
        else // count on the default Offer String
        System.out.println("Default offer:"+offerList.getDefaultString());
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getOffers call processed with a warning");
    }
    else
    {
        System.out.println("getOffers call processed with an error");
    }
    // Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazior
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getOffers",
   response.getAdvisoryMessages());
```

getOffersForMultipleInteractionPoints

Il metodo `getOffersForMultipleInteractionPoints` consente di richiedere offerte dal server di runtime per più IP con la deduplicazione.

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

- **sessionID** - una stringa che identifica la sessione corrente.
- **requestStr** - una stringa che fornisce un array di oggetti `GetOfferRequest`.

  Ogni ogetto `GetOfferRequest` specifica:

  – **ipName** - Il nome del punto di interazione (IP) per il quale l'oggetto sta richiedendo le offerte

  – **numberRequested** - Il numero di offerte univoche di cui necessita per l'IP specificato

  – **offerAttributes** - Requisiti sugli attributi delle offerte fornite utilizzando un'istanza di `OfferAttributeRequirements`

  – **duplicationPolicy** - L'ID della politica di duplicazione per le offerte che verranno fornite

    Le politiche di duplicazione determinano se le offerte duplicate verranno restituite nei differenti punti di interazione in una singola chiamata al metodo. (*All'interno* di un singolo punto di interazione, non vengono mai restituite offerte duplicate). Attualmente, sono supportate due politiche di duplicazione.

    - NO_DUPLICATION (valore ID = 1). Nessuna delle offerte incluse nelle istanze `GetOfferRequest` precedenti verrà inclusa in questa istanza `GetOfferRequest` (ossia, Interact applicherà la deduplicazione).
    - ALLOW_DUPLICATION (valore ID = 2). Le offerte che soddisfano i requisiti specificati in questa istanza di `GetOfferRequest` verranno incluse. Le offerte incluse nelle istanze `GetOfferRequest` precedenti non verranno riconciliate.

  L'ordine delle richieste nel parametro array è anche l'ordine di priorità quando vengono fornite le offerte.

  Ad esempio, si supponga che gli IP nella richiesta siano IP1, quindi IP2, che non siano consentite le offerte duplicate (un ID politica di duplicazione = 1), e ognuno richieda due offerte. Se Interact individua le offerte A, B e C per IP1 e le offerte A e D per IP2, la risposta conterrà le offerte A e B per IP1, e solo l'offerta D per IP2.

  Notare inoltre che quando l'ID politica di duplicazione è 1, le offerte che sono state fornite tramite un IP con una priorità più elevata non verranno fornite tramite questo IP.

Il metodo getOffersForMultipleInteractionPoints attende per il numero di millisecondi definito nella proprietà segmentationMaxWaitTimeInMS che siano completate tutte le nuove segmentazioni prima di passare all'esecuzione. Pertanto, se si chiama un metodo postEvent che attiva una nuova segmentazione o un metodo setAudience immediatamente prima di una chiamata getOffers, potrebbe verificarsi un ritardo.

Valore restituito

Il server di runtime risponde a getOffersForMultipleInteractionPoints con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- array di OfferList
- SessionID
- StatusCode

Esempio

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

Notare che la sintassi di requestStr è la seguente:

```
requests_for_IP[<requests_for_IP]
```

dove

```
<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
```

Nell'esempio mostrato in precedenza, `requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))})` significa, per il punto di interazione denominato IP1, la consegna al massimo di 5 offerte distinte che non possono anche essere restituite per altri punti di interazione durante questa stessa chiamata di metodo. Tutte e 5 le offerte devono avere un attributo numerico denominato `attr1` che deve avere il valore 1, e deve avere un attributo stringa denominato `attr2` che deve avere il valore *value2*. Di quelle 5 offerte, un massimo di tre deve avere un attributo stringa denominato `attr3` con valore *value3*, e un massimo di 3 deve avere un attributo numerico denominato `attr4` con valore *4*.

I tipi di attributo consentiti sono numerico, stringa e data/ora, e il formato di un attributo data/ora deve essere `MM/dd/yyyy HH:mm:ss`. Per recuperare le offerte restituite, utilizzare il metodo `Response.getAllOfferLists()`. Per facilitare la comprensione della sintassi, l'esempio in `setGetOfferRequests` crea la stessa istanza di `GetOfferRequests`, pur utilizzando oggetti Java, che è preferibile.

getProfile

Il metodo `getProfile` consente di recuperare le informazioni temporali e sul profilo relative al visitatore che visita il touchpoint.

`getProfile(String *sessionID*)`

- **sessionID** - una stringa che identifica l'ID sessione.

Valore restituito

Il server di runtime risponde a `getProfile` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

Esempio

Quello che segue è un esempio di utilizzo di `getProfile` e un modo di gestire la risposta.

`sessionId` è la stessa stringa che identifica la sessione utilizzata dalla chiamata `startSession` che ha avviato questa sessione.

```
response = api.getProfile(sessionId);
/** Elaborare la risposta in modo appropriato */
    // controllare l'esito della risposta
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("getProfile call processed with no warnings or errors");
        // Stampare il profilo - è semplicemente un array di oggetti NameValuePair
        for(NameValuePair nvp : response.getProfileRecord())
        {
            System.out.println("Name:"+nvp.getName());
            if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
            {
                System.out.println("Value:"+nvp.getValueAsDate());
            }
            else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
            {
```

```
                System.out.println("Value:"+nvp.getValueAsNumeric());
            }
            else
            {
                System.out.println("Value:"+nvp.getValueAsString());
            }
        }
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getProfile call processed with a warning");
    }
    else
    {
        System.out.println("getProfile call processed with an error");
    }
    // Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazione
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getProfile",
   response.getAdvisoryMessages());
```

getVersion

Il metodo `getVersion` restituisce la versione dell'implementazione corrente del server di runtime Interact.

```
getVersion()
```

La procedura ottimale è di utilizzare questo metodo quando si inizializza il touchpoint con l'API Interact.

Valore restituito

Il server di runtime risponde a `getVersion` con un oggetto risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- StatusCode

Esempio

Questo esempio mostra un metodo semplice per richiamare `getVersion` ed elaborare i risultati.

```
response = api.getVersion();
/** Elaborare la risposta in modo appropriato */
    // controllare l'esito della risposta
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("getVersion call processed with no warnings or errors");
        System.out.println("API Version:" + response.getApiVersion(););
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getVersion call processed with a warning");
    }
    else
    {
        System.out.println("getVersion call processed with an error");
    }
```

```
// Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazior
if(response.getStatusCode() != Response.STATUS_SUCCESS)
    printDetailMessageOfWarningOrError("getVersion",
response.getAdvisoryMessages());
```

postEvent

Il metodo postEvent consente di eseguire qualsiasi evento definito nel canale interattivo.

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

- **sessionID**: una string che identifica l'ID sessione.
- **eventName**: una stringa che identifica il nome dell'evento.

  Nota: il nome dell'evento deve corrispondere al nome dell'evento definito nel canale interattivo. Questo nome non è sensibile al maiuscolo/minuscolo.
- **eventParameters**. Oggetti NameValuePairImpl che identificano qualsiasi parametro debba essere trasmesso con l'evento. Questi valori sono memorizzati nei dati di sessione.

  Se questo evento attiva la risegmentazione, è necessario assicurarsi che tutti i dati richiesti dai diagrammi di flusso interattivi siano disponibili nei dati della sessione. Se qualcuno di questi valori non è stato popolato da azioni precedenti (ad esempio, da startSession o setAudience, oppure caricando la tabella profili) è necessario includere un eventParameter per ogni valore mancante. Ad esempio, se tutte le tabelle profili sono state configurate per essere caricate in memoria, è necessario includere un NameValuePair per i dati temporali richiesti per i diagrammi di flusso interattivi.

  Se si sta utilizzando più di un livello destinatario, è molto probabile che si disponga di serie di eventParameters differenti per ciascun livello destinatario. È necessario includere della logica per assicurarsi di selezionare la serie corretta di parametri per il livello destinatario.

  Importante: se questo evento viene registrato nella cronologia delle risposte, è necessario passare il codice trattamento per l'offerta. È necessario definire il nome per NameValuePair "UACIOfferTrackingCode".

  È possibile passare un solo codice trattamento per evento. Se non si passa il codice trattamento per un contatto di offerta, Interact registra un contatto di offerta per ogni offerta presente nell'ultimo elenco di offerte consigliate. Se non si passa il codice trattamento per una risposta, Interact restituisce un errore.
- Sono disponibili diversi altri parametri riservati utilizzati con postEvent ed altri metodi e verranno trattati in seguito in questa sezione.

Qualsiasi richiesta di risegmentazione o scrittura nella cronologia dei contatti o delle risposte non attende una risposta.

La risegmentazione non cancella i precedenti risultati di segmentazione per il livello destinatario corrente. È possibile utilizzare il parametro UACIExecuteFlowchartByName per definire diagrammi di flusso specifici da eseguire. Il metodo getOffers attende il termine della risegmentazione prima dell'esecuzione. Pertanto, se si richiama un metodo postEvent, ciò attiva una risegmentazione immediatamente prima di una chiamata getOffers, potrebbe verificarsi un ritardo.

Valore restituito

Il server di runtime risponde a `postEvent` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Esempio

Il seguente esempio di `postEvent` mostra l'invio di nuovi parametri per un evento che attiva la risegmentazione e un modo per gestire la risposta.

`sessionId` è la stessa stringa che identifica la sessione utilizzata dalla chiamata `startSession` che ha avviato questa sessione.

```
String eventName = "SearchExecution";

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

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

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

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

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

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

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

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

/** Elaborare la risposta in modo appropriato */
    // controllare l'esito della risposta
    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");
}

// Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazior
if(response.getStatusCode() != Response.STATUS_SUCCESS)
    printDetailMessageOfWarningOrError("postEvent",
response.getAdvisoryMessages());
```

setAudience

Il metodo `setAudience` consente di impostare ID e livello destinatario di un visitatore.

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

- **sessionID** - una stringa che identifica l'ID sessione.
- **audienceID** - un array di oggetti `NameValuePairImpl` che definisce l'ID destinatario.
- **audienceLevel** - una stringa che definisce il livello destinatario.
- **parameters** - oggetti `NameValuePairImpl` che identificano i parametri che devono essere trasmessi con `setAudience`. Questi valori vengono memorizzati nei dati di sessione e possono essere utilizzati per la segmentazione.

  È necessario avere un valore per ogni colonna nel profilo. Questo è un superinsieme di tutte le colonne in tutte le tabelle definite per il canale interattivo e qualsiasi dato in tempo reale. Se tutti i dati di sessione sono stati già popolati con `startSession` o `postEvent`, non è necessario inviare nuovi parametri.

Il metodo `setAudience` attiva una risegmentazione. Il metodo `getOffers` attende il termine della risegmentazione prima dell'esecuzione. Pertanto, se si richiama un metodo `setAudience` immediatamente prima di una chiamata `getOffers`, potrebbe esserci un ritardo.

Il metodo `setAudience` inoltre carica i dati del profilo per l'ID destinatario. È possibile utilizzare il metodo `setAudience` per forzare un ricaricamento degli stessi dati profilo caricati dal metodo `startSession`.

Valore restituito

Il server di runtime risponde a `setAudience` con un oggetto risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Esempio

Per questo esempio, il livello destinatario resta uguale ma l'ID cambia, come se un utente anonimo effettuasse l'accesso e divenisse noto.

`sessionId` e `audienceLevel` sono le stesse stringhe per identificare la sessione e il livello destinatario utilizzati dalla chiamata `startSession` che ha avviato questa sessione.

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

NameValuePair[] newAudienceId = { custId2 };

/** Possono essere trasmessi anche i parametri.  Per questo esempio, non ci sono parametri,
* pertanto la trasmissione è null */
NameValuePair[] noParameters=null;

/** Effettuare una chiamata */
response = api.setAudience(sessionId, newAudienceId, audienceLevel, noParameters);

 /** Elaborare la risposta in modo appropriato */
    // controllare l'esito della risposta
    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");
    }

    // Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazione del motivo
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setAudience",
   response.getAdvisoryMessages());
```

setDebug

Il metodo `setDebug` consente di impostare il livello di dettaglio della registrazione per tutti i percorsi di codice della sessione.

`setDebug(String sessionID, boolean debug)`

- **sessionID** - una stringa che identifica l'ID sessione.
- **debug** - un valore booleano che abilita o disabilita le informazioni di debug. I valori validi sono `true` o `false`. Se è true, Interact registra le informazioni di debug nel log del server di runtime.

Valore restituito

Il server di runtime risponde a `setDebug` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

Esempio

Il seguente esempio mostra la modifica del livello di debug della sessione.

sessionId è la stessa stringa che identifica la sessione utilizzata dalla chiamata startSession che ha avviato questa sessione.

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

/** Elaborare la risposta in modo appropriato */
    // controllare l'esito della risposta
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("setDebug call processed with no warnings or errors");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("setDebug call processed with a warning");
    }
    else
    {
        System.out.println("setDebug call processed with an error");
    }

    // Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazior
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setDebug",
   response.getAdvisoryMessages());
```

startSession

Il metodo startSession crea e definisce una sessione di runtime.

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

startSession può attivare fino a cinque azioni:

- Creare una sessione di runtime.
- Caricare i dati del profilo del visitatore per il livello destinatario corrente nella sessione di runtime, incluse le eventuali tabelle dimensionali contrassegnate per il caricamento nel mapping delle tabelle definito per il canale interattivo.
- Attivare la segmentazione, eseguendo tutti i diagrammi di flusso interattivi per il livello destinatario corrente.
- Caricare i dati di soppressione dell'offerta nella sessione, se la proprietà enableOfferSuppressionLookup è impostata su true.
- Caricare i dati di sovrascrittura dei punteggi nella sessione, se la proprietà enableScoreOverrideLookup è impostata su true.

Il metodo startSession richiede i seguenti parametri:

- **sessionID** - una stringa che identifica l'ID sessione. È necessario definire l'ID sessione. Ad esempio, si potrebbe utilizzare una combinazione di ID cliente e data/ora.

  Per definire cosa costituisce una sessione di runtime, è necessario specificare un id sessione. Questo valore è gestito dal client. Tutte le chiamate di metodo per lo stesso id sessione devono essere sincronizzate dal client. Il comportamento delle chiamate API simultanee con lo stesso id sessione non è definito.
- **relyOnExistingSession** - un valore booleano che definisce se questa sessione utilizza una nuova sessione o una esistente. I valori validi sono true o false. Se

è true, è necessario fornire un ID sessione esistente con il metodo startSession. Se è false, è necessario fornire un nuovo ID sessione.

Se si imposta relyOnExistingSession su true ed esiste una sessione, l'ambiente di runtime utilizza i dati della sessione esistente e non ricarica alcun dato o attiva la segmentazione. Se la sessione non esiste, l'ambiente di runtime crea una nuova sessione, incluso il caricamento dei dati e l'attivazione della segmentazione. L'impostazione di relyOnExistingSession su true e l'utilizzo con tutte le chiamate startSession è utile se il touchpoint ha una durata di sessione maggiore rispetto alla sessione di runtime. Ad esempio, una sessione del sito web è attiva per 2 ore, ma la sessione di runtime è attiva solo per 20 minuti.

Se si richiama startSession due volte con lo stesso ID sessione, tutti i dati di sessione della prima chiamata startSession vengono persi se relyOnExistingSession è false.

- **debug** - un valore booleano che abilita o disabilita le informazioni di debug. I valori validi sono true o false. Se è true, Interact registra le informazioni di debug nei log del server di runtime. L'indicatore di debug viene impostato singolarmente per ogni sessione. Pertanto è possibile tenere traccia dei dati di debug per una singola sessione.
- **interactiveChannel** - una stringa che definisce il nome del canale interattivo a cui questa sessione fa riferimento. Questo nome deve corrispondere esattamente al nome del canale interattivo definito in Campaign.
- **audienceID** - un array di oggetti NameValuePairImpl in cui i nomi devono corrispondere ai nomi colonna fisica di ogni tabella che contiene l'ID destinatario.
- **audienceLevel** - una stringa che definisce il livello destinatario.
- **parameters** - Oggetti NameValuePairImpl che identificano qualsiasi parametro debba essere trasmesso con startSession. Questi valori vengono memorizzati nei dati di sessione e possono essere utilizzati per la segmentazione.

  Se si dispone di diversi diagrammi di flusso interattivi per lo stesso livello destinatario, è necessario includere una superinsieme di tutte le colonne in tutte le tabelle. Se si configura il runtime per il caricamento della tabella profili, e la tabella profili contiene tutte le colonne richieste, non è necessario trasmettere i parametri, a meno che non di desideri sovrascrivere i dati nella tabella profili. Se la tabella profili contiene un sottoinsieme delle colonne richieste, è necessario includere le colonne mancanti come parametri.

Se audienceID o audienceLevel non sono validi e relyOnExistingSession è false, la chiamata startSession ha esito negativo. Se interactiveChannel non è valido, startSession ha esito negativo, a prescindere dal fatto che relyOnExistingSession sia true o false.

Se relyOnExistingSession è true, e si effettua una seconda chiamata startSession utilizzando lo stesso sessionID, ma la prima sessione è scaduta, Interact crea una nuova sessione.

Se relyOnExistingSession è true, e si effettua una seconda chiamata startSession utilizzando lo stesso sessionID ma un audienceID o audienceLevel differente, il server di runtime modifica il destinatario per la sessione esistente.

Se relyOnExistingSession è true, e si effettua una seconda chiamata startSession utilizzando lo stesso sessionID ma un interactiveChannel diverso, il server di runtime crea una nuova sessione.

Valore restituito

Il server di runtime risponde a `startSession` con un oggetto risposta con i seguenti attributi popolati:

- AdvisoryMessages (se StatusCode non è uguale a 0)
- ApiVersion
- SessionID
- StatusCode

Esempio

Il seguente esempio mostra un modo per richiamare `startSession`.

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

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

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

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

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

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

/** Specificare i parametri (facoltativo) */
NameValuePair[] initialParameters = { parm1,
 parm2,
 parm3,
 parm4,
 parm5,
 parm6
 };

/** Make the call */
response = api.startSession(sessionId, relyOnExistingSession, initialDebugFlag,
```

```
 interactiveChannel, initialAudienceId, audienceLevel, initialParameters);

/** Elaborare la risposta in modo appropriato */
processStartSessionResponse(response);
```

processStartSessionResponse è un metodo che gestisce l'oggetto risposta restituito da startSession.

```
public static void processStartSessionResponse(Response response)
{
  // controllare l'esito della risposta
  if(response.getStatusCode() == Response.STATUS_SUCCESS)
 {
  System.out.println("startSession call processed with no warnings or errors");
 }
  else if(response.getStatusCode() == Response.STATUS_WARNING)
 {
  System.out.println("startSession call processed with a warning");
 }
  else
 {
  System.out.println("startSession call processed with an error");
 }

 // Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazione del
 if(response.getStatusCode() != Response.STATUS_SUCCESS)
  printDetailMessageOfWarningOrError("StartSession",
   response.getAdvisoryMessages());
}
```

Attributi deduplicazione offerta nelle offerte

Utilizzando l'API (application programming interface) Interact, due chiamate API forniscono le offerte: getOffers e getOffersForMultipleInteractionPoints. getOffersForMultipleInteractionPoints può impedire la restituzione di offerte duplicate al livello *OfferID*, ma non offerte deduplicate nella categoria di offerte. Quindi, ad esempio, perché Interact possa restituire una sola offerta da ogni categoria di offerta, in precedenza era stata richiesta una soluzione temporanea. Con l'introduzione di due parametri alla chiamata API startSession, la deduplicazione delle offerte negli attributi offerta, come la categoria, ora è possibile.

Questo elenco riepiloga i parametri che sono stati aggiunti alla chiamata API startSession. Per ulteriori informazioni su questi parametri o qualsiasi aspetto dell'API Interact, consultare *IBM Interact - Guida dell'amministratore*, o i file Javadoc inclusi con l'installazione di Interact in <Interact_Home>/docs/apiJavaDoc.

- UACIOfferDedupeAttribute. Per creare una chiamata API startSession con la deduplicazione delle offerte, in modo che le chiamate getOffer successive restituiscano solo un'offerta da ciascuna categoria, è necessario includere il parametro UACIOfferDedupeAttribute come parte della chiamata API. È possibile specificare un parametro nel formato name,value,type, come mostrato qui:

  UACIOfferDedupeAttribute,<attributeName>,string

  In questo esempio, <*attributeName*> verrebbe sostituito con il nome dell'attributo offerta che si desidera utilizzare come criterio per la deduplicazione, ad esempio Categoria.

  Nota: Interact esamina le offerte che hanno lo stesso valore attributo che l'utente specifica (ad esempio Categoria) e deduplica per rimuovere tutte le offerte

tranne quella che ha il punteggio più alto. Se le offerte che hanno l'attributo di deduplicazione hanno anche punteggi identici, Interact restituisce una selezione casuale tra le offerte corrispondenti.

- `UACINoAttributeDedupeIfFewerOf`. Quando si include `UACIOfferDedupeAttribute` nella chiamata `startSession`, è possibile impostare anche questo parametro `UACINoAttributeDedupeIfFewerOf` per specificare il comportamento nei casi in cui l'elenco di offerte dopo la deduplicazione non contenga più offerte sufficienti a soddisfare la richiesta originale.

  Ad esempio, se si imposta `UACIOfferDedupeAttribute` per utilizzare la categoria di offerta per deduplicare le offerte, e la successiva chiamata getOffers richiede che vengano restituite otto offerte, la deduplicazione potrebbe avere come risultato un numero di offerte idonee inferiore a otto. In quel caso, impostando il parametro `UACINoAttributeDedupeIfFewerOf` su true comporterebbe l'aggiunta di alcune delle offerte duplicate all'elenco delle offerte idonee per soddisfare il numero di offerte richiesto. In questo esempio, se si imposta il parametro su false, il numero di offerte restituite sarebbe inferiore al numero richiesto.

  `UACINoAttributeDedupeIfFewerOf` è impostato su true per impostazione predefinita.

Ad esempio, si supponga di aver specificato come parametro `startSession` che il criterio di deduplicazione sia la categoria offerta, come mostrato qui:

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

A causa di questi parametri insieme Interact deduplica le offerte in base all'attributo offerta "Category," e restituisce solo le offerte deduplicate anche se il numero risultante delle offerte è inferiore a quello richiesto (`UACINoAttributeDedupeIfFewerOffer` è false).

Quando si invia una chiamata API `getOffers`, la serie originale di offerte idonee potrebbe includere queste offerte:

- Category=Electronics: Offerta A1 con un punteggio di 100 e Offerta A2 con un punteggio di 50.
- Category=Smartphones: Offerta B1 con un punteggio di 100, Offerta B2 con un punteggio di 80 e Offerta B3 con un punteggio di 50.
- Category=MP3Players: Offerta C1 con un punteggio di 100, Offerta C2 con un punteggio di 50.

In questo caso, ci sono due offerte duplicate che corrispondono alla prima categoria, tre offerte duplicate che corrispondono alla seconda categoria, e due offerte duplicate che corrispondono alla terza categoria. Le offerte restituite sarebbero le offerte con il punteggio più alto di ciascuna categoria, ovvero Offerta A1, Offerta B1 e Offerta C1.

Se la chiamata API `getOffers` ha richiesto sei offerte, questo esempio imposta `UACINoAttributeDedupeIfFewerOffer` su false, in modo da restituire solo tre offerte.

Se la chiamata API `getOffers` ha richiesto sei offerte, e questo esempio ha omesso il parametro `UACINoAttributeDedupeIfFewerOffer`, oppure è stato impostato specificatamente su true, verranno incluse offerte duplicate nel risultato per soddisfare il numero richiesto.

Parametri riservati

Vi sono diversi parametri riservati utilizzati con l'API di Interact. Alcuni sono richiesti per il server di runtime e altri possono essere utilizzati per ulteriori funzioni.

Funzioni postEvent

Funzione	Parametro	Descrizione
Registrazione nella tabella personalizzata	UACICustomLoggerTableName	Il nome di una tabella nell'origine dati delle tabelle di runtime. Se si fornisce questo parametro con un nome tabella valido, l'ambiente di runtime scrive tutti i dati di sessione nella tabella selezionata. Tutti i nomi colonna nella tabella che corrispondono ai dati di sessione NameValuePair vengono popolati. L'ambiente di runtime popola con un null qualsiasi colonna che non corrisponde ad una coppia nome-valore della sessione. È possibile gestire il processo che scrive nel database con le proprietà di configurazione customLogger.
Tipi di risposta multipli	UACILogToLearning	Un numero intero con valore 1 o 0. 1 indica che l'ambiente di runtime dovrebbe registrare l'evento come accettazione per il sistema di apprendimento o abilitare la soppressione dell'offerta in una sessione. 0 indica che l'ambiente di runtime non dovrebbe registrare l'evento per il sistema di apprendimento o abilitare la soppressione dell'offerta in una sessione. Questo parametro consente di creare vari metodi postEvent, che registrano differenti tipi di risposta, senza influire sull'apprendimento. Non è necessario definire questo parametro per gli eventi impostati sulla registrazione di un contatto, un'accettazione o un rifiuto. È necessario utilizzare questo parametro insieme a UACIResponseTypeCode. Se non si definisce UACILOGTOLEARNING, l'ambiente di runtime presume il valore predefinito 0 (a meno che l'evento non attivi una registrazione contatto, accettazione o rifiuto).
	UACIResponseTypeCode	Un valore che rappresenta un codice di tipo di risposta. Il valore deve essere una voce valida nella tabella UA_UsrResponseType

Funzione	Parametro	Descrizione
Tracciamento delle risposte	UACIOfferTrackingCode	Il codice trattamento per l'offerta. È necessario definire questo parametro se l'evento viene registrato nella cronologia dei contatti o delle risposte. È possibile passare un solo codice trattamento per evento. Se non si trasmette il codice trattamento per un contatto di offerta, l'ambiente di runtime registra un contatto di offerta per ogni offerta nell'ultimo elenco di offerte consigliate. Se non si passa il codice trattamento per una risposta, l'ambiente di runtime restituisce un errore. Se si configura il tracciamento della risposta delle sessioni incrociate, è possibile utilizzare il parametro UACIOfferTrackingcodeType per definire quale tipo di codice di tracciamento utilizzare invece del codice trattamento.
Tracciamento della risposta delle sessioni incrociate	UACIOfferTrackingCodeType	Un numero che definisce il tipo di codice di tracciamento. 1 è il codice trattamento predefinito e 2 è il codice offerta. Tutti i codici devono essere voci valide nella tabella UACI_TrackingType. È possibile aggiungere altri codici personalizzati a questa tabella.
Esecuzione di un diagramma di flusso specifico	UACIExecuteFlowchartByName	Se si definisce questo parametro per un metodo che attiva la segmentazione (startSession, setAudience o un postEvent che attiva la risegmentazione), invece di eseguire tutti i diagrammi di flusso per il livello destinatario corrente, Interact esegue solo i diagrammi di flusso indicati. È possibile fornire un elenco di diagrammi di flusso separati da un carattere barra verticale (|).

Parametri riservati per l'ambiente di runtime

I seguenti parametri riservati sono utilizzati dall'ambiente di runtime. Non utilizzare questi nomi per i parametri evento.

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

Informazioni sulla classe AdvisoryMessage

La classe advisoryMessage contiene metodi che definiscono l'oggetto messaggio di avviso. L'oggetto messaggio di avviso è contenuto nell'oggetto risposta. Ogni metodo nell'InteractAPI restituisce un oggetto risposta. (Fatta eccezione per il metodo executeBatch, che restituisce un oggetto batchResponse.)

Se si genera un errore o un'avvertenza, il server Interact popola l'oggetto messaggio di avviso. L'oggetto messaggio di avviso contiene i seguenti attributi:

- **DetailMessage**-a descrizione dettagliata del messaggio di avviso. Questo attributo potrebbe non essere disponibile per tutti i messaggi di avviso. Se disponibile, è possibile che DetailMessage non contenga una descrizione localizzata.
- **Message**-a breve descrizione del messaggio di avviso.
- **MessageCode**-a numero di codice per il messaggio di avviso.
- **StatusLevel**-a numero di codice relativo alla gravità del messaggio di avviso.

Si richiamano gli oggetti advisoryMessage utilizzando il metodo `getAdvisoryMessages`.

getDetailMessage

Il metodo `getDetailMessage` restituisce la descrizione dettagliata di un oggetto messaggio di avviso. Non tutti i messaggi hanno un messaggio dettagliato.

```
getDetailMessage()
```

Valore restituito

L'oggetto messaggio di avviso restituisce una stringa.

Esempio

```
// Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazione del r
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
 for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
  System.out.println(msg.getMessage());
  // Alcuni messaggi di avviso possono avere ulteriori dettagli:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessage

Il metodo `getMessage` restituisce una breve descrizione di un oggetto messaggio di avviso.

```
getMessage()
```

Valore restituito

L'oggetto messaggio di avviso restituisce una stringa.

Esempio

Il seguente metodo stampa il messaggio e il messaggio dettagliato di un oggetto AdvisoryMessage.

```
// Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazione del r
if(response.getStatusCode() != Response.STATUS_SUCCESS)
{
 for(AdvisoryMessage msg : response.getAdvisoryMessages())
 {
  System.out.println(msg.getMessage());
```

```
  // Alcuni messaggi di avviso possono avere ulteriori dettagli:
  System.out.println(msg.getDetailMessage());
 }
}
```

getMessageCode

Il metodo `getMessageCode` restituisce il codice di errore interno associato a un oggetto messaggio di avviso se il livello di stato è 2 (STATUS_LEVEL_ERROR).

```
getMessageCode()
```

Valore restituito

L'oggetto AdvisoryMessage restituisce un numero intero.

Esempio

Il seguente metodo stampa il codice messaggio di un oggetto AdvisoryMessage.

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

getStatusLevel

Il metodo `getStatusLevel` restituisce il livello di stato di un oggetto messaggio di avviso.

```
getStatusLevel()
```

Valore restituito

L'oggetto messaggio di avviso restituisce un numero intero.

- 0 - STATUS_LEVEL_SUCCESS-Il metodo richiamato è stato completato senza errori.
- 1 - STATUS_LEVEL_WARNING-Il metodo richiamato è stato completato con almeno un avviso (ma nessun errore).
- 2 - STATUS_LEVEL_ERROR-Il metodo richiamato non è stato completato correttamente e presenta almeno un errore.

Esempio

Il seguente metodo stampa il livello di stato di un oggetto AdvisoryMessage.

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

Informazioni sulla classe AdvisoryMessageCode

La classe `advisoryMessageCode` contiene metodi che definiscono i codici del messaggio di avviso. Si richiamano i codici del messaggio di avviso mediante il metodo `getMessageCode`.

Codici dei messaggi di avviso

I codici dei messaggi di avviso vengono recuperati con il metodo `getMessageCode`.

Questa tabella elenca e descrive i codici dei messaggi di avviso.

Codice	Testo del messaggio	Descrizione
1	INVALID_SESSION_ID	L'ID sessione non fa riferimento ad una sessione valida.
2	ERROR_TRYING_TO_ABORT_ SEGMENTATION	Si è verificato un errore nel tentativo di interrompere la segmentazione durante la chiamata `endSession`.
3	INVALID_INTERACTIVE_CHANNEL	L'argomento trasmesso per il canale interattivo non fa riferimento ad un canale interattivo valido.
4	INVALID_EVENT_NAME	L'argomento trasmesso per l'evento non fa riferimento ad un evento valido per il canale interattivo corrente.
5	INVALID_INTERACTION_POINT	L'argomento trasmesso per il punto di interazione non fa riferimento ad un punto di interazione valido per il canale interattivo corrente.
6	ERROR_WHILE_MAKING_INITIAL_ SEGMENTATION_REQUEST	Si è verificato un errore durante l'invio di una richiesta di segmentazione.
7	SEGMENTATION_RUN_FAILED	La segmentazione è stata eseguita parzialmente ma ha avuto come risultato un errore.
8	PROFILE_LOAD_FAILED	Il tentativo di caricare le tabelle dimensionali o dei profili non è riuscito.
9	OFFER_SUPPRESSION_LOAD_FAILED	Il tentativo di caricare la tabella delle soppressioni dell'offerta non è riuscito.
10	COMMAND_METHOD_ UNRECOGNIZED	Un metodo di comando specificato per un comando all'interno di una chiamata `executeBatch` non è valido.
11	ERROR_TRYING_TO_POST_EVENT _PARAMETERS	Si è verificato un errore durante l'invio dei parametri dell'evento.
12	LOG_SYSTEM_EVENT_EXCEPTION	Si è verificata un'eccezione durante il tentativo di inoltrare un evento di sistema (Fine sessione, Ottieni offerta, Ottieni profilo, Imposta destinatario, Imposta debug o Avvia sessione) per la registrazione.
13	LOG_USER_EVENT_EXCEPTION	Si è verificata un'eccezione durante il tentativo di inoltrare un evento utente per la registrazione.
14	ERROR_TRYING_TO_LOOK_UP_EVENT	Si è verificato un errore durante il tentativo di ricerca del nome evento.
15	ERROR_TRYING_TO_LOOK_UP _INTERACTIVE_CHANNEL	Si è verificato un errore durante il tentativo di ricerca del nome del canale interattivo.
16	ERROR_TRYING_TO_LOOK_UP _INTERACTION_POINT	Si è verificato un errore durante il tentativo di ricerca del nome del punto di interazione.

Codice	Testo del messaggio	Descrizione
17	RUNTIME_EXCEPTION_ ENCOUNTERED	È stata rilevata un'eccezione di runtime non prevista.
18	ERROR_TRYING_TO_EXECUTE_ ASSOCIATED_ACTION	Si è verificato un errore durante il tentativo di eseguire l'azione associata (Attiva risegmentazione, Registra contatto offerta, Registra accettazione offerta o Registra rifiuto offerta).
19	ERROR_TRYING_RUN_FLOWCHART	Si è verificato un errore durante il tentativo di eseguire il diagramma di flusso.
20	FLOWCHART_FAILED	L'esecuzione di un diagramma di flusso non è riuscita.
21	FLOWCHART_ABORTED	L'esecuzione di un diagramma di flusso è stata interrotta.
22	FLOWCHART_NEVER_RUN	Un diagramma di flusso specificato non è mai stato eseguito.
23	FLOWCHART_STILL_RUNNING	Un diagramma di flusso è ancora in esecuzione.
24	ERROR_WHILE_READING_ PARAMETERS	Si è verificato un errore durante la lettura dei parametri.
25	ERROR_WHILE_LOADING_ RECOMMENDED_OFFERS	Errore durante il caricamento delle offerte consigliate
26	ERROR_WHILE_LOGGING_DEFAULT _TEXT_STATISTICS	Si è verificato un errore durante la registrazione delle statistiche di testo predefinite (il numero di volte in cui viene visualizzata la stringa predefinita per il punto di interazione).
27	SCORE_OVERRIDE_LOAD_FAILED	Il caricamento della tabella di sovrascrittura dei punteggi non è riuscito.
28	NULL_AUDIENCE_ID	L'identificativo destinatario è vuoto.
29	UNRECOGNIZED_AUDIENCE_LEVEL	È stato specificato un livello destinatario non riconosciuto.
30	MISSING_AUDIENCE_FIELD	Manca un campo destinatario.
31	INVALID_AUDIENCE_FIELD_TYPE	È stato specificato un tipo di campo destinatario non valido.
32	UNSUPPORTED_AUDIENCE_FIELD _TYPE	Tipo di campo destinatario non supportato
33	TIMEOUT_REACHED_ON_GET_ OFFERS_CALL	La chiamata `getOffers` ha raggiunto il timeout senza restituire offerte.
34	INTERACT_INITIALIZATION_NOT_ COMPLETED_SUCCESSFULLY	L'inizializzazione del server di runtime non è stata completata correttamente.
35	SESSION_ID_UNDEFINED	L'identificativo sessione non è definito.
36	INVALID_NUMBER_OF_OFFERS_ REQUESTED	È stato richiesto un numero non valido di offerte.
37	NO_SESSION_EXIST_BUT_WILL_ CREATE_NEW_ONE	Non esisteva alcuna sessione ma ne è stata creata una.
38	AUDIENCE_ID_NOT_FOUND_IN_ PROFILE_TABLE	L'identificativo destinatario specificato non è nella tabella profili.

Codice	Testo del messaggio	Descrizione
39	LOG_CUSTOM_LOGGER_EVENT_ EXCEPTION	Si è verificata un'eccezione durante il tentativo di inoltrare un evento registrazione dati personalizzato.
40	SPECIFIED_FLOWCHART_FOR_ EXECUTION_DOES_NOT_EXIST	Il diagramma di flusso specificato non può essere eseguito perché non esiste.
41	AUDIENCE_NOT_DEFINED_IN_ CONFIGURATION	Il destinatario specificato non è definito nella configurazione.

Informazioni sulla classe BatchResponse

La classe `BatchResponse` contiene i metodi che definiscono i risultati del metodo `executeBatch`.

L'oggetto risposta batch contiene i seguenti attributi:

- **BatchStatusCode**-Il valore più elevato di codice stato per tutte le risposte richieste dal metodo `executeBatch`.
- **Responses**-Un array di oggetti risposta richiesti dal metodo `executeBatch`.

getBatchStatusCode

Il metodo `getBatchStatusCode` restituisce il codice di stato più elevato dall'array di comandi eseguiti dal metodo `executeBatch`.

```
getBatchStatusCode()
```

Valore restituito

Il metodo `getBatchStatusCode` restituisce un numero intero.

- 0 - STATUS_SUCCESS-Il metodo richiamato è stato completato senza errori.
- 1 - STATUS_WARNING-Il metodo richiamato è stato completato con almeno un avviso (ma senza errori).
- 2 - STATUS_ERROR-Il metodo richiamato non è stato completato correttamente ed ha almeno un errore.

Esempio

Il seguente codice campione fornisce un esempio di come recuperare il `BatchStatusCode`.

```
// Il codice di stato di livello principale è una scorciatoia per determinare se vi sono
// esiti negativi nell'array di oggetti risposta
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");
}

// Scorrere l'array, e stampare il messaggio per eventuali esiti negativi
for(Response response : batchResponse.getResponses())
{
```

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

getResponses

Il metodo `getResponses` restituisce l'array di oggetti risposta che corrisponde all'array di comandi eseguiti dal metodo `executeBatch`.

```
getResponses()
```

Valore restituito

Il metodo `getResponses` restituisce un array di oggetti `Response`.

Esempio

Il seguente esempio seleziona tutte le risposte e stampa gli eventuali messaggi di avviso se il comando non ha avuto esito positivo.

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

Informazioni sull'interfaccia comandi

Il metodo `executeBatch` richiede di trasmettere un array di oggetti che implementa l'interfaccia comandi. Si dovrebbe utilizzare l'implementazione predefinita, `CommandImpl` per trasmettere gli oggetti comando.

La seguente tabella elenca il comando, il metodo della classe InteractAPI che il comando rappresenta e i metodi dell'interfaccia comandi che è necessario utilizzare per ciascun comando. Non è necessario includere un ID sessione, poiché il metodo `executeBatch` include già l'ID sessione.

Comando	Metodo API Interact	Metodi interfaccia comandi
COMMAND_ENDSESSION	endSession	Nessuno.
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	Nessuno.
COMMAND_GETVERSION	getVersion	Nessuno.
COMMAND_POSTEVENT	postEvent	• setEvent • setEventParameters
COMMAND_SETAUDIENCE	setAudience	• setAudienceID • setAudienceLevel • setEventParameters
COMMAND_SETDEBUG	setDebug	setDebug

Comando	Metodo API Interact	Metodi interfaccia comandi
COMMAND_STARTSESSION	startSession	• setAudienceID • setAudienceLevel • setDebug • setEventParameters • setInteractiveChannel • setRelyOnExistingSession

setAudienceID

Il metodo `setAudienceID` definisce l'AudienceID per i comandi `setAudience` e `startSession`.

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

- **audienceID** - un array di oggetti NameValuePair che definiscono l'AudienceID.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo `executeBatch` che richiama `startSession` e `setAudience`.

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

/** Elaborare la risposta in modo appropriato */
    processExecuteBatchResponse(batchResponse);
```

setAudienceLevel

Il metodo `setAudienceLevel` definisce il livello destinatario per i comandi `setAudience` e `startSession`.

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

- *audienceLevel* - una stringa che contiene il livello destinatario.

Importante: Il nome dell'*audienceLevel* deve corrispondere esattamente al nome del livello destinatario definito in Campaign. È sensibile al maiuscolo/minuscolo.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo `executeBatch` che richiama `startSession` e `setAudience`.

```
String audienceLevel="Customer";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceLevel(audienceLevel);
. . .
/** Build command array */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Make the call */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

/** Elaborare la risposta in modo appropriato */
    processExecuteBatchResponse(batchResponse);
```

setDebug

Il metodo `setDebug` definisce il livello di debug per il comando `startSession`.

`setDebug(`*`debug`*`)`

Se è `true`, il server di runtime registra le informazioni di debug nel log del server di runtime. Se è `false`, il server di runtime non registra informazioni di debug. L'indicatore di debug viene impostato singolarmente per ogni sessione. Pertanto è possibile tenere traccia dei dati debut per una singola sessione di runtime.

- **debug** - un valore booleano (`true` o `false`).

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo `executeBatch` che richiama `startSession` e `setDebug`.

```
boolean initialDebugFlag=true;
boolean newDebugFlag=false;
. . .
/* creare il comando startSession */
Command startSessionCommand = new CommandImpl();
startSessionCommand.setDebug(initialDebugFlag);
. . .

/* creare il comando setDebug */
```

```
Command setDebugCommand = new CommandImpl();
setDebugCommand.setMethodIdentifier(Command.COMMAND_SETDEBUG);
setDebugCommand.setDebug(newDebugFlag);

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

/** Elaborare la risposta in modo appropriato */
    processExecuteBatchResponse(batchResponse);
```

setEvent

Il metodo `setEvent` definisce il nome dell'evento utilizzato dal comando `postEvent`.

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

- **event** - Una stringa che contiene il nome dell'evento.

Importante: Il nome dell'*event* deve corrispondere esattamente al nome dell'evento definito nel canale interattivo. È sensibile al maiuscolo/minuscolo.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo `executeBatch` che richiama `postEvent`.

```
String eventName = "SearchExecution";

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

setEventParameters

Il metodo `setEventParameters` definisce i parametri evento utilizzati dal comando `postEvent`. Questi valori sono memorizzati nei dati di sessione.

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

- **eventParameters** - un array di oggetti NameValuePair che definisce i parametri evento.

Ad esempio, se l'evento sta registrando un offerta nella cronologia dei contatti, è necessario includere il codice trattamento dell'offerta.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo `executeBatch` che richiama `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

Il metodo **setGetOfferRequests** imposta il parametro per recuperare le offerte utilizzate dal comando `getOffersForMultipleInteractionPoints`.

`setGetOfferRequests(`*numberRequested*`)`

- **numberRequested** - un array di oggetti `GetOfferRequest` che definisce il parametro per recuperare le offerte.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo `GetOfferRequest` che richiama `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

Il metodo `setInteractiveChannel` definisce il nome del canale interattivo utilizzato dal comando `startSession`.

`setInteractiveChannel(`*`interactiveChannel`*`)`

- **interactiveChannel** - una stringa che contiene il nome del canale interattivo.

Importante: L'*interactiveChannel* deve corrispondere esattamente al nome del canale interattivo definito in Campaign È sensibile al maiuscolo/minuscolo.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo `executeBatch` che richiama `startSession`.

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

setInteractionPoint

Il metodo `setInteractionPoint` definisce il nome del punto di interazione utilizzato dai comandi `getOffers` e `postEvent`.

setInteractionPoint(*interactionPoint*)

- **interactionPoint** - una stringa che contiene il nome del punto di interazione.

Importante: L'*interactionPoint* deve corrispondere esattamente al nome del punto di interazione definito nel canale interattivo. È sensibile al maiuscolo/minuscolo.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo executeBatch che richiama 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

Il metodo setMethodIdentifier definisce il tipo di comando contenuto nell'oggetto comando.

setMethodIdentifier(*methodIdentifier*)

- **methodIdentifier** - una stringa che contiene il tipo di comando.

  I valori validi sono:
  - **COMMAND_ENDSESSION** - rappresenta il metodo endSession.
  - **COMMAND_GETOFFERS** - rappresenta il metodo getOffers.
  - **COMMAND_GETPROFILE** - rappresenta il metodo getProfile.
  - **COMMAND_GETVERSION** - rappresenta il metodo getVersion.
  - **COMMAND_POSTEVENT** - rappresenta il metodo postEvent.
  - **COMMAND_SETAUDIENCE** - rappresenta il metodo setAudience.
  - **COMMAND_SETDEBUG** - rappresenta il metodo setDebug.
  - **COMMAND_STARTSESSION** - rappresenta il metodo startSession.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo executeBatch che richiama getVersion e endSession.

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

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

Command[] commands =
```

```
{
    getVersionCommand,
    endSessionCommand
};
```

setNumberRequested

Il metodo setNumberRequested definisce il numero di offerte richieste dal comando getOffers.

```
setNumberRequested(numberRequested)
```

- **numberRequested** - un numero intero che definisce il numero di offerte richieste dal comando getOffers.

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo executeBatch che richiama 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

Il metodo setRelyOnExistingSession definisce un valore booleano che indica se il comando startSession utilizza o meno una sessione esistente.

```
setRelyOnExistingSession(relyOnExistingSession)
```

Se è true, l'ID sessione per executeBatch deve corrispondere a un ID sessione esistente. Se è false, è necessario fornire un nuovo ID sessione con il metodo executeBatch.

- **relyOnExistingSession** - un valore booleano (true o false).

Valore restituito

Nessuno.

Esempio

Il seguente esempio è un estratto di un metodo executeBatch che richiama startSession.

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

Informazioni sull'interfaccia NameValuePair

Molti metodi nell'API di Interact, restituiscono oggetti NameValuePair o richiedono la trasmissione di oggetti NameValuePair come argomenti. Quando si trasmettono argomenti ad un metodo, sarebbe opportuno utilizzare l'implementazione predefinita `NameValuePairImpl`.

getName

Il metodo `getName` restituisce il componente nome di un oggetto NameValuePair.

```
getName()
```

Valore restituito

Il metodo `getName` restituisce una stringa.

Esempio

Il seguente esempio è un estratto di un metodo che elabora l'oggetto risposta per `getProfile`.

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

getValueAsDate

Il metodo `getValueAsDate` restituisce il valore di un oggetto NameValuePair.

```
getValueAsDate()
```

Utilizzare `getValueDataType` prima di utilizzare `getValueAsDate` per confermare che si sta facendo riferimento al tipo di dati corretto.

Valore restituito

Il metodo `getValueAsDate` restituisce una data.

Esempio

Il seguente esempio è un estratto di un metodo che elabora NameValuePair e stampa il valore se si tratta di una data.

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

getValueAsNumeric

Il metodo `getValueAsNumeric` restituisce il valore di un oggetto NameValuePair.

```
getValueAsNumeric()
```

Utilizzare `getValueDataType` prima di utilizzare `getValueAsNumeric` per confermare che si sta facendo riferimento al tipo di dati corretto.

Valore restituito

Il metodo `getValueAsNumeric` restituisce un valore doppio. Se, ad esempio, si sta recuperando un valore memorizzato originariamente nella tabella profili come

numero intero, getValueAsNumeric restituisce un valore doppio.

Esempio

Il seguente esempio è un estratto di un metodo che elabora un NameValuePair e stampa il valore se è numerico.

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

getValueAsString

Il metodo getValueAsString restituisce il valore di un oggetto NameValuePair.

```
getValueAsString()
```

Utilizzare getValueDataType prima di utilizzare getValueAsString per confermare che si sta facendo riferimento al tipo di dati corretto.

Valore restituito

Il metodo getValueAsString restituisce una stringa. Se, ad esempio, si sta recuperando un valore originariamente memorizzato nella tabella profili come char, varchar o char[10], getValueAsString restituisce una stringa.

Esempio

Il seguente esempio è un estratto di un metodo che elabora un NameValuePair e stampa il valore se si tratta di una stringa.

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

getValueDataType

Il metodo getValueDataType restituisce il tipo di dati di un oggetto NameValuePair.

```
getValueDataType()
```

Utilizzare getValueDataType prima di utilizzare getValueAsDate, getValueAsNumeric, o getValueAsString per confermare che si sta facendo riferimento al tipo di dati corretto.

Valore restituito

Il metodo getValueDataType restituisce una stringa che indica se NameValuePair contiene dati, un numero o una stringa.

I valori validi sono:

- **DATA_TYPE_DATETIME** - una data che contiene un valore di data e ora.
- **DATA_TYPE_NUMERIC** - un valore doppio che contiene un valore numerico.
- **DATA_TYPE_STRING** - una stringa che contiene un valore di testo.

Esempio

Il seguente esempio è un estratto di un metodo che elabora l'oggetto risposta da un metodo getProfile.

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

setName

Il metodo setName definisce il componente nome di un oggetto NameValuePair.

setName(*name*)

- **name** - una stringa che contiene il componente nome di un oggetto NameValuePair.

Valore restituito

Nessuno.

Esempio

Il seguente esempio mostra in che modo definire il componente nome di un NameValuePair.

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

setValueAsDate

Il metodo setValueAsDate definisce il valore di un oggetto NameValuePair.

setValueAsDate(*valueAsDate*)

- **valueAsDate** - una data che contiene il valore di data e ora dell'oggetto NameValuePair.

Valore restituito

Nessuno.

Esempio

Il seguente esempio mostra come definire il componente valore di un NameValuePair se il valore è una data.

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

setValueAsNumeric

Il metodo `setValueAsNumeric` definisce il valore di un oggetto NameValuePair.

`setValueAsNumeric(`*`valueAsNumeric`*`)`

- **valueAsNumeric** - un valore doppio che contiene il valore numerico di un oggetto NameValuePair.

Valore restituito

Nessuno.

Esempio

Il seguente esempio mostra come definire il componente valore di un oggetto NameValuePair se il valore è numerico.

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

setValueAsString

Il metodo `setValueAsString` definisce il valore di un oggetto NameValuePair.

`setValueAsString(`*`valueAsString`*`)`

- **valueAsString** - una stringa che contiene il valore di un oggetto NameValuePair

Valore restituito

Nessuno.

Esempio

Il seguente esempio mostra come definire il componente valore di un oggetto NameValuePair se il valore è numerico.

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

setValueDataType

Il metodo `setValueDataType` definisce il tipo di dati di un oggetto NameValuePair.

`getValueDataType(`*`valueDataType`*`)`

I valori validi sono:

- **DATA_TYPE_DATETIME** - una data che contiene un valore di data e ora.
- **DATA_TYPE_NUMERIC** - un valore doppio che contiene un valore numerico.
- **DATA_TYPE_STRING** - una stringa che contiene un valore di testo.

Valore restituito

Nessuno.

Esempio

I seguenti esempi mostrano come impostare il tipo di dati del valore di un NameValuePair.

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

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

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

Informazioni sulla classe Offerta

Una classe `Offerta` contiene metodi che definiscono un oggetto offerta. Tale oggetto offerta contiene molte delle proprietà definite per un'offerta in Campaign.

L'oggetto offerta contiene i seguenti attributi:

- **AdditionalAttributes**-NameValuePair contenenti eventuali attributi offerta personalizzati, definiti in Campaign.
- **Description**-La descrizione dell'offerta.
- **EffectiveDate**-La data di validità dell'offerta.
- **ExpirationDate**-La data di scadenza dell'offerta.
- **OfferCode**-Il codice offerta relativo all'offerta.
- **OfferName**-Il nome dell'offerta.
- **TreatmentCode**-Il codice trattamento per l'offerta.
- **Score**-Il punteggio di marketing attribuito all'offerta oppure il punteggio definito da `ScoreOverrideTable` se la proprietà `enableScoreOverrideLookup` è impostata su true.

getAdditionalAttributes

Il metodo `getAdditionalAttributes` restituisce gli attributo personalizzato dell'offerta definiti in Campaign.

```
getAdditionalAttributes()
```

Valore restituito

Il metodo `getAdditionalAttributes` restituisce un array di oggetti NameValuePair.

Esempio

Il seguente esempio ordina tutti gli attributi aggiuntivi, controllando la data di validità e la data di scadenza, e stampa gli altri attributi.

```
for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
{
    // controlla se esiste la data di validità
    if(offerAttribute.getName().equalsIgnoreCase("effectiveDate"))
    {
        System.out.println("Found effective date");
```

```
        }
        // controlla se esiste la data di scadenza
        else if(offerAttribute.getName().equalsIgnoreCase("expirationDate"))
        {
            System.out.println("Found expiration date");
        }
        printNameValuePair(offerAttribute);
        }
    }
    public static void printNameValuePair(NameValuePair nvp)
    {
        // stampa il nome:
        System.out.println("Name:"+nvp.getName());

        // in base al tipo di dati, chiama il metodo appropriato per ottenere il valore
        if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_DATETIME)
            System.out.println("DateValue:"+nvp.getValueAsDate());
        else if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_NUMERIC)
            System.out.println("NumericValue:"+nvp.getValueAsNumeric());
        else
            System.out.println("StringValue:"+nvp.getValueAsString());
    }
```

getDescription

Il metodo `getDescription` restituisce la descrizione dell'offerta definita in Campaign.

```
getDescription()
```

Valore restituito

Il metodo `getDescription` restituisce una stringa.

Esempio

Il seguente esempio stampa la descrizione di un'offerta.

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

getOfferCode

Il metodo `getOfferCode` restituisce il codice offerta dell'offerta definito in Campaign.

```
getOfferCode()
```

Valore restituito

Il metodo `getOfferCode` restituisce un array di stringhe che contengono il codice offerta dell'offerta.

Esempio

Il seguente esempio stampa il codice offerta di un'offerta.

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

getOfferName

Il metodo `getOfferName` restituisce il nome dell'offerta definito in Campaign.

```
getOfferName()
```

Valore restituito

Il metodo `getOfferName` restituisce una stringa.

Esempio

Il seguente esempio stampa il nome di un'offerta.

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

getScore

Il metodo `getScore` restituisce un punteggio, che s basa sulle offerte configurate.

```
getScore()
```

Il metodo `getScore` restituisce uno dei seguenti punteggi:

- Se non è stata abilitata la tabella delle offerte predefinite, la tabella di sovrascrittura dei punteggi o l'apprendimento integrato, questo metodo restituisce il punteggio di marketing dell'offerta definito nella scheda della strategia di interazione.
- Se è stata abilitata la tabella delle offerte predefinite o di sovrascrittura dei punteggi e non è stato abilitato l'apprendimento integrato, questo metodo restituisce il punteggio dell'offerta definito dall'ordine di precedenza tra la tabella di offerte predefinite, il punteggio del marketer e la tabella di sovrascrittura dei punteggi.
- Se si abilital'apprendimento integrato, questo metodo restituisce il punteggio finale che l'apprendimento integrato ha utilizzato per ordinare le offerte.

Valore restituito

Il metodo `getScore` restituisce un numero intero che rappresenta il punteggio dell'offerta.

Esempio

Il seguente esempio stampa il punteggio di un'offerta.

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

getTreatmentCode

Il metodo `getTreatmentCode` restituisce il codice trattamento dell'offerta definito in Campaign.

```
getTreatmentCode()
```

Poiché Campaign utilizza il codice trattamento per identificare l'istanza dell'offerta presentata, questo codice deve essere restituito come parametro di evento quando si utilizza il metodo `postEvent` per registrare un evento di contatto, accettazione o rifiuto dell'offerta. Se si sta registrando un'accettazione o un rifiuto di un'offerta, è necessario impostare il valore nome di NameValuePair che rappresenta il codice trattamento per `UACIOfferTrackingCode`.

Valore restituito

Il metodo `getTreatmentCode` restituisce una stringa.

Esempio

Il seguente esempio stampa il codice trattamento di un'offerta.

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

Informazioni sulla classe OfferList

La classe `OfferList` contiene metodi che definiscono i risultati del metodo `getOffers`.

L'oggetto OfferList contiene i seguenti attributi:

- **DefaultString**- La stringa predefinita, definita per il punto di interazione, nel canale interattivo.
- **RecommendedOffers**-Un array di oggetti offerta richiesti dal metodo `getOffers`.

La classe `OfferList` utilizza elenchi di offerte. Questa classe non è correlata agli elenchi di offerte di Campaign.

getDefaultString

Il metodo `getDefaultString` restituisce la stringa predefinita per il punto di interazione definito in Campaign.

```
getDefaultString()
```

Se l'oggetto RecommendedOffers è vuoto, configurare il touchpoint per presentare questa stringa per assicurarsi che venga presentato del contenuto. Interact popola l'oggetto DefaultString solo se l'oggetto RecommendedOffers è vuoto.

Valore restituito

Il metodo `getDefaultString` restituisce una stringa.

Esempio

Il seguente esempio ottiene la stringa predefinita se l'oggetto offerList non contiene offerte.

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

```
    }
}
else // count on the default Offer String
    System.out.println("Default offer:"+offerList.getDefaultString());
```

getRecommendedOffers

Il metodo `getRecommendedOffers` restituisce un array di oggetti offerta richiesti dal metodo `getOffers`.

```
getRecommendedOffers()
```

Se la risposta a `getRecommendedOffer` è vuota, il touchpoint deve presentare il risultato di `getDefaultString`.

Valore restituito

Il metodo `getRecommendedOffers` restituisce un oggetto offerta.

Esempio

Il seguente esempio elabora l'oggetto OfferList e stampa il nome offerta per tutte le offerte consigliate.

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

Informazioni sulla classe risposta

La classe `risposta` contiene metodi che definiscono i risultati di uno qualsiasi dei metodi della classe InteractAPI.

L'oggetto risposta contiene i seguenti attributi:

- **AdvisoryMessages**-un array di messaggi di avviso. Questo attributo è popolato solo se sono stati rilevati avvertenze o errori durante l'esecuzione del metodo.
- **ApiVersion**-una stringa contenente la versione dell'API. Questo attributo viene popolato dal metodo `getVersion`.
- **OfferList**-l'oggetto OfferList contenente le offerte richieste dal metodo `getOffers`.
- **ProfileRecord**-un array di NameValuePair contenenti i dati del profilo. Questo attributo viene popolato dal metodo `getProfile`.
- **SessionID**-una stringa che definisce l'ID sessione. Questo valore viene restituito da tutti i metodi della classe `InteractAPI`.
- **StatusCode**-un numero che indica se l'esecuzione del metodo si è svolta senza errori, con un'avvertenza o con errori. Questo valore viene restituito da tutti i metodi della classe `InteractAPI`.

getAdvisoryMessages

Il metodo `getAdvisoryMessages` restituisce un array di messaggi di avviso dall'oggetto risposta.

```
getAdvisoryMessages()
```

Valore restituito

Il metodo `getAdvisoryMessages` restituisce un array di oggetti messaggio di avviso.

Esempio

Il seguente esempio ottiene gli oggetti AdvisoryMessage dall'oggetto risposta e li scorre, stampando i messaggi.

```
AdvisoryMessage[] messages = response.getAdvisoryMessages();
    for(AdvisoryMessage msg : messages)
    {
        System.out.println(msg.getMessage());
        // Alcuni messaggi di avviso possono avere ulteriori dettagli:
        System.out.println(msg.getDetailMessage());
    }
```

getApiVersion

Il metodo `getApiVersion` restituisce la versione API di un oggetto risposta.

```
getApiVersion()
```

Il metodo `getVersion` popola l'attributo ApiVersion di un oggetto risposta.

Valore restituito

L'oggetto risposta restituisce una stringa.

Esempio

Il seguente esempio è un estratto di un metodo che elabora l'oggetto risposta per `getVersion`.

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

getOfferList

Il metodo `getOfferList` restituisce l'oggetto OfferList di un oggetto risposta.

```
getOfferList()
```

Il metodo `getOffers` popola l'oggetto OfferList di un oggetto risposta.

Valore restituito

L'oggetto risposta restituisce un oggetto OfferList.

Esempio

Il seguente esempio è un estratto di un metodo che elabora l'oggetto risposta per `getOffers`.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
```

```
    // stampare l'offerta
    System.out.println("Offer Name:"+offer.getOfferName());
    }
}
```

getAllOfferLists

Il metodo `getAllOfferLists` restituisce un array di tutti gli OfferLists di un oggetto risposta.

```
getAllOfferLists()
```

Viene utilizzato dal metodo `getOffersForMultipleInteractionPoints` che popola l'oggetto array OfferList di un oggetto risposta.

Valore restituito

L'oggetto risposta restituisce un array OfferList.

Esempio

Il seguente esempio è un estratto di un metodo che elabora l'oggetto risposta per `getOffers`.

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

getProfileRecord

Il metodo `getProfileRecord` restituisce i record di profilo per la sessione corrente come array di oggetti NameValuePair. Questi record di profilo includono anche gli eventuali `eventParameters` aggiunti in precedenza nella sessione di runtime.

```
getProfileRecord()
```

Il metodo `getProfile` popola gli oggetti NameValuePair dei record di profilo di un oggetto risposta.

Valore restituito

L'oggetto risposta restituisce un array di oggetti NameValuePair.

Esempio

Il seguente esempio è un estratto di un metodo che elabora l'oggetto risposta per `getOffers`.

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

```
            System.out.println("Value:"+nvp.getValueAsNumeric());
        }
        else
        {
            System.out.println("Value:"+nvp.getValueAsString());
        }
    }
```

getSessionID

Il metodo getSessionID restituisce l'ID sessione.

```
getSessionID()
```

Valore restituito

Il metodo getSessionID restituisce una stringa.

Esempio

Il seguente esempio mostra un messaggio che è possibile visualizzare al termine o all'inizio della gestione errori per indicare a quale sessione appartiene un errore.

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

getStatusCode

Il metodo getStatusCode restituisce il codice di stato di un oggetto risposta.

```
getStatusCode()
```

Valore restituito

L'oggetto risposta restituisce un numero intero.

- 0 - STATUS_SUCCESS - Il metodo richiamato è stato completato senza errori. Potrebbero essere o non essere presenti messaggi di avviso.
- 1 - STATUS_WARNING - Il metodo richiamato è stato completato con almeno un messaggio di avviso (ma nessun errore). Per ulteriori dettagli eseguire query nei messaggi di avviso.
- 2 - STATUS_ERROR - Il metodo richiamato non è stato completato correttamente edha almeno un messaggio di errore. Per ulteriori dettagli eseguire query nei messaggi di avviso.

Esempio

Di seguito è riportato un esempio di come è possibile utilizzare getStatusCode nella gestione degli errori.

```
public static void processSetDebugResponse(Response response)
{
    // controllare l'esito della risposta
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("setDebug call processed with no warnings or errors");
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("setDebug call processed with a warning");
    }
    else
    {
        System.out.println("setDebug call processed with an error");
    }
```

```
    // Per le risposte con esito negativo, devono esserci dei messaggi di avviso con la spiegazior
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("setDebug",
    response.getAdvisoryMessages());
}
```

Capitolo 8. Classi e metodi per l'API JavaScript IBM Interact

Le seguenti sezioni elencano requisiti e altri dettagli che sarebbe necessario conoscere prima di utilizzare l'API JavaScript Interact.

L'API Interact supporta javascript per consentire la comunicazione da client utente finale (browser) a server.

Nota: in questa sezione si presuppone esperienza con un API basata su JavaScript.

Nota: non sono supportate più ricorrenze di un qualunque parametro in una singola chiamata all'API.

Prerequisiti JavaScript

Prima di utilizzare l'API JavaScript Interact su un sito web, è necessario includere il file `interactapi.js` nelle pagine web.

Gestione dei dati di sessione

Quando si avvia una sessione con il metodo `startSession`, i dati di sessione vengono caricati nella memoria. Nel corso di tutta la sessione, è possibile leggere e scrivere i dati di sessione (che rappresentano una super serie dei dati del profilo statici).

La sessione contiene i seguenti dati:

- Dati del profilo statici
- Assegnazioni del segmento
- Dati in tempo reale
- Raccomandazioni di offerte

Tutti i dati di sessione rimangono disponibili fino a quando non viene richiamato il metodo `endSession` o non scade il tempo specificato per `sessionTimeout`. Una volta terminata la sessione, tutti i dati non salvati esplicitamente nella cronologia dei contatti o delle risposta oppure in qualche altra tabella database, verranno persi.

I dati vengono memorizzati come una serie di coppie nome-valore. Se i dati vengono letti da una tabella del database, il nome corrisponde alla colonna della tabella.

È possibile creare queste coppie nome-valore mentre si utilizza l'API di Interact. Non è necessario dichiarare tutte le coppie nome-valore in un'area globale. Se si impostano nuovi parametri evento come coppie nome-valore, l'ambiente di runtime aggiunge le coppie nome-valore ai dati di sessione. Ad esempio, se si utilizzano parametri evento con il metodo `postEvent`, l'ambiente di runtime aggiunge i parametri evento ai dati di sessione, anche se i parametri evento non erano disponibili nei dati del profilo. Questi dati esistono solo nei dati di sessione.

È possibile sovrascrivere i dati di sessione in qualsiasi momento. Ad esempio, se parte del profilo cliente include `creditScore`, è possibile trasmettere un parametro evento utilizzando il tipo personalizzato `NameValuePair`. Nella classe `NameValuePair`, è possibile utilizzare i metodi `setName` e `setValueAsNumeric` per

modificare il valore. Il nome deve corrispondere. Nei dati di sessione, il nome non è sensibile al maiuscolo/minuscolo. Pertanto, ambedue i nomi `creditscore` o `CrEdItScOrE` sovrascriverebbero `creditScore`.

Solo gli ultimi dati scritti nei dati di sessione vengono conservati. Ad esempio, `startSession` carica i dati del profilo per il valore relativo a `lastOffer`. Un metodo `postEvent` sovrascrive `lastOffer`. Quindi, un secondo metodo `postEvent` sovrascrive `lastOffer`. L'ambiente di runtime conserva solo i dati scritti dal secondo metodo `postEvent` nei dati di sessione.

Quando una sessione termina, i dati vengono persi, a meno che non vengano effettuate considerazioni speciali, quali ad esempio utilizzare un processo Snapshot nel diagramma di flusso interattivo per la scrittura dei dati in una tabella del database. Si si sta pianificando l'utilizzo di processi Snapshot, tenere a mente che i nomi devono conformarsi alle limitazioni previste dal proprio database. Ad esempio, se è consentito un massimo di 256 caratteri per il nome di una colonna, allora il nome della coppia nome-valore non dovrebbe superare i 256 caratteri.

Utilizzo del parametro di richiamata

La funzione di richiamata è un parametro aggiuntivo di ciascun metodo dell'API JavaScript Interact.

Il processo browser principale è un loop evento con singolo thread. L'esecuzione di un'operazione di lunga durata in un loop evento con singolo thread, blocca il processo. Cioè impedisce al processo di elaborare altri eventi mentre attende il completamento dell'operazione. Per evitare blocchi su operazioni di lunga durata, XMLHttpRequest fornisce un'interfaccia asincrona. Gli si passa una richiamata da eseguire dopo il completamento dell'operazione e, durante l'elaborazione, riporta il controllo al loop evento principale invece di bloccare il processo.

Se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Ad esempio, se si desidera visualizzare offerte sulla pagina web, è possibile utilizzare il metodo `getOffers` e utilizzare la richiamata per la visualizzazione sulla pagina. La pagina web si comporta normalmente e non attende che Interact restituisca le offerte. Invece, quando Interact restituisce le offerte, la risposta viene reinviata nel parametro di richiamata. È possibile analizzare i dati di richiamata e mostrare le offerte sulla pagina.

È possibile utilizzare una richiamata generica per tutte le funzioni o utilizzare richiamate specifiche per funzioni specifiche.

È possibile utilizzare `var callback = InteractAPI.Callback.create(onSuccess, onError);` per creare una funzione di richiamata generica.

È possibile utilizzare la seguente funzione per creare una funzione di richiamata specifica per il metodo getOffers.

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

Informazioni sulla classe InteractAPI

La classe InteractAPI contiene i metodi che è possibile utilizzare per integrare il touchpoint con il server di runtime. Tutte le altre classi e metodi nell'API di Interact supportano i metodi contenuti in questa classe.

È necessario compilare la propria implementazione facendo riferimento al file `interact_client.jar`, che si trova nella directory `lib` della propria installazione dell'ambiente di runtime Interact.

startSession

Il metodo `startSession` crea e definisce una sessione di runtime.

```
function callStartSession(commandsToExecute, callback) {

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

        InteractAPI.startSession(ssId, icName,
                                    getNameValuePairs(audId), audLevel,
                                    getNameValuePairs(params), relyOldSs,
                                    debug, callback) ;

}
```

`startSession` può attivare fino a cinque azioni:

- Creare una sessione di runtime.
- Caricare i dati del profilo del visitatore per il livello destinatario corrente nella sessione di runtime, incluse le eventuali tabelle dimensionali contrassegnate per il caricamento nel mapping delle tabelle definito per il canale interattivo.
- Attivare la segmentazione, eseguendo tutti i diagrammi di flusso interattivi per il livello destinatario corrente.
- Caricare i dati di soppressione dell'offerta nella sessione, se la proprietà `enableOfferSuppressionLookup` è impostata su true.
- Caricare i dati di sovrascrittura dei punteggi nella sessione, se la proprietà `enableScoreOverrideLookup` è impostata su true.

Il metodo `startSession` richiede i seguenti parametri:

- **sessionID** - una stringa che identifica l'ID sessione. È necessario definire l'ID sessione. Ad esempio, si potrebbe utilizzare una combinazione di ID cliente e data/ora.

  Per definire cosa costituisce una sessione di runtime, è necessario specificare un id sessione. Questo valore è gestito dal client. Tutte le chiamate di metodo per lo stesso id sessione devono essere sincronizzate dal client. Il comportamento delle chiamate API simultanee con lo stesso id sessione non è definito.
- **relyOnExistingSession** - un valore booleano che definisce se questa sessione utilizza una nuova sessione o una esistente. I valori validi sono `true` o `false`. Se è `true`, è necessario fornire un ID sessione esistente con il metodo `startSession`. Se è `false`, è necessario fornire un nuovo ID sessione.

  Se si imposta `relyOnExistingSession` su `true` ed esiste una sessione, l'ambiente di runtime utilizza i dati della sessione esistente e non ricarica alcun dato o

attiva la segmentazione. Se la sessione non esiste, l'ambiente di runtime crea una nuova sessione, incluso il caricamento dei dati e l'attivazione della segmentazione. L'impostazione di `relyOnExistingSession` su true e l'utilizzo con tutte le chiamate `startSession` è utile se il touchpoint ha una durata di sessione maggiore rispetto alla sessione di runtime. Ad esempio, una sessione del sito web è attiva per 2 ore, ma la sessione di runtime è attiva solo per 20 minuti.

Se si richiama `startSession` due volte con lo stesso ID sessione, tutti i dati di sessione della prima chiamata `startSession` vengono persi se `relyOnExistingSession` è false.

- **debug** - un valore booleano che abilita o disabilita le informazioni di debug. I valori validi sono `true` o `false`. Se è true, Interact registra le informazioni di debug nei log del server di runtime. L'indicatore di debug viene impostato singolarmente per ogni sessione. Pertanto è possibile tenere traccia dei dati di debug per una singola sessione.
- **interactiveChannel** - una stringa che definisce il nome del canale interattivo a cui questa sessione fa riferimento. Questo nome deve corrispondere esattamente al nome del canale interattivo definito in Campaign.
- **audienceID** - un array di oggetti NameValuePairImpl in cui i nomi devono corrispondere ai nomi colonna fisica di ogni tabella che contiene l'ID destinatario.
- **audienceLevel** - una stringa che definisce il livello destinatario.
- **parameters** - oggetti NameValuePairImpl che identificano qualsiasi parametro debba essere trasmesso con `startSession`. Questi valori vengono memorizzati nei dati di sessione e possono essere utilizzati per la segmentazione.

  Se si dispone di diversi diagrammi di flusso interattivi per lo stesso livello destinatario, è necessario includere una superinsieme di tutte le colonne in tutte le tabelle. Se si configura il runtime per il caricamento della tabella profili, e la tabella profili contiene tutte le colonne richieste, non è necessario trasmettere i parametri, a meno che non di desideri sovrascrivere i dati nella tabella profili. Se la tabella profili contiene un sottoinsieme delle colonne richieste, è necessario includere le colonne mancanti come parametri.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Se `audienceID` o `audienceLevel` non sono validi e `relyOnExistingSession` è false, la chiamata `startSession` ha esito negativo. Se `interactiveChannel` non è valido, `startSession` ha esito negativo, a prescindere dal fatto che `relyOnExistingSession` sia true o false.

Se `relyOnExistingSession` è true, e si effettua una seconda chiamata `startSession` utilizzando lo stesso `sessionID`, ma la prima sessione è scaduta, Interact crea una nuova sessione.

Se `relyOnExistingSession` è true, e si effettua una seconda chiamata `startSession` utilizzando lo stesso `sessionID` ma un `audienceID` o `audienceLevel` differente, il server di runtime modifica il destinatario per la sessione esistente.

Se `relyOnExistingSession` è true, e si effettua una seconda chiamata `startSession` utilizzando lo stesso `sessionID` ma un `interactiveChannel` diverso, il server di runtime crea una nuova sessione.

Valore restituito

Il server di runtime risponde a `startSession` con un oggetto risposta con i seguenti attributi popolati:

- AdvisoryMessages (se StatusCode non è uguale a 0)
- ApiVersion
- SessionID
- StatusCode

Attributi deduplicazione offerta nelle offerte

Utilizzando l'API (application programming interface) Interact, due chiamate API forniscono le offerte: `getOffers` e `getOffersForMultipleInteractionPoints`. `getOffersForMultipleInteractionPoints` può impedire la restituzione di offerte duplicate al livello *OfferID*, ma non offerte deduplicate nella categoria di offerte. Quindi, ad esempio, perché Interact possa restituire una sola offerta da ogni categoria di offerta, in precedenza era stata richiesta una soluzione temporanea. Con l'introduzione di due parametri alla chiamata API `startSession`, la deduplicazione delle offerte negli attributi offerta, come la categoria, ora è possibile.

Questo elenco riepiloga i parametri che sono stati aggiunti alla chiamata API `startSession`. Per ulteriori informazioni su questi parametri o qualsiasi aspetto dell'API Interact, consultare *IBM Interact - Guida dell'amministratore*, o i file Javadoc inclusi con l'installazione di Interact in `<Interact_Home>/docs/apiJavaDoc`.

- `UACIOfferDedupeAttribute`. Per creare una chiamata API `startSession` con la deduplicazione delle offerte, in modo che le chiamate `getOffer` successive restituiscano solo un'offerta da ciascuna categoria, è necessario includere il parametro `UACIOfferDedupeAttribute` come parte della chiamata API. È possibile specificare un parametro nel formato `name,value,type`, come mostrato qui:

  `UACIOfferDedupeAttribute,<attributeName>,string`

  In questo esempio, *<attributeName>* verrebbe sostituito con il nome dell'attributo offerta che si desidera utilizzare come criterio per la deduplicazione, ad esempio Categoria.

  Nota: Interact esamina le offerte che hanno lo stesso valore attributo che l'utente specifica (ad esempio Categoria) e deduplica per rimuovere tutte le offerte tranne quella che ha il punteggio più alto. Se le offerte che hanno l'attributo di deduplicazione hanno anche punteggi identici, Interact restituisce una selezione casuale tra le offerte corrispondenti.

- `UACINoAttributeDedupeIfFewerOffer`. Quando si include `UACIOfferDedupeAttribute` nella chiamata `startSession`, è possibile impostare anche questo parametro `UACINoAttributeDedupeIfFewerOffer` per specificare il comportamento nei casi in cui l'elenco di offerte dopo la deduplicazione non contenga più offerte sufficienti a soddisfare la richiesta originale.

  Ad esempio, se si imposta `UACIOfferDedupeAttribute` per utilizzare la categoria di offerta per deduplicare le offerte, e la successiva chiamata getOffers richiede che vengano restituite otto offerte, la deduplicazione potrebbe avere come risultato un numero di offerte idonee inferiore a otto. In quel caso, impostando il parametro `UACINoAttributeDedupeIfFewerOffer` su true comporterebbe l'aggiunta di alcune delle offerte duplicate all'elenco delle offerte idonee per soddisfare il

numero di offerte richiesto. In questo esempio, se si imposta il parametro su false, il numero di offerte restituite sarebbe inferiore al numero richiesto.

`UACINoAttributeDedupeIfFewerOffer` è impostato su true per impostazione predefinita.

Ad esempio, si supponga di aver specificato come parametro `startSession` che il criterio di deduplicazione sia la categoria offerta, come mostrato qui:

```
UACIOfferDedupeAttribute,Category,string;

UACINoAttributeDedupeIfFewerOffer,1,string
```

Per impostazione predefinita, `UACIOfferDedupeAttribute` non deduplica le offerte se ne viene restituito un numero inferiore a quello richiesto. Tuttavia, per assicurarsi che la deduplica si verifichi quando viene restituito un numero di offerte inferiore a quello richiesto, è necessario fornire il parametro `UACINoAttributeDedupeIfFewerOffer` e impostarlo su 1.

A causa di questi parametri insieme Interact deduplica le offerte in base all'attributo offerta "Category," e restituisce solo le offerte deduplicate anche se il numero risultante delle offerte è inferiore a quello richiesto (`UACINoAttributeDedupeIfFewerOffer` è false).

Quando si invia una chiamata API `getOffers`, la serie originale di offerte idonee potrebbe includere queste offerte:

- Category=Electronics: Offerta A1 con un punteggio di 100 e Offerta A2 con un punteggio di 50.
- Category=Smartphones: Offerta B1 con un punteggio di 100, Offerta B2 con un punteggio di 80 e Offerta B3 con un punteggio di 50.
- Category=MP3Players: Offerta C1 con un punteggio di 100, Offerta C2 con un punteggio di 50.

In questo caso, ci sono due offerte duplicate che corrispondono alla prima categoria, tre offerte duplicate che corrispondono alla seconda categoria, e due offerte duplicate che corrispondono alla terza categoria. Le offerte restituite sarebbero le offerte con il punteggio più alto di ciascuna categoria, ovvero Offerta A1, Offerta B1 e Offerta C1.

Se la chiamata API `getOffers` ha richiesto sei offerte, questo esempio imposta `UACINoAttributeDedupeIfFewerOffer` su false, in modo da restituire solo tre offerte.

Se la chiamata API `getOffers` ha richiesto sei offerte, e questo esempio ha omesso il parametro `UACINoAttributeDedupeIfFewerOffer`, oppure è stato impostato specificatamente su true, verranno incluse offerte duplicate nel risultato per soddisfare il numero richiesto.

postEvent

Il metodo `postEvent` consente di eseguire qualsiasi evento definito nel canale interattivo.

```
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 string che identifica l'ID sessione.
- **eventName**: una stringa che identifica il nome dell'evento.

  Nota: il nome dell'evento deve corrispondere al nome dell'evento definito nel canale interattivo. Questo nome non è sensibile al maiuscolo/minuscolo.
- **eventParameters**. Oggetti `NameValuePairImpl` che identificano qualsiasi parametro debba essere trasmesso con l'evento. Questi valori sono memorizzati nei dati di sessione.

  Se questo evento attiva la risegmentazione, è necessario assicurarsi che tutti i dati richiesti dai diagrammi di flusso interattivi siano disponibili nei dati della sessione. Se qualcuno di questi valori non è stato popolato da azioni precedenti (ad esempio, da `startSession` o `setAudience`, oppure caricando la tabella profili) è necessario includere un eventParameter per ogni valore mancante. Ad esempio, se tutte le tabelle profili sono state configurate per essere caricate in memoria, è necessario includere un NameValuePair per i dati temporali richiesti per i diagrammi di flusso interattivi.

  Se si sta utilizzando più di un livello destinatario, è molto probabile che si disponga di serie di eventParameters differenti per ciascun livello destinatario. È necessario includere della logica per assicurarsi di selezionare la serie corretta di parametri per il livello destinatario.

  Importante: se questo evento viene registrato nella cronologia delle risposte, è necessario passare il codice trattamento per l'offerta. È necessario definire il nome per NameValuePair "`UACIOfferTrackingCode`".

  È possibile passare un solo codice trattamento per evento. Se non si passa il codice trattamento per un contatto di offerta, Interact registra un contatto di offerta per ogni offerta presente nell'ultimo elenco di offerte consigliate. Se non si passa il codice trattamento per una risposta, Interact restituisce un errore.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.
- Sono disponibili diversi altri parametri riservati utilizzati con `postEvent` ed altri metodi e verranno trattati in seguito in questa sezione.

Qualsiasi richiesta di risegmentazione o scrittura nella cronologia dei contatti o delle risposte non attende una risposta.

La risegmentazione non cancella i precedenti risultati di segmentazione per il livello destinatario corrente. È possibile utilizzare il parametro `UACIExecuteFlowchartByName` per definire diagrammi di flusso specifici da eseguire. Il metodo `getOffers` attende il termine della risegmentazione prima dell'esecuzione. Pertanto, se si richiama un metodo `postEvent`, ciò attiva una risegmentazione immediatamente prima di una chiamata `getOffers`, potrebbe verificarsi un ritardo.

Valore restituito

Il server di runtime risponde a `postEvent` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages

- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getOffers

Il metodo `getOffers` consente di richiedere offerte dal server di runtime.

```
function callGetOffers(commandsToExecute, callback) {

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

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

}
```

- **ID sessione** - una stringa che identifica la sessione corrente.
- **Punto interazione** - una stringa che identifica il nome del punto di interazione a cui questo metodo fa riferimento.

  Nota: questo nome deve corrispondere esattamente al nome del punto di interazione definito nel canale interattivo.
- **nofRequested** - un numero intero che identifica il numero di offerte richieste.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Il metodo `getOffers` attende per il numero di millisecondi definito nella proprietà `segmentationMaxWaitTimeInMS` che siano completate tutte le nuove segmentazioni prima di passare all'esecuzione. Pertanto, se si chiama un metodo `postEvent` che attiva una nuova segmentazione o un metodo `setAudience` immediatamente prima di una chiamata `getOffers`, potrebbe verificarsi un ritardo.

Valore restituito

Il server di runtime risponde a `getOffers` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profilo
- SessionID
- StatusCode

getOffersForMultipleInteractionPoints

Il metodo `getOffersForMultipleInteractionPoints` consente di richiedere offerte dal server di runtime per più IP con la deduplicazione.

```
function callGetOffersForMultipleInteractionPoints(commandsToExecute, callback) {

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

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

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

        //build get offer requests
        var getOffReqs = [];
        for(var i = 0; i < parts.length; i += 1) {
            var getofReqObj = parseGetOfferReq(parts[i]);
            if (getofReqObj) {
                getOffReqs.push(getofReqObj);

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

- **ID sessione** - una stringa che identifica la sessione corrente.
- **requestDetailsStr** - una stringa che fornisce un array di oggetti `GetOfferRequest`.

  Ogni ogetto `GetOfferRequest` specifica:
  - **ipName** - Il nome del punto di interazione (IP) per il quale l'oggetto sta richiedendo le offerte
  - **numberRequested** - Il numero di offerte univoche di cui necessita per l'IP specificato
  - **offerAttributes** - Requisiti sugli attributi delle offerte fornite utilizzando un'istanza di `OfferAttributeRequirements`
  - **duplicationPolicy** - L'ID della politica di duplicazione per le offerte che verranno fornite

    Le politiche di duplicazione determinano se le offerte duplicate verranno restituite nei differenti punti di interazione in una singola chiamata al metodo. (*All'interno* di un singolo punto di interazione, non vengono mai restituite offerte duplicate). Attualmente, sono supportate due politiche di duplicazione.
    - NO_DUPLICATION (valore ID = 1). Nessuna delle offerte incluse nelle istanze `GetOfferRequest` precedenti verrà inclusa in questa istanza `GetOfferRequest` (ossia, Interact applicherà la deduplicazione).
    - ALLOW_DUPLICATION (valore ID = 2). Le offerte che soddisfano i requisiti specificati in questa istanza di `GetOfferRequest` verranno incluse. Le offerte incluse nelle istanze `GetOfferRequest` precedenti non verranno riconciliate.
  - **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

  L'ordine delle richieste nel parametro array è anche l'ordine di priorità quando vengono fornite le offerte.

  Ad esempio, si supponga che gli IP nella richiesta siano IP1, quindi IP2, che non siano consentite le offerte duplicate (un ID politica di duplicazione = 1), e ognuno richieda due offerte. Se Interact individua le offerte A, B e C per IP1 e le offerte A e D per IP2, la risposta conterrà le offerte A e B per IP1, e solo l'offerta D per IP2.

Notare inoltre che quando l'ID politica di duplicazione è 1, le offerte che sono state fornite tramite un IP con una priorità più elevata non verranno fornite tramite questo IP.

Il metodo `getOffersForMultipleInteractionPoints` attende per il numero di millisecondi definito nella proprietà `segmentationMaxWaitTimeInMS` che siano completate tutte le nuove segmentazioni prima di passare all'esecuzione. Pertanto, se si chiama un metodo `postEvent` che attiva una nuova segmentazione o un metodo `setAudience` immediatamente prima di una chiamata `getOffers`, potrebbe verificarsi un ritardo.

Valore restituito

Il server di runtime risponde a `getOffersForMultipleInteractionPoints` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- Array di OfferList
- Profilo
- SessionID
- StatusCode

setAudience

Il metodo `setAudience` consente di impostare ID e livello destinatario di un visitatore.

```
function callSetAudience(commandsToExecute, callback) {

        var ssId = document.getElementById('sa_sessionId').value;
        var audId = document.getElementById('sa_audienceId').value;
        var audLevel = document.getElementById('sa_audienceLevel').value;
        var params = document.getElementById('sa_parameters').value;

        InteractAPI.setAudience(ssId, getNameValuePairs(audId),audLevel,
                                    getNameValuePairs(params), callback);

}
```

- **sessionID** - una stringa che identifica l'ID sessione.
- **audienceID** - un array di oggetti `NameValuePairImpl` che definisce l'ID destinatario.
- **audienceLevel** - una stringa che definisce il livello destinatario.
- **parameters** - oggetti `NameValuePairImpl` che identificano i parametri che devono essere trasmessi con `setAudience`. Questi valori vengono memorizzati nei dati di sessione e possono essere utilizzati per la segmentazione.

  È necessario avere un valore per ogni colonna nel profilo. Questo è un superinsieme di tutte le colonne in tutte le tabelle definite per il canale interattivo e qualsiasi dato in tempo reale. Se tutti i dati di sessione sono stati già popolati con `startSession` o `postEvent`, non è necessario inviare nuovi parametri.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Il metodo setAudience attiva una risegmentazione. Il metodo getOffers attende il termine della risegmentazione prima dell'esecuzione. Pertanto, se si richiama un metodo setAudience immediatamente prima di una chiamata getOffers, potrebbe esserci un ritardo.

Il metodo setAudience inoltre carica i dati del profilo per l'ID destinatario. È possibile utilizzare il metodo setAudience per forzare un ricaricamento degli stessi dati profilo caricati dal metodo startSession.

Il metodo setAudience ricarica la tabella della white list e della black list in una sessione esistente. È possibile utilizzare il metodo setAudience con i parametri UACIPurgePriorWhiteListOnLoad e UACIPurgePriorBlackListOnLoad per ricaricare la tabella della white list e della black list in una sessione esistente.

Per impostazione predefinita, quando il metodo setAudience viene richiamato, tutti i contenuti della black list vengono rimossi. È possibile impostare i parametri UACIPurgePriorWhiteListOnLoad e UACIPurgePriorBlackListOnLoad nella chiamata setAudience come segue:

- Se si imposta UACIPurgePriorBlackListOnLoad= 0, tutti i contenuti della tabella della white list vengono conservati.
- Se si imposta UACIPurgePriorWhiteListOnLoad= 1 i contenuti della tabella vengono rimossi e i contenuti della white list o della black list per l'ID destinatario verranno caricati dal database. Una volta completata l'operazione, verrà avviata la risegmentazione.

Valore restituito

Il server di runtime risponde a setAudience con un oggetto risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profilo
- SessionID
- StatusCode

getProfile

Il metodo getProfile consente di recuperare le informazioni temporali e sul profilo relative al visitatore che visita il touchpoint.

```
function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        InteractAPI.getProfile(ssId, callback);

}
```

- **ID sessione** - una stringa che identifica l'ID sessione.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama onSuccess. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama onError.

Valore restituito

Il server di runtime risponde a `getProfile` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- OfferList
- ProfileRecord
- SessionID
- StatusCode

endSession

Il metodo `endSession` contrassegna la fine della sessione di runtime. Quando il server di runtime riceve questo metodo, il server di runtime registra nella cronologia, ripulisce la memoria e così via.

```
function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        InteractAPI.endSession(ssId, callback);

}
```

- **ID sessione** - una stringa univoca che identifica la sessione.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Se il metodo endSession non viene richiamato, le sessioni di runtime vanno in timeout. Il periodo di timeout è configurabile con la proprietà `sessionTimeout`.

Valore restituito

Il server di runtime risponde al metodo `endSession` con l'oggetto `Response` con i seguenti attributi popolati:

- SessionID
- ApiVersion
- OfferList
- Profile
- StatusCode
- AdvisoryMessages

setDebug

Il metodo `setDebug` consente di impostare il livello di dettaglio della registrazione per tutti i percorsi di codice della sessione.

```
function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        InteractAPI.setDebug(ssId, isDebug, callback);

}
```

- **sessionID** - una stringa che identifica l'ID sessione.

- **debug** - un valore booleano che abilita o disabilita le informazioni di debug. I valori validi sono `true` o `false`. Se è true, Interact registra le informazioni di debug nel log del server di runtime.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Valore restituito

Il server di runtime risponde a `setDebug` con un oggetto di risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

getVersion

Il metodo `getVersion` restituisce la versione dell'implementazione corrente del server di runtime Interact.

```
function callGetVersion(commandsToExecute, callback) {

        InteractAPI.getVersion(callback);


}
```

La procedura ottimale è di utilizzare questo metodo quando si inizializza il touchpoint con l'API Interact.

- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Valore restituito

Il server di runtime risponde a `getVersion` con un oggetto risposta con i seguenti attributi popolati:

- AdvisoryMessages
- ApiVersion
- OfferList
- Profile
- SessionID
- StatusCode

executeBatch

Il metodo `executeBatch` consente di eseguire diversi metodi con un'unica richiesta al server di runtime.

```
function callExecuteBatch(commandsToExecute, callback) {

        if (!commandsToExecute)
            return ;
```

```
        InteractAPI.executeBatch(commandsToExecute.ssid,
        commandsToExecute.commands, callback);
}
```

- **ID sessione** - Una stringa che identifica l'ID sessione. Questo ID sessione viene utilizzato per tutti i comandi eseguiti da questa chiamata di metodo.
- **commands** - un array di oggetti comando, uno per ogni comando che si desidera eseguire.
- **callback** - se il metodo ha avuto esito positivo, la funzione di richiamata chiama `onSuccess`. Se il metodo ha avuto esito negativo, la funzione di richiamata chiama `onError`.

Il risultato della chiamata a questo metodo corrisponde a chiamare esplicitamente ogni metodo contenuto nell'array di comando. Questo metodo riduce al minimo il numero di richieste effettive al server di runtime. Il server di runtime esegue ogni metodo in serie; per ogni chiamata, vengono catturati gli eventuali errori o avvisi nell'oggetto risposta che corrisponde a quella chiamata di metodo. Se viene rilevato un errore, executeBatch continua con il resto delle chiamate contenute nel batch. Se l'esecuzione di un metodo ha come risultato un errore, lo stato di livello principale per l'oggetto BatchResponse riflette quell'errore. Se non si sono verificati errori, lo stato di livello principale riflette gli eventuali avvisi che possono essersi verificati. Se non si sono verificati avvisi, lo stato di livello principale riflette un'esecuzione corretta del batch.

Valore restituito

Il server di runtime risponde al `executeBatch` con un oggetto BatchResponse.

Esempio API JavaScript

```
function isJavaScriptAPISelected() {
    var radios = document.getElementsByName('api');
    for (var i = 0, length = radios.length; i < length; i++) {
        if (radios[i].checked) {
            if (radios[i].value === 'JavaScript')
                return true ;
            else  // only one radio can be logically checked
                break;
        }
    }
    return false;
}

function processFormForJSInvocation(e) {

    if (!isJavaScriptAPISelected())
        return;

    if (e.preventDefault) e.preventDefault();

    var serverurl = document.getElementById('serviceUrl').value ;
    InteractAPI.init( { "url" : serverurl } );

    var commandsToExecute = { "ssid" : null, "commands" : [] };
    var callback = InteractAPI.Callback.create(onSuccess, onError);

    callStartSession(commandsToExecute, callback);
    callGetOffers(commandsToExecute, callback);
    callGetOffersForMultipleInteractionPoints(commandsToExecute, callback);
    callPostEvent(commandsToExecute, callback);
    callSetAudience(commandsToExecute, callback);
```

```
    callGetProfile(commandsToExecute, callback);
    callEndSession(commandsToExecute, callback);
    callSetDebug(commandsToExecute, callback);
    callGetVersion(commandsToExecute, callback);

    callExecuteBatch(commandsToExecute, callback);

    // You must return false to prevent the default form behavior
    return false;
}

function callStartSession(commandsToExecute, callback) {

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

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                createStartSessionCmd(
                                icName, getNameValuePairs(audId),
                                audLevel, getNameValuePairs(params),
                                relyOldSs, debug));
    }
    else {
        InteractAPI.startSession(ssId, icName,
                               getNameValuePairs(audId), audLevel,
                               getNameValuePairs(params), relyOldSs,
                               debug, callback) ;
    }

}

function callGetOffers(commandsToExecute, callback) {

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

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                createGetOffersCmd(ip, nofRequested));
    }
    else {
        InteractAPI.getOffers(ssId, ip, nofRequested, callback);
    }
}

function callPostEvent(commandsToExecute, callback) {

    var ssId = document.getElementById('pe_sessionId').value;
```

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

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.
                                    CommandUtil.createPostEventCmd
                                    (ev, getNameValuePairs(params)));
    }
    else {
        InteractAPI.postEvent(ssId, ev, getNameValuePairs(params), callback);
    }
}

function callGetOffersForMultipleInteractionPoints
(commandsToExecute, callback) {

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

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

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

    //build get offer requests
    var getOffReqs = [];
    for(var i = 0; i < parts.length; i += 1) {
        var getofReqObj = parseGetOfferReq(parts[i]);
        if (getofReqObj) {
            getOffReqs.push(getofReqObj);
        }
    }

    if (commandsToExecute && !commandsToExecute.ssid) {
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                    createGetOffersForMultiple
                                    InteractionPointsCmd(getOffReqs));
    }
    else {
        InteractAPI.getOffersForMultipleInteractionPoints
                                    (ssId, getOffReqs, callback);
    }
}

function parseGetOfferReq(ofReqStr) {

    if (!ofReqStr || ofReqStr==="")
        return null;

    var posIp = ofReqStr.indexOf(',');
    var ip = ofReqStr.substring(0,posIp);
    var posNmReq = ofReqStr.indexOf(',', posIp+1);
    var numReq = ofReqStr.substring(posIp+1,posNmReq);
    var posDup = ofReqStr.indexOf(',', posNmReq+1);
    var dupPolicy = null;
```

```
    var reqAttributes = null;

    if (posDup===-1)
        dupPolicy = ofReqStr.substring(posNmReq+1);
    else
        dupPolicy = ofReqStr.substring(posNmReq+1,posDup);

    //check if request string has attributes
    var reqAttrPos = ofReqStr.search(/\(/g);
    if (reqAttrPos!==-1) {
        var reqAttributesStr = ofReqStr.substring(reqAttrPos);
        reqAttributesStr = trimString(reqAttributesStr);
        reqAttributesStr = removeOpenCloseBrackets(reqAttributesStr);
        reqAttributes = parseReqAttributes(reqAttributesStr);
    }

   return InteractAPI.GetOfferRequest.create(ip, parseInt(numReq),
                                parseInt(dupPolicy), reqAttributes);
}

//trim string
function trimString(strToTrim) {
    if (strToTrim)
        return strToTrim.replace(/^\s+|\s+$/g, "");
    else
        return null;
}

function trimStrArray(strArray) {
    if (!strArray) return ;
    for(var i = 0; i < strArray.length; i += 1) {
        strArray[i] = trimString(strArray[i]);
    }
}

//remove open and close brackets in the end
function removeOpenCloseBrackets(strToUpdate) {
    if (strToUpdate)
        return strToUpdate.replace(/^\(+|\)+$/g, "");
    else
        return null;
}

function parseReqAttributes(ofReqAttrStr) {

    //sanitize string
    ofReqAttrStr = trimString(ofReqAttrStr);
    ofReqAttrStr = removeOpenCloseBrackets(ofReqAttrStr);

    if (!ofReqAttrStr || ofReqAttrStr==="")
        return null;

    //get the number requested
    var pos = ofReqAttrStr.indexOf(",");
    var numRequested = ofReqAttrStr.substring(0,pos);
    ofReqAttrStr = ofReqAttrStr.substring(pos+1);

    //first part will be attribute and rest will be child attributes
    var parts = [];
    pos = ofReqAttrStr.indexOf(",");
    if (pos!==-1) {
        parts.push(ofReqAttrStr.substring(0,pos));
        parts.push(ofReqAttrStr.substring(pos+1));
    }
    else {
        parts.push(ofReqAttrStr);
    }
```

```
    for(var i = 0; i < parts.length; i += 1) {
        //sanitize string
        parts[i] = trimString(parts[i]);
        parts[i] = removeOpenCloseBrackets(parts[i]);
        parts[i] = trimString(parts[i]);
    }

    //build list of attributes
    var attributes = [];
    var idx = 0;
    if (parts[0]) {
        var attParts = parts[0].split(";");
        for (idx=0; idx<attParts.length; idx++) {
            attParts[idx] = trimString(attParts[idx]);
            attParts[idx] = removeOpenCloseBrackets(attParts[idx]);
            attParts[idx] = trimString(attParts[idx]);

            var atrObj = parseAttribute(attParts[idx]);
            if (atrObj) attributes.push(atrObj);
        }
    }

    //build list of child attributes
    var childAttributes = [];
    if (parts[1]) {
        var childAttParts = parts[1].split(")");
        for (idx=0; idx<childAttParts.length; idx++) {

            childAttParts[idx] = trimString(childAttParts[idx]);
            childAttParts[idx] = removeOpenCloseBrackets(childAttParts[idx]);
            childAttParts[idx] = trimString(childAttParts[idx]);


            //get the number requested
            var noReqPos = childAttParts[idx].indexOf(",");
            var numReqAt = childAttParts[idx].substring(0,noReqPos);
            childAttParts[idx] = childAttParts[idx].substring(noReqPos+1);
            childAttParts[idx] = trimString(childAttParts[idx]);


            var atrObjParsed = parseAttribute(childAttParts[idx]);
            if (atrObjParsed) {
                var childReq = InteractAPI.OfferAttributeRequirements.create
                (parseInt(numReqAt), [atrObjParsed], null);
                childAttributes.push(childReq);
            }
        }
    }

    return InteractAPI.OfferAttributeRequirements.create(parseInt(numRequested),
    attributes, childAttributes);
}

function parseAttribute(attStr) {

    attStr = trimString(attStr);

    if (!attStr || attStr==="")
        return null;

    var pos1 = attStr.indexOf("=");
    var pos2 = attStr.indexOf("|");
    var nvp = InteractAPI.NameValuePair.create
                          ( attStr.substring(0,pos1),
                            attStr.substring(pos1+1, pos2),
                            attStr.substring(pos2+1));
```

```
        return nvp;
    }
function callSetAudience(commandsToExecute, callback) {
        if (!document.getElementById('checkSetAudience').checked)
            return ;

        var ssId = document.getElementById('sa_sessionId').value;
        var audId = document.getElementById('sa_audienceId').value;
        var audLevel = document.getElementById('sa_audienceLevel').value;
        var params = document.getElementById('sa_parameters').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
                                            createSetAudienceCmd
            (getNameValuePairs(audId), audLevel, getNameValuePairs(params)));
        }
        else {
            InteractAPI.setAudience(ssId, getNameValuePairs(audId),
                                    audLevel, getNameValuePairs(params),
                                    callback);
        }
    }

    function callGetProfile(commandsToExecute, callback) {

        var ssId = document.getElementById('gp_sessionId').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createGetProfileCmd());
        }
        else {
            InteractAPI.getProfile(ssId, callback);
        }
    }

    function callEndSession(commandsToExecute, callback) {

        var ssId = document.getElementById('es_sessionId').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
            commandsToExecute.ssid = ssId;
        }

        if (commandsToExecute && commandsToExecute.commands) {
            commandsToExecute.commands.push(InteractAPI.CommandUtil.
            createEndSessionCmd());
        }
        else {
            InteractAPI.endSession(ssId, callback);
        }
    }

    function callSetDebug(commandsToExecute, callback) {

        var ssId = document.getElementById('sd_sessionId').value;
        var isDebug = document.getElementById('isDebug').value;

        if (commandsToExecute && !commandsToExecute.ssid) {
```

```
        commandsToExecute.ssid = ssId;
    }

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createSetDebugCmd(isDebug));
    }
    else {
        InteractAPI.setDebug(ssId, isDebug, callback);
    }
}

function callGetVersion(commandsToExecute, callback) {

    if (commandsToExecute && commandsToExecute.commands) {
        commandsToExecute.commands.push(InteractAPI.CommandUtil.
        createGetVersionCmd());
    }
    else {
        InteractAPI.getVersion(callback);
    }
}

function callExecuteBatch(commandsToExecute, callback) {

    if (!commandsToExecute)
        return ;

    InteractAPI.executeBatch(commandsToExecute.ssid,
    commandsToExecute.commands, callback);
}

function getNameValuePairs(parameters) {

    if (parameters === '')
        return null;

    var parts = parameters.split(';');
    var nvpArray = new Array(parts.length);

    for(i = 0; i < parts.length; i += 1) {
        var nvp = parts[i].split(',') ;
        var value = null;
        if (nvp[2]===InteractAPI.NameValuePair.prototype.TypeEnum.NUMERIC) {
            if (isNaN(nvp[1])) {
                value = nvp[1];  //a non number was provided as number,
                pass it to API as it is
            }
            else {
                value = Number(nvp[1]);
            }
        }
        else {
            value = nvp[1];
        }
        //special handling NULL value
        if (value && typeof value === 'string') {
            if (value.toUpperCase() === 'NULL') {
                value = null;
            }
        }
        nvpArray[i] = InteractAPI.NameValuePair.create(nvp[0], value, nvp[2]) ;
    }

    return nvpArray;
}
```

```
function showResponse(textDisplay) {
    var newWin = open('','Response','height=300,width=300,titlebar=no,
    scrollbars=yes,toolbar=no,
   resizable=yes,menubar=no,location=no,status=no');

    if (newWin.locationbar !== 'undefined' && newWin.locationbar
    && newWin.locationbar.visible)
        newWin.locationbar.visible = false;

    var displayHTML = '<META HTTP-EQUIV="Content-Type"
    CONTENT="text/html; charset=UTF-8">
    <html><head><style>TD { border-width : thin; border-style : solid }</style.'
                        + "<script language='Javascript'>"
                        + "var desiredDomain = 'unicacorp.com'; "
                        + "if (location.href.indexOf(desiredDomain)>=0) "
                        + "{ document.domain = desiredDomain;} "
                        + "</script></head><body> "
                        + textDisplay
                        + "</body></html>" ;
    newWin.document.body.innerHTML = displayHTML;
    newWin.focus() ;
}

function onSuccess(response) {
    showResponse("********Response********<br> " + JSON.stringify(response)) ;
}

function onError(response) {
    showResponse("********Error********<br> " + response) ;
}

function formatResoponse(response) {

}

function printBatchResponse(batResponse) {

}

function printResponse(response) {

}
```

Esempio di oggetto JavaScript risposta onSuccesss

Questo esempio mostra le tre variabili per l'oggetto JavaScript risposta; offerLists, messages e profile.

`offerList` restituisce un elenco non null se si chiama `getOffer` o `getOffersForMultipleInteractionPoints` come API o come parte dei comandi batch. È opportuno verificare sempre i valori null prima di eseguire operazioni su questa variabile.

È opportuno verificare sempre lo stato della risposta JavaScript `messages`.

`Profile` viene restituito come non null se si utilizza `getProfile` come API o come parte dei comandi batch. Se non si utilizza `getProfile`, è possibile ignorare questa variabile. È opportuno verificare sempre i valori null prima di eseguire operazioni su questa variabile.

```
function onSuccess(response)
InteractAPI.ResponseTransUtil._buildResponse = function(response) {
        'use strict';
```

```
    if (!response) return null;

    var offerList = null;
    //transform offerLists to JS Objects
    if (response.offerLists) {
        offerList = [];
        for (var ofListCt=0; ofListCt<response.offerLists.length;ofListCt++) {
            var ofListObj = this._buildOfferList(response.offerLists[ofListCt]);
            if (ofListObj) offerList.push(ofListObj);
        }
    }

    var messages = null;
    //transform messages to JS Objects
    if (response.messages) {
        messages = [];
        for (var msgCt=0; msgCt<response.messages.length;msgCt++) {
            var msgObj = this._buildAdvisoryMessage(response.messages[msgCt]);
            if (msgObj) messages.push(msgObj);
        }
    }

    var profile = null;
    //transform profile nvps to JS Objects
    if (response.profile) {
        profile = [];
        for (var nvpCt=0; nvpCt<response.profile.length;nvpCt++) {
            var nvpObj = this._buildNameValuePair(response.profile[nvpCt]);
            if (nvpObj) profile.push(nvpObj);
        }
    }

    return InteractAPI.Response.create(response.sessionId,
                                       response.statusCode, offerList,
                                       profile, response.version,
                                       messages) ;
};
```

Capitolo 9. Informazioni sull'API ExternalCallout

Interact offre una macro estendibile, EXTERNALCALLOUT, da utilizzare con i diagrammi di flusso interattivi. Questa macro consente di eseguire una logica personalizzata per comunicare con i sistemi esterni durante le esecuzioni del diagramma di flusso. Ad esempio, se si desidera calcolare il punteggio di credito di un cliente durante un'esecuzione del diagramma di flusso, è possibile creare una classe Java (un callout) per effettuare questa operazione e quindi utilizzare la macro EXTERNALCALLOUT in un processo Seleziona nel diagramma di flusso interattivo per ottenere il punteggio di credito dal callout.

La configurazione di EXTERNALCALLOUT è costituita da due step principali. Innanzitutto è necessario creare una classe Java che implementi l'API ExternalCallout. In secondo luogo è necessario configurare le proprietà di configurazione Marketing Platform necessarie sul server di runtime nella categoria Interact | flowchart | ExternalCallouts.

Oltre alle informazioni contenute in questa sezione, è disponibile il JavaDoc per l'API ExternalCallout su qualsiasi server di runtime Interact nella directory Interact/docs/externalCalloutJavaDoc.

Interfaccia IAffiniumExternalCallout

L'API ExternalCallout è contenuta nell'interfaccia IAffiniumExternalCallout. È necessario implementare l'interfaccia IAffiniumExternalCallout per utilizzare la macro EXTERNALCALLOUT.

La classe che implementa IAffiniumExternalCallout deve disporre di un costruttore con cui può essere inizializzata dal server di runtime.

- Se non sono presenti costruttori nella classe, il programma di compilazione Java crea un costruttore predefinito e questo è sufficiente.
- Se sono presenti costruttori con argomenti, deve essere fornito un costruttore pubblico senza argomenti, che verrà utilizzato dal server di runtime.

Quando si sviluppa il callout esterno, tenere presente quanto segue:

- Ogni valutazione di espressione con un callout esterno crea una nuova istanza della classe. È necessario gestire i problemi di sicurezza dei thread per i membri statici nella classe.
- Se il callout esterno utilizza le risorse di sistema, come i file o una connessione database, è necessario gestire le connessioni. Il server di runtime non dispone di una funzione per ripulire le connessioni automaticamente.

È necessario compilare l'implementazione con interact_externalcallout.jar che si trova nella directory lib dell'installazione dell'ambiente di runtime di IBM Interact.

IAffiniumExternalCallout abilita il server di runtime alla richiesta di dati dalla classe Java. L'interfaccia è costituita da quattro metodi:

- getNumberOfArguments
- getValue
- initialize

- shutdown

Aggiunta di un servizio web da utilizzare con la macro EXTERNALCALLOUT

Utilizzare questa procedura per aggiungere un servizio web da utilizzare con la macro EXTERNALCALLOUT. La macro EXTERNALCALLOUT riconosce i callout solo se vengono definite le proprietà di configurazione appropriate.

Procedura

In Marketing Platform per l'ambiente di runtime, aggiungere o definire le seguenti proprietà di configurazione nella categoria Interact > flowchart > externalCallouts.

Proprietà di configurazione	Impostazione
Categoria externalCallouts	Creare una categoria per il callout esterno
class	I nomi classe per il callout esterno
classpath	Il percorso classi dei file di classe del callout esterno
Categoria Parameter Data	Se il callout esterno richiede dei parametri, creare nuove proprietà di configurazione dei parametri ed assegnare a ognuna un valore

getNumberOfArguments

Il metodo getNumberOfArguments restituisce il numero di argomenti previsti dalla classe Java con cui si sta effettuando l'integrazione.

```
getNumberOfArguments()
```

Valore restituito

Il metodo getNumberOfArguments restituisce un numero intero.

Esempio

Il seguente esempio visualizza la stampa del numero di argomenti.

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

Il metodo getValue esegue la funzionalità principale del callout e restituisce i risultati.

```
getValue(audienceID, configData, arguments)
```

Il metodo getValue richiede i seguenti parametri:

- **audienceID** - un valore che identifica l'ID destinatario.
- **configData** - un'associazione con le coppie chiave-valore dei dati di configurazione richiesti dal callout.
- **arguments** - gli argomenti richiesti dal callout. Ogni argomento può essere una stringa (String), un valore doppio (Double), una data (Date) o un elenco (List) di uno di questi. Un argomento List può contenere valori null, tuttavia non può contenere, ad esempio, una stringa o un valore doppio.

Il controllo del tipo di argomento deve essere eseguito all'interno dell'implementazione.

Se il metodo `getValue` per qualche motivo ha esito negativo, restituisce `CalloutException`.

Valore restituito

Il metodo getValue restituisce un elenco di stringhe.

Esempio

```
public List<String> getValue(AudienceId audienceId, Map<String,
  String> configurationData, Object... arguments) throws CalloutException
{
 Long customerId = (Long) audienceId.getComponentValue("Customer");
 // eseguire la query di scoreQueryUtility per il punteggio di credito di customerId
 Double score = scoreQueryUtility.query(customerId);
 String str = Double.toString(score);
 List<String> list = new LinkedList<String>();
 list.add(str);
 return list;
}
```

initialize

Il metodo `initialize` viene richiamato una volta quando viene avviato il server di runtime. Se vi sono operazioni che possono ostacolare le prestazioni durante il runtime, come il caricamento di una tabella database,devono essere eseguite da questo metodo.

```
initialize(configData)
```

Il metodo `initialize` richiede il seguente parametro:

- **configData** - un'associazione con le coppie chiave-valore dei dati di configurazione richiesti dal callout.

  Interact legge questi valori dai parametri del callout esterno definiti nella categoria `Interact > Flowchart > External Callouts > [External Callout] > Parameter Data`.

Se il metodo `initialize` per qualche motivo ha esito negativo, restituisce `CalloutException`.

Valore restituito

Nessuno.

Esempio

```
public void initialize(Map<String, String> configurationData) throws CalloutException
{
    // configurationData contiene le coppie chiave-valore specifiche per l'ambiente
    // in cui è in esecuzione il server
    // inizializzare scoreQueryUtility qui
}
```

shutdown

Il metodo `shutdown` viene richiamato una sola volta quando il server di runtime si chiude. Se sono richieste attività di ripulitura dal callout, devono essere eseguite in questo momento.

```
shutdown(configData)
```

Il metodo shutdown richiede il seguente parametro:

- **configData**-un'associazione con le coppie chiave-valore dei dati di configurazione richiesti dal callout.

Se il metodo shutdown per qualche motivo ha esito negativo, restituisce CalloutException.

Valore restituito

Nessuno.

Esempio

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
    // chiudere scoreQueryUtility qui
}
```

Esempio di API ExternalCallout

Questo esempio crea un callout esterno che ottiene un punteggio di credito.

Creare un callout esterno che ottiene un punteggio credito:

1. Creare un file denominato GetCreditScore.java con il contenuto riportato di seguito. Questo file presume che sia presente una classe denominata ScoreQueryUtility che estrae un punteggio da un'applicazione di modeling.

```
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 classe che contiene la logica per eseguire la query di un sistema esterno per il punteggio di credito di un cliente
 private static ScoreQueryUtility scoreQueryUtility;
 public void initialize(Map<String, String> configurationData) throws CalloutException
 {
  // configurationData contiene le coppie chiave-valore specifiche per l'ambiente in cui è in esecuzione il server
  // inizializzare scoreQueryUtility qui
 }

 public void shutdown(Map<String, String> configurationData) throws CalloutException
 {
 // chiudere scoreQueryUtility qui
 }

 public int getNumberOfArguments()
 {
  // non prevedere altri argomenti oltre all'id del cliente
  return 0;
 }

 public List<String> getValue(AudienceId audienceId, Map<String, String> configurationData,
  Object... arguments) throws CalloutException
 {
  Long customerId = (Long) audienceId.getComponentValue("Customer");
  // eseguire la query di scoreQueryUtility per il punteggio di credito di customerId
  Double score = scoreQueryUtility.query(customerId);
  String str = Double.toString(score);
  List<String> list = new LinkedList<String>();
```

```
  list.add(str);
  return list;
 }
}
```

2. Compilare `GetCreditScore.java` in `GetCreditScore.class`.
3. Creare un file JAR denominato `creditscore.jar` che contiene `GetCreditScore.class` e gli altri file di classe che utilizza.
4. Copiare il file JAR in un'ubicazione sul server di runtime, ad esempio `/data/interact/creditscore.jar`.
5. Creare un callout esterno con nome `GetCreditScore` e percorso classi `/data/interact/creditscore.jar` nella categoria `externalCallouts` sulla pagina Gestisci configurazioni.
6. In un diagramma di flusso interattivo il callout può essere utilizzato come `EXTERNALCALLOUT('GetCreditScore')`.

Interfaccia iInteractProfileDataService

L'API Profile Data Services è contenuta nell'interfaccia `iInteractProfileDataService`. Questa interfaccia consente di importare i dati gerarchici in una sessione Interact tramite una o più origini dati esterne (ad esempio un file flat, un servizio web, e così via) quando viene avviata la sessione Interact o l'ID destinatario di una sessione Interact cambia.

Per sviluppare l'importazione di dati gerarchici utilizzando l'API Profile Data Services, è necessario scrivere una classe Java che estrae le informazioni da qualsiasi origine dati e li associa ad un oggetto `ISessionDataRootNode`, per poi fare riferimento a quei dati associati utilizzando la macro EXTERNALCALLOUT in un processo Seleziona di un diagramma di flusso interattivo.

È necessario compilare l'implementazione con `interact_externalcallout.jar` che si trova nella directory `lib` dell'installazione dell'ambiente di runtime di IBM Interact.

Per una serie completa della documentazione Javadoc sull'utilizzo di questa interfaccia, visualizzare i file in *Interact_home*`/docs/externalCalloutJavaDoc` con qualsiasi browser web.

Per un esempio di implementazione su come utilizzare Profile Data Service, incluse le descrizioni di commento su come l'esempio sia stato implementato, vedere *Interact_home*`/samples/externalcallout/XMLProfileDataService.java`.

Nota: l'esempio di implementazione è progettato per essere utilizzato unicamente come esempio. Non utilizzare questo esempio nella propria implementazione.

Aggiunta di un'origine dati da utilizzare con Profile Data Services

Utilizzare questa procedura per aggiungere un'origine dati da utilizzare con Profile Data Services.

Informazioni su questa attività

La macro `EXTERNALCALLOUT` riconosce un'origine dati per l'importazione di dati gerarchici di Profile Data Services solo se sono state definite le proprietà di configurazione appropriate.

Procedura

In Marketing Platform per l'ambiente di runtime, aggiungere o definire le seguenti proprietà di configurazione nella categoria `Interact > profile > Audience Levels > [AudienceLevelName] > Profile Data Services`.

Proprietà di configurazione	Impostazione
Categoria `Nuovo nome categoria`	Il nome dell'origine dati che si sta definendo. Il nome immesso qui deve essere univoco tra le origini dati per lo stesso livello destinatario.
`enabled`	Indica se l'origine dati è abilitata per il livello destinatario in cui è definita.
`className`	Il nome completo della classe origine dati che implementa `IInteractProfileDataService`
`classPath`	Il percorso classi dei file di classe di Profile Data Services. Se non si specifica questa proprietà, verrà utilizzato per impostazione predefinita il percorso classi del server delle applicazioni contenitore.
Categoria `priority`	La priorità di questa origine dati nell'ambito di questo livello destinatario. Deve trattarsi di un valore univoco tra tutte le origini dati per ogni livello destinatario. (Ossia, se priority è impostata su 100 per un'origine dati, nessun'altra origine dati nello stesso livello destinatario potrà avere un valore priority di 100).

Interfaccia IParameterizableCallout

L'API Callout parametrizzabile è contenuta nell'interfaccia `IParameterizableCallout`.

Questa interfaccia è l'interfaccia di base delle interfacce API esposte che può accettare i parametri dalla configurazione tramite Marketing Platform. Poiché si tratta di un'interfaccia di base, non deve essere implementata direttamente. I parametri vengono richiamati dai nodi figlio del nodo `Parameter Data` nella categoria che fa riferimento a questa implementazione. Nel seguente esempio, ESB è un'implementazione personalizzata del servizio dati del profilo, che a sua volta implementa l'interfaccia `IParameterizableCallout`. I parametri endPoint e login, insieme con i relativi valori, sono trasmessi in questa classe di implementazione quando il motore Interact tenta di inizializzarla e terminarla.

```
Profile Data Services
...ESB
   ...Parameter Data
      ...endPoint
      ...login
```

L'interfaccia è costituita da due metodi:

- `initialize`
- `shutdown`

initialize

Il metodo `initialize` inizializza questa classe di implementazione.

```
void initialize(java.util.Map<java.lang.String,java.lang.String> configurationData)
               throws CalloutException
```

Il metodo `initialize` richiede il seguente parametro:

- **configurationData** - un'associazione con coppie nome/valore di parametri configurati dagli utenti

Throws

`CalloutException`

shutdown

Il metodo `shutdown` chiude questa classe di implementazione.

```
void shutdown(java.util.Map<java.lang.String,java.lang.String> configurationData)
              throws CalloutException
```

Il metodo `shutdown` richiede il seguente parametro:

- **configurationData** - un'associazione con coppie nome/valore di parametri configurati dagli utenti

Throws

`CalloutException`

Interfaccia ITriggeredMessageAction

L'API Azione messaggio attivato è contenuta nell'interfaccia `ITriggeredMessageAction`. Questa interfaccia consente di ottenere e impostare il nome di questa istanza.

L'interfaccia `ITriggeredMessageAction` serve da interfaccia di base per altre interfacce e non deve mai essere implementata direttamente.

L'interfaccia è costituita da due metodi:

- `getName`
- `setName`

getName

Il metodo `getName` restituisce il nome dell'istanza `ITriggeredMessageAction`.

```
java.lang.String getName()
```

setName

Il metodo `setName` imposta il nome dell'istanza `ITriggeredMessageAction`.

```
void setName(java.lang.String name)
```

Quando si inizializza la classe di implementazione di questa interfaccia, Interact imposta il nome dell'interfaccia con il nome specificato nella IU di configurazione.

Nel seguente esempio, il nome di questo gateway è `InteractLog`.

```
triggeredMessage
           ...gateways
              ...InteractLog
```

Il metodo `setName` richiede il seguente parametro:

- name - il nome che si desidera impostare per l'istanza `ITriggeredMessageAction`.

Interfaccia IChannelSelector

L'API Selettore canale è contenuta nell'interfaccia `IChannelSelector`. Questa interfaccia consente di selezionare i canali in uscita in base all'offerta da inviare e agli attributi della sessione.

Per un esempio di implementazione su come utilizzare Azione messaggio attivato, incluse le descrizioni di commento su come l'esempio sia stato implementato, vedere *Interact_home*`/samples/triggeredmessage/SampleChannelSelector.java`.

Nota: l'esempio di implementazione è progettato per essere utilizzato unicamente come esempio. Non utilizzare questo esempio nella propria implementazione.

Si consiglia di provare ad utilizzare questa implementazione invece di scriverne di propria.

L'interfaccia è costituita da un metodo:

- `selectChannels`

selectChannels

Il metodo `selectChannels` seleziona i canali in uscita cui l'offerta passata deve essere inviata con l'interfaccia `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 tenta di inviare questa offerta a tutti quei canali restituiti.

Il metodo `selectChannels` richiede i seguenti parametri:

- **availableChannels** – una mappa di canali in uscita disponibili, che sono configurati nella IU Messaggio attivato, nelle impostazioni della fase di progettazione di Interact. In ciascuna voce della mappa, la chiave è il nome del canale e il valore sono i parametri configurati per quel canale nella fase di progettazione di Interact. L'ordine di iterazione di questa mappa corrisponde all'ordine definito su tale IU. Se viene utilizzato Canale preferito cliente nella IU Messaggio attivato, viene sostituito dal canale reale prima che questo metodo venga richiamato. Inoltre, se lo stesso canale compare più volte nella IU, viene conservata soltanto la ricorrenza con la priorità più alta e tutti i duplicati vengono rimossi.
- **offer** - l'offerta da consegnare
- **sessionData** - gli attributi attualmente memorizzati nella sessione Interact associata

Interfaccia IDispatcher

L'API Dispatcher è contenuta nell'interfaccia `IDispatcher`. Questa interfaccia invia offerte da gateway di destinazione.

Poiché esiste solo un'istanza di questa classe per ciascun dispatcher configurato, l'implementazione di questa interfaccia deve essere senza stato dalla prospettiva di Interact.

Per un esempio di implementazione su come utilizzare Azione messaggio attivato, incluse le descrizioni di commento su come l'esempio sia stato implementato, vedere *Interact_home*/samples/triggeredmessage/SampleDispatcher.java.

Nota: l'esempio di implementazione è progettato per essere utilizzato unicamente come esempio. Non utilizzare questo esempio nella propria implementazione.

Si consiglia di provare ad utilizzare questa implementazione invece di scriverne di propria.

L'interfaccia è costituita da un metodo:

- dispatch

dispatch

Il metodo dispatch invia le offerte ai gateway di destinazione nell'interfaccia 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
```

Quando i canali in uscita vengono selezionati per un'offerta candidata, Interact tenta di inviare le offerte candidate ai gestori associati al canale. I gestori vengono tentati in base alle priorità definite dall'alto in basso. Per ogni gestore, Interact richiama questo metodo del dispatcher configurato. La modalità di instradamento dell'offerta al gateway di destinazione, che è configurato nello stesso gestore, dipende dall'implementazione di questa istanza del dispatcher. Se sono presenti più offerte inviate allo stesso gestore come risultato della stessa valutazione del messaggio attivato, Interact tenta di inviare tutte le offerte in un batch.

Il metodo dispatch richiede i seguenti parametri:

- **channel** - il canale in uscita cui vengono inviate queste offerte
- **gatewayName** - il nome del gateway di destinazione
- **offers** – le offerte da inviare al gateway in un batch
- **profileData** – gli attributi del profilo vengono popolati da IGateway.validate e vengono passati a IGateway.deliver

Valore restituito

Il metodo dispatch restituisce se la distribuzione ha avuto esito positivo o negativo

Throws

com.unicacorp.interact.exceptions.InteractException

Interfaccia IGateway

L'API Gateway è contenuta nell'interfaccia IGateway. Questa interfaccia riceve offerte da Interact e invia le offerte alla relativa destinazione.

Ogni implementazione di questa interfaccia comunica con una determinata destinazione. La destinazione deve eseguire la trasformazione dei dati necessaria, la popolazione degli attributi e altro lavoro correlato alla destinazione simile.

Per un esempio di implementazione su come utilizzare Azione messaggio attivato, incluse le descrizioni di commento su come l'esempio sia stato implementato, vedere *Interact_home*/samples/triggeredmessage/SampleOutboundGateway.java.

Nota: l'esempio di implementazione è progettato per essere utilizzato unicamente come esempio. Non utilizzare questo esempio nella propria implementazione.

L'interfaccia è costituita da due metodi:

- deliver
- validate

deliver

Il metodo deliver viene richiamato per inviare l'offerta o le offerte a una destinazione nell'interfaccia IGateway.

```
void deliver(java.util.Collection<com.unicacorp.interact.api.Offer> offers,
                    com.unicacorp.interact.api.NameValuePair[] profileData,
                    java.lang.String channel)
```

Il metodo deliver richiede i seguenti parametri:

- **offers** – l'offerta da inviare
- **profileData** – gli attributi del profilo che il metodo di convalida popola in parameterMap
- **channel** - il canale in uscita cui verranno inviate queste offerte

validate

Il metodo validate convalida le offerte candidate nell'interfaccia 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)
```

Il motore Interact richiama questo metodo per convalidare le offerte candidate. L'implementazione di questo metodo controlla le offerte, gli attributi offerta e gli attributi sessione rispetto ai requisiti della destinazione per determinare l'offerta o le offerte che possono essere inviate attraverso questo gateway. Inoltre, può aggiungere i parametri necessari nella mappa passata, che viene ritrasmessa al metodo di consegna.

Il metodo validate richiede i seguenti parametri:

- **sessionData** - gli attributi attualmente memorizzati nella sessione Interact associata
- **candidateOffers** - le offerte che Interact ha selezionato in base al metodo di selezione dell'offerta, ai relativi parametri e ad altri fattori. Tali offerte sono idonee per essere consegnate dalla prospettiva di Interact, ma comunque soggette al gateway.
- **parameterMap** – una mappa che l'implementazione di questo metodo deve utilizzare per passare i parametri al metodo di consegna
- **channel** - il canale in uscita cui verranno inviate queste offerte

Capitolo 10. Programma di utilità di IBM Interact

Questa sezione descrive i programmi di utilità di amministrazione disponibili con Interact.

Programma di utilità per l'esecuzione della distribuzione (runDeployment.sh/.bat)

Lo strumento della riga di comando `runDeployment` consente di distribuire un canale interattivo per un gruppo di server specifico dalla riga di comando, utilizzando le impostazioni fornite da un file `deployment.properties` che indica tutti i parametri possibili ed è disponibile nella stessa ubicazione dello strumento `runDeployment` stesso. La capacità di eseguire la distribuzione di un canale interattivo dalla riga di comando è soprattutto utile quando si utilizza la funzione `OffersBySQL`. Ad esempio, è possibile configurare un diagramma di flusso del batch Campaign per l'esecuzione su base periodica. Quando l'esecuzione del diagramma di flusso viene completata, è possibile richiamare un trigger per inizializzare la distribuzione delle offerte nella tabella `OffersBySQL` utilizzando questo strumento della riga di comando.

Descrizione

È possibile trovare lo strumento della riga di comando `runDeployment` installato automaticamente sul server Interact Design Time, nell'ubicazione seguente:

Interact_home`/interactDT/tools/deployment/runDeployment.sh` (o `runDeployment.bat` su un server Windows)

Il solo argomento inviato al comando è l'ubicazione di un file denominato `deployment.properties` che descrive tutti i possibili parametri necessari per distribuire la combinazione canale interattivo/gruppo di server di runtime. Viene fornito un file di esempio come riferimento.

Nota: prima di utilizzare il programma di utilità `runDeployment`, è necessario prima modificarlo con un editor di testo per fornire il percorso dell'ambiente di runtime Java sul server. Ad esempio, è possibile specificare *Interact_home*`/jre` o *Platform_home*`/jre` come percorso, se una di queste directory contiene il runtime Java che si desidera che il programma di utilità utilizzi. In alternativa, è possibile fornire il percorso per ogni ambiente di runtime Java supportato per l'utilizzo con questa release dei prodotti IBM .

Utilizzo del programma di utilità runDeployment in un ambiente protetto (SSL)

Per utilizzare il programma di utilità runDeployment quando la sicurezza è stata abilitata sul server Interact (quindi la connessione è su una porta SSL), è necessario aggiungere la proprietà Java truststore nel modo seguente:

1. Quando si modifica il file `deployment.properties` per la distribuzione del canale interattivo, modificare la proprietà `deploymentURL` per utilizzare l'URL SSL protetto, come in questo esempio:

   `deploymentURL=https://<HOST>.<DOMAIN>:<PORT>/Campaign/interact/InvokeDeploymentServlet`

2. Modificare lo script runDeployment.sh o runDeployment.bat utilizzando un qualsiasi editor di testo per aggiungere il seguente argomento alla riga che inizia con ${JAVA_HOME}:

   -Djavax.net.ssl.trustStore=<*TrustStorePath*>

   Ad esempio, la riga potrebbe avere questo aspetto dopo aver aggiunto l'argomento truststore:

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

   Sostituire <*TrustStorePath*> con il percorso al truststore SSL reale.

Esecuzione del programma di utilità

Dopo aver modificato il programma di utilità per fornire l'ambiente di runtime Java e aver personalizzato una copia del file deployment.properties in modo che corrispondano al proprio ambiente, è possibile eseguire il programma di utilità con questo comando:

Interact_home/interactDT/tools/deployment/runDeployment.sh *deployment.properties*

Sostituire *Interact_home* con il valore reale dell'installazione di Interact Design Time e sostituire *deployment.properties* con il percorso reale e il nome del file delle proprietà personalizzato per questa distribuzione.

File deployment.properties di esempio

Il file di esempio deployment.properties contiene un elenco commentato di tutti i parametri che è necessario personalizzare per la corrispondenza al proprio ambiente. Il file di esempio contiene inoltre commenti che illustrano la finalità di ogni parametro e il motivo per cui potrebbe essere necessario personalizzare un particolare valore.

```
###############################################################################
#
# Le seguenti proprietà sono utilizzate dal programma InvokeDeploymentClient.
# Il programma cercherà una impostazione deploymentURL.  Il programma invierà una
# richiesta su tale url; tutte le altre impostazioni sono inviate come parametri in
# tale richiesta. Il programma poi controlla lo stato della distribuzione e
# restituisce un valore quando la distribuzione è allo stato terminale (o se
# è stato raggiunto il waitTime specificato).
#
# l'output del programma sarà nel seguente formato:
# <STATE> : <Misc Detail>
#
# dove state può essere uno dei seguenti:
# ERROR
# RUNNING
# SUCCESS
#
# Misc Detail sono i dati che popolano l'area messaggi di stato
# nella gui di distribuzione della pagina di riepilogo IC. NOTA: possono essere presenti tag HTML
# in Misc Detail
#
###############################################################################

###############################################################################
# deploymentURL: l'url al servlet InvokeDeployment che risiede in Interact
# Design Time.  Deve essere nel formato seguente:
# http://dt_host:port/Campaign/interact/InvokeDeploymentServlet
###############################################################################
```

```
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet

################################################################################
# dtLogin: il login utilizzato per accedere a Design Time se
# se si è distribuito IC tramite la gui di distribuzione nella
# pagina di riepilogo IC.
################################################################################
dtLogin=asm_admin

################################################################################
# dtPW: la PW per dtLogin
################################################################################
dtPW=

################################################################################
# icName: il nome del canale interattivo che si desidera distribuire
################################################################################
icName=ic1

################################################################################
# partition: il nome della partizione
################################################################################
partition=partition1

################################################################################
# request: il tipo di richiesta che si desidera questo strumento esegua,
# ci sono due comportamenti.  Se il valore è "deploy",la distribuzione
# viene eseguita.  Tutti gli altri valori fanno sì che lo strumento restituisca semplicemente
# lo stato dell'ultima distribuzione dell'IC specificato.
################################################################################
request=deploy

################################################################################
# serverGroup: il nome del gruppo di server cui si desidera
# distribuire l'IC.
################################################################################
serverGroup=defaultServerGroup

################################################################################
# serverGroupType: indica se la distribuzione è o meno per
# un gruppo di server di produzione o per un gruppo di server di test. 1 indica produzione
# 2 indica test.
################################################################################
serverGroupType=1

################################################################################
# rtLogin: l'account utilizzato per autenticarsi con il gruppo di server
# cui si sta distribuendo.
################################################################################
rtLogin=asm_admin

################################################################################
# rtPW: la password associata a rtLogin
################################################################################
rtPW=

################################################################################
# waitTime: dopo che lo strumento ha inoltrato la richiesta di distribuzione
# ne controlla lo stato.  Se la distribuzione non è stata completata (o
# non è riuscita), lo strumento continua a eseguire il polling del sistema per lo stato finché
# non si raggiunge lo stato completo OPPURE il waitTime specificato (in
# secondi).
################################################################################
waitTime=5

################################################################################
# pollTime: se lo stato di una distribuzione è ancora in esecuzione,
```

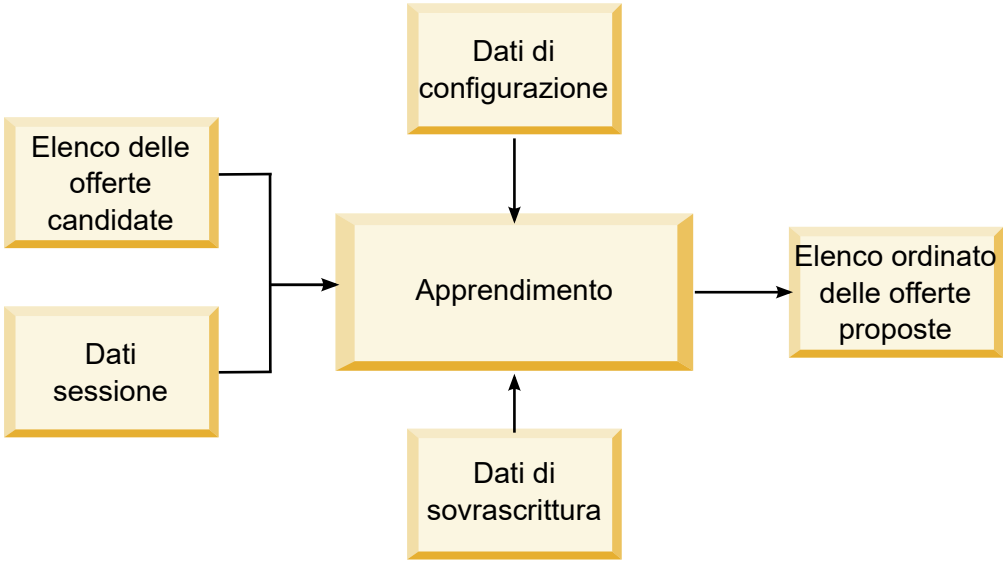
```
# lo strumento continua a controllare lo stato.  Sarà inattivo nei controlli
# di stato intermedi per un numero di secondi che dipende dall'impostazione pollTime.
################################################################################
pollTime=3

################################################################################
# global: se si imposta su false lo strumento NON distribuisce le impostazioni globali.
# la non disponibilità della proprietà distribuirà comunque le impostazioni globali.
################################################################################
global=true
```

Capitolo 11. Informazioni sull'API di apprendimento

Interact fornisce un modulo di apprendimento che utilizza un algoritmo naive-bayesiano per monitorare le azioni del visitatore e propone offerte ottimali (in termini di accettazione). È possibile implementare la stessa interfaccia Java con algoritmi propri utilizzando l'API di apprendimento per creare un proprio modulo di apprendimento.

Nota: se si utilizza l'apprendimento esterno, i report dimostrativi relativi all'apprendimento (report Dettagli di apprendimento dell'offerta interattiva e Analisi accuratezza segmento interattivo) non restituiscono dati validi.

Al livello più semplice, l'API di apprendimento fornisce i metodi per raccogliere i dati dall'ambiente di runtime e restituire un elenco ordinato di offerte consigliate.



È possibile raccogliere i seguenti dati da Interact

- Dati di contatto dell'offerta
- Dati di accettazione dell'offerta
- Tutti i dati di sessione
- Dati dell'offerta specifici per Campaign
- Proprietà di configurazione definite nella categoria `learning` per l'ambiente di progettazione e la categoria `offerserving` per l'ambiente di runtime

È possibile utilizzare questi dati negli algoritmi per creare un elenco di offerte proposte. È possibile restituire un elenco delle offerte consigliate, in ordine decrescente di raccomandazione.

Sebbene non sia visualizzato nel diagramma, è possibile utilizzare l'API di apprendimento anche per raccogliere i dati per l'implementazione di apprendimento. È possibile conservare questi dati in memoria o registrarli in un file o database per un'ulteriore analisi.

Dopo aver creato le classi Java, è possibile convertirle in file jar. Una volta creati i file jar, è necessario anche configurare l'ambiente di runtime per riconoscere il

modulo di apprendimento esterno modificando le proprietà di configurazione. È necessario copiare le classi Java o i file jar in ogni server di runtime che utilizza il modulo di apprendimento esterno.

Oltre alle informazioni contenute in questa sezione, su ogni server di runtime nella directory `Interact/docs/learningOptimizerJavaDoc` è disponibile il JavaDoc per l'API optimizer di apprendimento.

È necessario compilare l'implementazione con `interact_learning.jar` che si trova nella directory `lib` dell'installazione dell'ambiente di runtime di Interact.

Quando si scrive l'implementazione di apprendimento personalizzata, tenere presenti le linee guida riportate di seguito.

- Le prestazioni sono critiche.
- Deve funzionare con multi-threading e deve essere thread-safe.
- Deve gestire tutte le risorse esterne con in mente le modalità di errore e le prestazioni.
- Utilizzare in modo appropriato le eccezioni, la registrazione (log4j) e la memoria.

Configurazione dell'ambiente di runtime per riconoscere moduli di apprendimento esterno

È possibile utilizzare l'API di apprendimento Java per scrivere un proprio modulo di apprendimento. È necessario configurare l'ambiente di runtime per riconoscere il proprio programma di utilità di apprendimento in Marketing Platform.

Informazioni su questa attività

È necessario riavviare il server di runtime di Interact perché queste modifiche diventino effettive.

Procedura

1. In Marketing Platform per l'ambiente di runtime, modificare le seguenti proprietà di configurazione nella categoria `Interact > offerserving`. Le proprietà di configurazione per l'API optimizer di apprendimento sono presenti nella categoria `Interact > offerserving > External Learning Config`.

Proprietà di configurazione	Impostazione
`optimizationType`	**ExternalLearning**
`externalLearningClass`	Il nome classe dell'apprendimento esterno
`externalLearningClassPath`	Il percorso dei file di classe o JAR sul server di runtime per l'apprendimento esterno. Se si utilizza un gruppo di server e tutti i server di runtime fanno riferimento alla stessa istanza di Marketing Platform, ogni server deve avere una copia dei file di classe o JAR nella stessa ubicazione.

2. Riavviare il server di runtime Interact perché queste modifiche diventino effettive.

Interfaccia ILearning

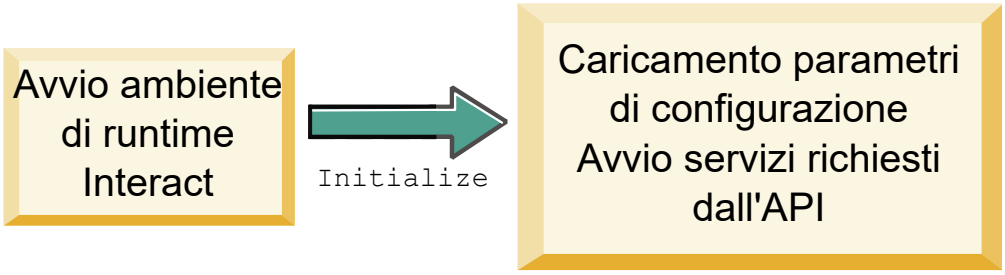
L'API di apprendimento è creata intorno all'interfaccia `ILearning`. È necessario implementare l'interfaccia `ILearning` per supportare la logica personalizzata del proprio modulo di apprendimento.

Tra le altre cose, l'interfaccia `ILearning` consente all'utente di raccogliere i dati dall'ambiente di runtime per la classe Java e di inviare un elenco delle offerte consigliate al server di runtime.

initialize

Il metodo `initialize` viene richiamato una volta quando viene avviato il server di runtime. Se vi sono operazioni che non devono essere ripetute ma che possono ostacolare le prestazioni durante il runtime, come il caricamento di dati statici da una tabella di database, devono essere eseguite da questo metodo.

```
initialize(ILearningConfig config, boolean debug)
```



- **config** - un oggetto `ILearningConfig` definisce tutte le proprietà di configurazione pertinenti all'apprendimento.
- **debug** - un valore booleano. Se è `true`, indica che il tipo di dettaglio del livello di registrazione per il sistema dell'ambiente di runtime è impostato su debug. Per risultati ottimali, selezionare questo valore prima di scrivere in un log.

Se il metodo `initialize` per qualche motivo ha esito negativo, genera una `LearningException`.

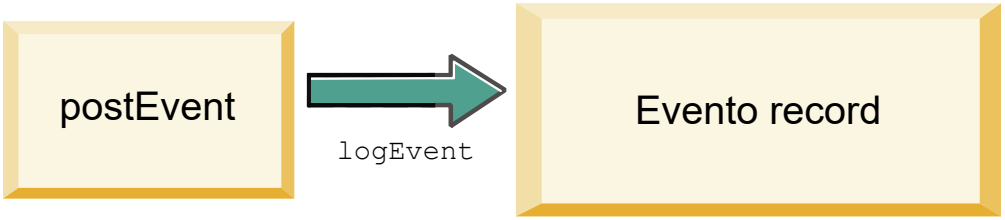
Valore restituito

Nessuno.

logEvent

Il metodo `logEvent` viene richiamato dal server di runtime ogni volta che l'API di Interact invia un evento configurato nel log come contatto o risposta. Utilizzare questo metodo per registrare i dati di contatto e di risposta in un database o in un file a scopo di report e apprendimento. Ad esempio, se si desidera determinare algoritmicamente la probabilità che un cliente accetti un'offerta in base a dei criteri, utilizzare questo metodo per registrare i dati.

```
logEvent(ILearningContext context,
             IOffer offer,
             IClientArgs clientArgs,
             IInteractSession session,
             boolean debug)
```



- **context** - un oggetto `ILearningContext` che definisce il contesto di apprendimento dell'evento, ad esempio, contatto, accettazione o rifiuto.
- **offer** - un oggetto `IOffer` che definisce l'offerta per la quale questo evento è stato registrato.
- **clientArgs** - un oggetto `IClientArgs` che definisce qualsiasi parametro. Attualmente `logEvent` non richiede alcun `clientArgs`, quindi questo parametro potrebbe essere vuoto.
- **session** - un oggetto `IInteractSession` che definisce tutti i dati della sessione.
- **debug** - un valore booleano. Se è `true`, indica che il tipo di dettaglio del livello di registrazione per il sistema dell'ambiente di runtime è impostato su debug. Per risultati ottimali, selezionare questo valore prima di scrivere in un log.

Se il metodo `logEvent` ha esito negativo, genera una `LearningException`.

Valore restituito

Nessuno.

optimizeRecommendList

Il metodo `optimizeRecommendList` prende l'elenco delle offerte consigliate e i dati di sessione e restituisce un elenco che contiene il numero di offerte richiesto. Il metodo `optimizeRecommendList` ordina le offerte in qualche modo, con l'algoritmo di apprendimento proprio dell'utente. L'elenco di offerte deve essere ordinato in modo che le offerte che si desidera presentare per prime siano all'inizio dell'elenco. Ad esempio, se l'algoritmo di apprendimento fornisce un punteggio basso alle offerte migliori, le offerte devono seguire l'ordine 1, 2, 3. Se l'algoritmo di apprendimento fornisce un punteggio alto alle offerte migliori, le offerte devono seguire l'ordine 100, 99, 98.

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

Il metodo `optimizeRecommendList` richiede i seguenti parametri:

- **recList** - un elenco di oggetti trattamento (offerte) consigliati dall'ambiente di runtime.

- **clientArg** - un oggetto IClientArgs che contiene almeno il numero di offerte richieste dall'ambiente di runtime.
- **session** - un oggetto IInteractSession che contiene tutti i dati della sessione.
- **debug** - un valore booleano. Se è `true`, indica che il tipo di dettaglio del livello di registrazione per il sistema dell'ambiente di runtime è impostato su debug. Per risultati ottimali, selezionare questo valore prima di scrivere in un log.

Se il metodo `optimizeRecommendList` ha esito negativo, genera una `LearningException`.

Valore restituito

Il metodo `optimizeRecommendList` restituisce un elenco di oggetti `ITreatment`.

reinitialize

L'ambiente di runtime richiama il metodo `reinitialize` ogni volta che vi è una nuova distribuzione. Questo metodo passa tutti i dati di configurazione dell'apprendimento. Se l'API di apprendimento richiede dei servizi che leggono le proprietà di configurazione, questa interfaccia li dovrebbe riavviare.

```
reinitialize(ILearningConfig config,
                  boolean debug)
```

- **config** - un oggetto `ILearningConfig` che contiene tutte le proprietà di configurazione.
- **debug** - un valore booleano. Se è `true`, indica che il tipo di dettaglio del livello di registrazione per il sistema dell'ambiente di runtime è impostato su debug. Per risultati ottimali, selezionare questo valore prima di scrivere in un log.

Se il metodo `logEvent` ha esito negativo, genera una `LearningException`.

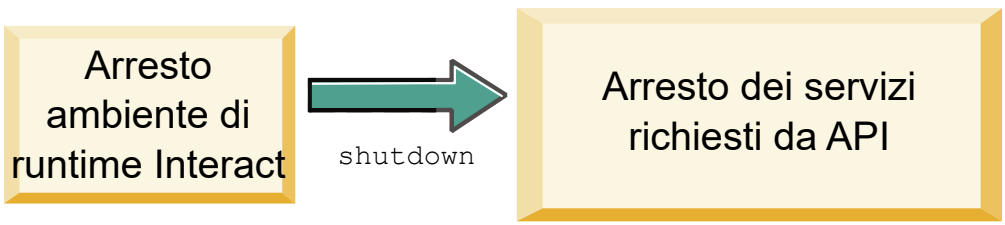
Valore restituito

Nessuno.

shutdown

L'ambiente di runtime richiama il metodo `shutdown` quando il server di runtime si chiude. Se sono richieste attività di ripulitura dal modulo di apprendimento, devono essere eseguite in questo momento.

```
shutdown(ILearningConfig config, boolean debug)
```



Il metodo shutdown richiede i seguenti parametri.

- **config** - un oggetto ILearningConfig che definisce tutte le proprietà di configurazione.
- **debug** - un valore booleano. Se è true, indica che il tipo di dettaglio del livello di registrazione per il sistema dell'ambiente di runtime è impostato su debug. Per risultati ottimali, selezionare questo valore prima di scrivere in un log.

Se il metodo shutdown per qualche motivo ha esito negativo, genera una LearningException.

Valore restituito

Nessuno.

Interfaccia IAudienceID

L'interfaccia IAudienceID supporta l'interfaccia IInteractSession. È un'interfaccia per l'ID destinatario. Poiché l'ID destinatario può essere costituito da diverse parti, questa interfaccia consente all'utente di accedere a tutti gli elementi dell'ID destinatario oltre al nome del livello destinatario.

getAudienceLevel

Il metodo getAudienceLevel restituisce il livello destinatario.

```
getAudienceLevel()
```

Valore restituito

Il metodo getAudienceLevel restituisce una stringa che definisce il livello destinatario.

getComponentNames

Il metodo getComponentNames richiama una serie di nomi dei componenti che comprendono l'ID destinatario. Ad esempio, se l'ID destinatario è costituito dai valori di customerName e accountID, getComponentNames restituirebbe una serie contenente le stringhe customerName e accountID.

```
getComponentNames()
```

Valore restituito

Una serie di stringhe che contengono i nomi dei componenti dell'ID destinatario.

getComponentValue

Il metodo getComponentValue restituisce il valore per il componente nominato.

```
getComponentValue(String componentName)
```

- **componentName**-una stringa che definisce il nome del componente per il quale si desidera recuperare il valore. Questa stringa non è sensibile al maiuscolo/minuscolo.

Valore restituito

Il metodo getComponentValue restituisce un oggetto che definisce il valore del componente.

IClientArgs

L'interfaccia IClientArgs supporta l'interfaccia ILearning. Questa interfaccia è un'astrazione per coprire tutti i dati trasmessi al server dal touchpoint che non sono stati già coperti dai dati di sessione. Ad esempio, il numero di offerte richieste dal metodo getOffers dell'API Interact. Questi dati vengono memorizzati in un'associazione.

getValue

Il metodo getValue restituisce il valore dell'elemento di associazione richiesto.

```
getValue(int clientArgKey)
```

Sono richiesti i seguenti elementi nell'associazione.

- **1** - NUMBER_OF_OFFERS_REQUESTED. Il numero di offerte richieste dal metodo getOffers dell'API Interact. Questa costante restituisce un numero intero.

Valore restituito

Il metodo getValue restituisce un oggetto che definisce il valore della costante di associazione richiesta.

IInteractSession

L'interfaccia IInteractSession supporta l'interfaccia ILearning. È un'interfaccia per la sessione corrente nell'ambiente di runtime.

getAudienceId

Il metodo getAudienceId restituisce un oggetto AudienceID. Utilizzare l'interfaccia IAudienceID per estrarre i valori.

```
getAudienceId()
```

Valore restituito

Il metodo getAudienceId restituisce un oggetto AudienceID.

getSessionData

Il metodo getSessionData restituisce un'associazione non modificabile dei dati di sessione in cui il nome della variabile di sessione è la chiave. Il nome della variabile di sessione è sempre a lettere maiuscole. Utilizzare l'interfaccia IInteractSessionData per estrarre i valori.

```
getSessionData()
```

Valore restituito

Il metodo getSessionData restituisce un oggetto IInteractSessionData.

Interfaccia IInteractSessionData

L'interfaccia `IInteractSessionData` supporta l'interfaccia `ILearning`. È un'interfaccia per i dati della sessione di runtime per il visitatore corrente. I dati della sessione vengono memorizzati come elenco di coppie nome-valore. È possibile inoltre utilizzare questa interfaccia per modificare il valore dei dati nella sessione di runtime.

getDataType

Il metodo `getDataType` restituisce il tipo di dati per il nome parametro specificato.

```
getDataType(string parameterName)
```

Valore restituito

Il metodo `getDataType` restituisce un oggetto InteractDataType. IntearctDataType è un'enumerazione Java rappresentata da Sconosciuto, Stringa, Doppio, Data o Elenco.

getParameterNames

Il metodo `getParameterNames` restituisce un insieme di tutti i nomi dei dati nella sessione corrente.

```
getParameterNames()
```

Valore restituito

Il metodo `getParameterNames` restituisce un insieme dei nomi per i quali sono stati impostati i valori. Ciascun nome presente nell'insieme può essere passato in getValue(String) per restituire un valore.

getValue

Il metodo `getValue` restituisce il valore dell'oggetto corrispondente al `parameterName` specificato. L'oggetto può essere una stringa (String), un valore doppio (Double) o una data (Date).

```
getValue(parameterName)
```

Il metodo `getValue` richiede il seguente parametro:

- **parameterName** - una stringa che definisce il nome della coppia nome-valore dei dati di sessione.

Valore restituito

Il metodo `getValue` restituisce un oggetto che contiene il valore del parametro nominato.

setValue

Il metodo `setValue` consente di impostare un valore per il `parameterName` specificato. Il valore può essere una stringa (String), un valore doppio (Double) o una data (Date).

```
setValue(string parameterName, object value)
```

Il metodo `setValue` richiede i seguenti parametri:

- **parameterName** - una stringa che definisce il nome della coppia nome-valore dei dati di sessione.

- **value** - un oggetto che definisce il valore del parametro designato.

Valore restituito

Nessuno.

ILearningAttribute

L'interfaccia `ILearningAttribute` supporta l'interfaccia `ILearningConfig`. È un'interfaccia per gli attributi di apprendimento definiti nelle proprietà di configurazione nella categoria `learningAttributes`.

getName

Il metodo `getName` restituisce il nome dell'attributo di apprendimento.

```
getName()
```

Valore restituito

Il metodo `getName` restituisce una stringa che definisce il nome dell'attributo di apprendimento.

ILearningConfig

L'interfaccia `ILearningConfig` supporta l'interfaccia `ILearning`. È un'interfaccia per le proprietà di configurazione per l'apprendimento. Tutti questi metodi restituiscono il valore della proprietà.

L'interfaccia è costituita da 15 metodi:

- **getAdditionalParameters** - restituisce un'associazione delle eventuali proprietà aggiuntive definite nella categoria `External Learning Config`
- **getAggregateStatsIntervalInMinutes** - restituisce un int
- **getConfidenceLevel** - restituisce un int
- **getDataSourceName** - restituisce una stringa
- **getDataSourceType** - restituisce una stringa
- **getInsertRawStatsIntervalInMinutes** - restituisce un int
- **getLearningAttributes** - restituisce un elenco di oggetti `ILearningAttribute`
- **getMaxAttributeNames** - restituisce un int
- **getMaxAttributeValues** - restituisce un int
- **getMinPresentCountThreshold** - restituisce un int
- **getOtherAttributeValue** - restituisce una stringa
- **getPercentRandomSelection** - restituisce un int
- **getRecencyWeightingFactor** – restituisce un float
- **getRecencyWeightingPeriod** - restituisce un int
- **isPruningEnabled** - restituisce un booleano

ILearningContext

L'interfaccia `ILearningContext` supporta l'interfaccia `ILearning`.

getLearningContext

Il metodo `getLearningContext` restituisce la costante che informa se questo è uno scenario di contatto, di accettazione o di rifiuto.

```
getLearningContext()
```

- **1**-LOG_AS_CONTACT
- **2**-LOG_AS_ACCEPT
- **3**-LOG_AS_REJECT

4 e 5 sono riservati per un futuro utilizzo.

Valore restituito

Il metodo `getLearningContext` restituisce un numero intero.

getResponseCode

Il metodo `getResponseCode` restituisce il codice di risposta assegnato a questa offerta. Questo valore deve essere presente nella tabella `UA_UsrResponseType` nelle tabelle di sistema di Campaign.

```
getResponseCode()
```

Valore restituito

Il metodo `getResponseCode` restituisce una stringa che definisce il codice di risposta.

IOffer

L'interfaccia `IOffer` supporta l'interfaccia `ITreatment`. È un'interfaccia per l'oggetto offerta definito nell'ambiente di progettazione. Utilizzare l'interfaccia `IOffer` per raccogliere i dettagli dell'offerta dall'ambiente di runtime.

getCreateDate

Il metodo `getCreateDate` restituisce la data in cui è stata creata l'offerta.

```
getCreateDate()
```

Valore restituito

Il metodo `getCreateDate` restituisce una data che definisce la data in cui è stata creata l'offerta.

getEffectiveDateFlag

Il metodo `getEffectiveDateFlag` restituisce un numero che definisce la data di validità dell'offerta.

```
getEffectiveDateFlag()
```

- **0** - la data di validità è una data assoluta, ad esempio 15 marzo 2010.
- **1** - la data di validità è la data della raccomandazione.

Valore restituito

Il metodo `getEffectiveDateFlag` restituisce un numero intero che definisce la data di validità dell'offerta.

getExpirationDateFlag

Il metodo `getExpirationDateFlag` restituisce un valore intero che descrive la data di scadenza dell'offerta.

```
getExpirationDateFlag()
```

- **0** - una data assoluta, ad esempio 15 marzo 2010.
- **1** - un numero di giorni dopo la raccomandazione, ad esempio 14.
- **2** - fine del mese dopo la raccomandazione. Se un'offerta viene presentata il 31 marzo, l'offerta scade quel giorno.

Valore restituito

Il metodo `getExpirationDateFlag` restituisce un numero intero descrive la data di scadenza dell'offerta.

getOfferAttributes

Il metodo `getOfferAttributes` restituisce gli attributi offerta definiti per l'offerta come oggetto `IOfferAttributes`.

```
getOfferAttributes()
```

Valore restituito

Il metodo `getOfferAttributes` restituisce un oggetto `IOfferAttributes`.

getOfferCode

Il metodo `getOfferCode` restituisce il codice offerta dell'offerta definito in Campaign.

```
getOfferCode()
```

Valore restituito

Il metodo `getOfferCode` restituisce un oggetto `IOfferCode`.

getOfferDescription

Il metodo `getOfferDescription` restituisce la descrizione dell'offerta definita in Campaign.

```
getOfferDescription()
```

Valore restituito

Il metodo `getOfferDescription` restituisce una stringa.

getOfferID

Il metodo `getOfferID` restituisce l'ID offerta definito in Campaign.

```
getOfferID()
```

Valore restituito

Il metodo `getOfferID` restituisce un valore lungo che definisce l'ID offerta.

getOfferName

Il metodo `getOfferName` restituisce il nome dell'offerta definito in Campaign.

```
getOfferName()
```

Valore restituito

Il metodo `getOfferName` restituisce una stringa.

getUpdateDate

Il metodo `getUpdateDate` restituisce la data dell'ultimo aggiornamento dell'offerta.

```
getUpdateDate()
```

Valore restituito

Il metodo `getUpdateDate` restituisce una data che definisce quando è stata aggiornata l'offerta l'ultima volta.

IOfferAttributes

L'interfaccia `IOfferAttributes` supporta l'interfaccia `IOffer`. È un'interfaccia per gli attributi dell'offerta definiti per un'offerta nell'ambiente di progettazione. Utilizzare l'interfaccia `IOfferAttributes` per raccogliere gli attributi dell'offerta dall'ambiente di runtime.

getParameterNames

Il metodo `getParameterNames` restituisce un elenco dei nomi parametro dell'offerta.

```
getParameterNames()
```

Valore restituito

Il metodo `getParameterNames` restituisce una serie che definisce l'elenco di nomi parametro dell'offerta.

getValue

Il metodo `getValue` restituisce un oggetto che definisce il valore dell'attributo offerta.

```
getValue(String parameterName)
```

Il metodo `getValue` restituisce il valore dell'attributo offerta indicato.

Valore restituito

Interfaccia IOfferCode

L'interfaccia `IOfferCode` supporta l'interfaccia `ILearning`. È un'interfaccia per il codice offerta definito per un'offerta nell'ambiente di progettazione. Un codice offerta può essere costituito da una a molte stringhe. Utilizzare l'interfaccia `IOfferCode` per raccogliere il codice offerta dall'ambiente di runtime.

getPartCount

Il metodo `getPartCount` restituisce il numero di parti che compongono un codice offerta.

getPartCount()

Valore restituito

Il metodo getPartCount restituisce un numero intero che definisce il numero di parti del codice offerta.

getParts

Il metodo getParts richiama un elenco non modificabile delle parti del codice offerta.

getParts()

Valore restituito

Il metodo getParts restituisce un elenco non modificabile delle parti del codice offerta.

LearningException

La classe LearningException supporta l'interfaccia ILearning. Alcuni metodi all'interno dell'interfaccia richiederanno le implementazioni per generare una LearningException che è una semplice sottoclasse di java.lang.Exception. Si consiglia vivamente a scopo di debug che la LearningException venga costruita con l'eccezione root se ne esiste una.

IScoreOverride

L'interfaccia IScoreOverride supporta l'interfaccia ITreatment. Questa interfaccia consente all'utente di leggere i dati definiti nella tabella di sovrascrittura del punteggio o nella tabella delle offerte predefinite.

getOfferCode

Il metodo getOfferCode restituisce il valore delle colonne di codice offerta nella tabella di sovrascrittura dei punteggi per questo membro destinatario.

getOfferCode()

Valore restituito

Il metodo getOfferCode restituisce un oggetto IOfferCode che definisce il valore delle colonne di codice offerta nella tabella di sovrascrittura dei punteggi.

getParameterNames

Il metodo getParameterNames restituisce l'elenco di parametri.

getParameterNames()

Valore restituito

Il metodo getParameterNames restituisce una serie che definisce l'elenco di parametri.

Il metodo IScoreOverride contiene i seguenti parametri. A meno che non sia indicato diversamente, questi parametri sono gli stessi della tabella di sovrascrittura dei punteggi.

- ADJ_EXPLORE_SCORE_COLUMN

- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT - Per sovrascrivere il conteggio presente stimato (durante il calcolo del peso dell'offerta)
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN - Per la creazione di un'espressione booleana per determinare l'idoneità dell'offerta
- PREDICATE_SCORE - Per la creazione di un'espressione che ha come risultato un punteggio numerico
- SCORE_COLUMN
- ZONE_COLUMN

È possibile inoltre fare riferimento a qualsiasi colonna aggiunta alla tabella di sovrascrittura dei punteggi o delle offerte predefinite utilizzando lo stesso nome della colonna.

getValue

Il metodo `getValue` restituisce il valore della colonna zona nella tabella di sovrascrittura dei punteggi per questo membro destinatario.

`getValue(String parameterName)`

- **parameterName** - una stringa che definisce il nome del parametro per il quale si desidera il valore.

Valore restituito

Il metodo `getValue` restituisce un oggetto che definisce il valore del parametro richiesto.

ISelectionMethod

L'interfaccia `ISelection` indica il metodo utilizzato per ottenere l'elenco consigliato. Il valore predefinito per l'oggetto trattamento è EXTERNAL_LEARNING quindi non è necessario impostare questo valore. Il valore viene infine memorizzato nella cronologia dei contatti dettagliata a scopo di report.

È possibile estendere questa interfaccia oltre le costanti esistenti se si desidera memorizzare i dati per una successiva analisi. Ad esempio, si potrebbe creare due diversi moduli di apprendimento ed implementarli su gruppi di server separati. Si potrebbe estendere l'interfaccia `ISelection` per includere SERVER_GROUP_1 e SERVER_GROUP_2. Si potrebbero quindi confrontare i risultati dei due moduli di apprendimento.

Interfaccia ITreatment

L'interfaccia `ITreatment` supporta l'interfaccia `ILearning` come interfaccia per le informazioni sul trattamento. Un trattamento rappresenta l'offerta assegnata ad una particolare cella definita nell'ambiente di progettazione. Da questa interfaccia è possibile ottenere informazioni sulla cella e sull'offerta oltre al punteggio di marketing assegnato.

getCellCode

Il metodo `getCellCode` restituisce il codice cella definito in Campaign. La cella è la cella assegnata al segmento smart associato a questa offerta.

```
getCellCode()
```

Valore restituito

Il metodo `getCellCode` restituisce una stringa che definisce il codice cella.

getCellId

Il metodo `getCellId` restituisce l'ID interno della cell come definito in Campaign. La cella è la cella assegnata al segmento smart associato a questa offerta.

```
getOfferName()
```

Valore restituito

Il metodo `getCellId` restituisce un long che definisce ID di chiamata.

getCellName

Il metodo `getCellName` restituisce il nome della cella definito in Campaign. La cella è la cella assegnata al segmento smart associato a questa offerta.

```
getCellName()
```

Valore restituito

Il metodo `getCellName` restituisce una stringa che definisce il nome della cella.

getLearningScore

Il metodo `getLearningScore` restituisce il punteggio per questo trattamento.

```
getLearningScore()
```

La precedenza è quella riportata di seguito.

1. Restituire il valore di sovrascrittura, se presente nell'associazione di valori di sovrascrittura inseriti da `IScoreoveride.PREDICATE_SCORE_COLUMN`
2. Restituire il punteggio predicato se il valore non è null
3. Restituire il punteggio marketer, se presente nell'associazione dei valori di sovrascrittura inseriti da `IScoreoveride.SCORE`
4. Restituire il punteggio marketer

Valore restituito

Il metodo `getLearningScore` restituisce un numero intero che definisce il punteggio determinato dall'algoritmo di apprendimento.

getMarketerScore

Il metodo `getMarketerScore` restituisce il punteggio marketer definito dall'indicatore nella scheda della strategia di interazione per l'offerta.

```
getMarketerScore()
```

Per recuperare un punteggio marketer definito dalle opzioni avanzate della scheda della scheda della strategia di interazione, utilizzare `getPredicateScore`.

Per recuperare il punteggio marketer realmente utilizzato dal trattamento, utilizzare `getLearningScore`.

Valore restituito

Il metodo `getMarketerScore` restituisce un numero intero che definisce il punteggio marketer.

getOffer

Il metodo `getOffer` restituisce l'offerta per il trattamento.

```
getOffer()
```

Valore restituito

Il metodo `getOffer` restituisce un oggetto `IOffer` che definisce l'offerta per questo trattamento.

getOverrideValues

Il metodo `getOverrideValues` restituisce le sovrascritture definite nelle offerte predefinite o nelle tabelle di sovrascrittura dei punteggi.

```
getOverrideValues()
```

Valore restituito

Il metodo `getOverrideValues` restituisce un oggetto `IScoreOverride`.

getPredicate

Il metodo `getPredicate` restituisce il predicato definito dalla colonna predicato della tabella di offerte predefinite, della tabella di sovrascrittura dei punteggi o delle opzioni avanzate delle regole di trattamento.

```
getPredicate()
```

Valore restituito

Il metodo `getPredicate` restituisce una stringa che definisce il predicato definito dalla colonna predicato della tabella di offerte predefinite, della tabella di sovrascrittura dei punteggi o delle opzioni avanzate delle regole di trattamento.

getPredicateScore

Il metodo `getPredicateScore` restituisce il punteggio impostato dalla colonna predicato della tabella di offerte predefinite, della tabella di sovrascrittura dei punteggi o delle opzioni avanzate delle regole di trattamento.

```
getPredicateScore()
```

Valore restituito

Il metodo `getPredicateScore` restituisce un valore doppio che definisce il punteggio impostato dalla colonna predicato della tabella di offerte predefinite, della tabella di sovrascrittura dei punteggi o delle opzioni avanzate delle regole di trattamento.

getScore

Il metodo getScore restituisce il punteggio di marketing definito dalla strategia di interazione in Campaign o dalla tabella di sovrascrittura dei punteggi.

```
getScore()
```

Il metodo getScore restituisce uno dei seguenti punteggi:

- Il punteggio di marketing dell'offerta definito nella scheda della strategia di interazione in Campaign se la proprietà enableScoreOverrideLookup è impostata su false.
- Il punteggio dell'offerta definito da scoreOverrideTable se la proprietà enableScoreOverrideLookup è impostata su true.

Valore restituito

Il metodo getScore restituisce un numero intero che rappresenta il punteggio dell'offerta.

getTreatmentCode

Il metodo getTreatmentCode restituisce il codice trattamento.

```
getTreatmentCode()
```

Valore restituito

Il metodo getTreatmentCode restituisce una stringa che definisce il codice trattamento.

setActualValueUsed

Utilizzare il metodo setActualValueUsed per definire quali valori vengono utilizzati nelle varie fasi durante l'esecuzione dell'algoritmo di apprendimento.

```
setActualValueUsed(string parmName, object value)
```

Ad esempio, se si utilizza questo metodo per scrivere nelle tabelle della cronologia dei contatti e delle risposte, e si modificano i report di esempio esistenti, è possibile includere nel report i dati del proprio algoritmo di apprendimento.

- **parmName** - una stringa che definisce il nome del parametro che si sta impostando.
- **value** - un oggetto che definisce il valore del parametro che si sta impostando.

Valore restituito

Nessuno.

Esempio di API di apprendimento

Questa sezione contiene un'implementazione di esempio di ILearningInterface. Tenere presente che questa implementazione è semplicemente un esempio e non è progettata per essere utilizzata in un ambiente di produzione.

Questo esempio tiene traccia dei conteggi delle accettazioni e dei contatti ed utilizza il rapporto tra accettazioni e contatti relativo ad una particolare offerta come tasso di probabilità di accettazione dell'offerta. Le offerte non presentate

hanno priorità maggiore per la raccomandazione. Le offerte con almeno un contatto vengono ordinate in base al tasso decrescente di probabilità di accettazione.

In questo esempio, tutti i conteggi vengono conservati in memoria. Non è uno scenario realistico in quanto il server di runtime esaurirà la memoria. In un vero scenario di produzione, i conteggi devono essere resi permanenti in un database.

```
package com.unicacorp.interact.samples.learning.v2;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.unicacorp.interact.samples.learning.SampleOptimizer.MyOfferSorter;
import com.unicacorp.interact.treatment.optimization.IClientArgs;
import com.unicacorp.interact.treatment.optimization.IInteractSession;
import com.unicacorp.interact.treatment.optimization.ILearningConfig;
import com.unicacorp.interact.treatment.optimization.ILearningContext;
import com.unicacorp.interact.treatment.optimization.IOffer;
import com.unicacorp.interact.treatment.optimization.LearningException;
import com.unicacorp.interact.treatment.optimization.v2.ILearning;
import com.unicacorp.interact.treatment.optimization.v2.ITreatment;

/**
 * Questo è un esempio di implementazione dell'optimizer di apprendimento.
 * L'interfaccia ILearning si trova nella libreria interact.jar.
 *
 * Per utilizzare realmente questa implementazione, selezionare ExternalLearning come optimizationType nel nodo offerServing
 * dell'applicazione Interact nella configurazione della piattaforma.  All'interno del nodo offerserving è presente anche
 * una categoria di configurazione di apprendimento esterno - al cui interno è necessario specificare il nome della classe su quanto segue:
 * com.unicacorp.interact.samples.learning.v2.SampleLearning.  tenere presente, tuttavia, che questa implementazione è solo un esempio
 * e non è stata progettata per essere utilizzata in un ambiente di produzione.
 *
 *
 * Questo esempio tiene traccia dei conteggi delle accettazioni e dei contatti ed utilizza il rapporto tra accettazioni e contatti
 * relativo ad una particolare offerta come tasso di probabilità di accettazione dell'offerta.
 *
 *
 * Le offerte non presentate avranno una priorità maggiore per la raccomandazione.
 * Le offerte con almeno un contatto verranno ordinate in base al tasso decrescente di probabilità di accettazione.
 *
 * Nota: tutti i conteggi vengono conservati in memoria.  Questo non è uno scenario realistico poiché prima o poi si esaurirebbe la
 * memoria.  In un vero scenario di produzione, i conteggi devono essere resi permanenti in un database.
 *
 */

public class SampleLearning implements ILearning
{

    // Un'associazione del conteggio di id offerte e contatti per l'id offerta
    private Map<Long,Integer> _offerToContactCount = new HashMap<Long, Integer>();

    // Un'associazione del conteggio di id offerte e contatti per l'id offerta
    private Map<Long,Integer> _offerToAcceptCount = new HashMap<Long, Integer>();


    /* (non Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#initialize
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void initialize(ILearningConfig config, boolean debug) throws LearningException
    {
        // Se sono richieste connessioni remote, questo è il punto giusto in cui inizializzare tali connessioni in quanto
        // questo metodo viene richiamato una sola volta all'avvio della webapp di runtime Interact.
        // Questo esempio non presenta connessioni remote e stampa a scopo di debug che questo metodo verrà
        // richiamato
        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
{
    // Se viene distribuito un IC, viene richiamato questo metodo di reinizializzazione per consentire all'implementazione di
    // aggiornare le eventuali impostazioni di configurazione aggiornate
    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");

        // Tenere traccia di tutti i contatti in 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");
        // Tenere traccia di tutti i conteggi di accettazione in memoria effettuando l'aggiunta all'associazione
        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");

    // Ordinare i trattamenti candidati richiamando il sorter definito in questa classe e restituire l'elenco ordinato
    Collections.sort(recList,new MyOfferSorter());

    // ora si restituisce ciò che era stato chiesto tramite le variabile "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
    {
        // Se esistono connessioni remote, questo sarebbe un buon punto per effettuarne tranquillamente la
        // disconnessione poiché questo metodo viene richiamato alla chiusura della webapp di runtime
        // di Interact.   Per questo esempio, non c'è in realtà molto da fare se non
        // stampare un resoconto per il debug.
        System.out.println("Calling shutdown for SampleLearning");

    }
    // Ordinare in base a:
    // 1. offerte con zero contatti  - per i legami, l'ordine si basa sull'input originale
    // 2. tasso discendente di probabilità di accettazione - per i legami, l'ordine si basa sull'input originale

    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)
        {

            // ottenere il conteggio contatti di entrambi i trattamenti
            Integer contactCount1 = _offerToContactCount.get(treatment1.getOffer().getOfferId());
            Integer contactCount2 = _offerToContactCount.get(treatment2.getOffer().getOfferId());

            // se il trattamento non è stato contattato, prevale quello
            if(contactCount1 == null || contactCount1 == 0)
                return -1;

            if(contactCount2 == null || contactCount2 == 0)
                return 1;

            // ottenere i conteggi accettazioni
            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;

            // ordine discendente
            return (int) (acceptProbability2 - acceptProbability1);

        }
    }

}
```

Capitolo 12. WSDL di IBM Interact

L'installazione di Interact include due file XML WSDL (Web Services Description Language), atti a descrivere i servizi Web disponibili e la modalità di accesso a tali servizi. Si possono esaminare questi file nella directory principale di Interact e in questa sezione viene mostrato un esempio.

Dopo aver installato Interact, si possono reperire i file WSDL di Interact nella seguente ubicazione:

- *<Interact_home>*/conf/InteractService.wsdl
- *<Interact_home>*/conf/InteractAdminService.wsdl

In ogni release del software o fix pack, è possibile che vi siano delle modifiche al WSDL di Interact. Consultare il manuale *Interact - Note sulla release* o i file readme allegati alla release, per dettagli.

Come riferimento, in questa sezione viene presentata una copia del file InteractService.wsdl. Per assicurarsi di stare utilizzando le informazioni più recenti, esaminare i file WSDL installati 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>
```

Capitolo 13. Proprietà di configurazione dell'ambiente di runtime di Interact

Questa sezione descrive tutte le proprietà di configurazione per l'ambiente di runtime di Interact.

Interact | general

Queste proprietà di configurazione definiscono le impostazioni generali per l'ambiente di runtime, incluso il livello di registrazione predefinito e l'impostazione della locale.

log4jConfig

Descrizione

L'ubicazione del file che contiene le proprietà log4j. Questo percorso deve essere relativo alla variabile di ambiente `INTERACT_HOME`. `INTERACT_HOME` è l'ubicazione della directory di installazione di Interact.

Valore predefinito

`./conf/interact_log4j.properties`

asmUserForDefaultLocale

Descrizione

La proprietà `asmUserForDefaultLocale` definisce l'utente IBM Marketing Software da cui Interact deriva le relative impostazioni della locale.

Le impostazioni della locale definiscono la lingua visualizzata nella fase di progettazione e la lingua in cui sono presentati i messaggi di avviso dell'API di Interact. Se l'impostazione della locale non corrisponde alle impostazioni del sistema operativo delle macchine, Interact continua a funzionare, tuttavia la visualizzazione della fase di progettazione e i messaggi di avviso potrebbero essere in una lingua diversa.

Valore predefinito

`asm_admin`

Interact | general | learningTablesDataSource

Queste proprietà di configurazione definiscono le impostazioni dell'origine dati per le tabelle di apprendimento integrato. È necessario definire questa origine dati, se si sta utilizzando l'apprendimento integrato di Interact.

Se si crea una propria implementazione di apprendimento utilizzando l'API di apprendimento, è possibile configurare l'implementazione di apprendimento personalizzata per leggere questi valori utilizzando l'interfaccia `ILearningConfig`.

jndiName

Descrizione

Utilizzare la proprietà jndiName per identificare l'origine dati JNDI (Java Naming and Directory Interface), definita nel server delle applicazioni (Websphere o WebLogic) per le tabelle di apprendimento a cui accedono i server di runtime Interact.

Le tabelle di apprendimento vengono create dal file ddl aci_lrntab e contengono le seguenti tabelle (tra le altre): UACI_AttributeValue e UACI_OfferStats.

Valore predefinito

Non è specificato alcun valore predefinito.

type

Descrizione

Il tipo di database per l'origine dati utilizzata dalle tabelle di apprendimento a cui accedono i server di runtime Interact.

Le tabelle di apprendimento vengono create dal file ddl aci_lrntab e contengono le seguenti tabelle (tra le altre): UACI_AttributeValue e UACI_OfferStats.

Valore predefinito

SQLServer

Valori validi

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descrizione

La proprietà ConnectionRetryPeriod specifica il periodo di tempo, in secondi, durante il quale Interact ripete automaticamente la richiesta di connessione al database, quando non riesce, per le tabelle di apprendimento. Interact tenta automaticamente di riconnettersi al database per questo periodo di tempo, prima di riportare un errore o malfunzionamento del database. Se il valore è impostato su 0, Interact riproverà per un tempo indefinito; se il valore è impostato su -1, non verranno effettuati tentativi.

Le tabelle di apprendimento vengono create dal file ddl aci_lrntab e contengono le seguenti tabelle (tra le altre): UACI_AttributeValue e UACI_OfferStats.

Valore predefinito

-1

connectionRetryDelay

Descrizione

La proprietà ConnectionRetryDelay specifica il periodo di tempo, in secondi, durante il quale Interact rimane in attesa, prima di riprovare a connettersi al database dopo un errore, per le tabelle di apprendimento. Se il valore è impostato su -1, non verranno effettuati tentativi.

Le tabelle di apprendimento vengono create dal file ddl aci_lrntab e contengono le seguenti tabelle (tra le altre): UACI_AttributeValue e UACI_OfferStats.

Valore predefinito

-1

schema

Descrizione

Il nome dello schema che contiene le tabelle per il modulo di apprendimento integrato. Interact inserisce il valore di questa proprietà prima di tutti i nomi tabella, ad esempio `UACI_IntChannel` diventa `schema.UACI_IntChannel`.

Non è necessario definire uno schema. Se non si definisce uno schema, Interact presuppone che il proprietario delle tabelle sia lo stesso dello schema. Si consiglia di impostare questo valore per eliminare l'ambiguità.

Valore predefinito

Non è specificato alcun valore predefinito.

Interact | general | prodUserDataSource

Queste proprietà di configurazione definiscono le impostazioni dell'origine dati per le tabelle dei profili di produzione. È necessario definire questa origine dati. È l'origine dati a cui fa riferimento l'ambiente di runtime, quando esegue diagrammi di flusso interattivo dopo la distribuzione.

jndiName

Descrizione

Utilizzare la proprietà `jndiName` per identificare l'origine dati JNDI (Java Naming and Directory Interface), definita nel server delle applicazioni (Websphere o WebLogic) per le tabelle cliente a cui accedono i server di runtime Interact.

Valore predefinito

Non è specificato alcun valore predefinito.

type

Descrizione

Il tipo di database per le tabelle cliente a cui accedono i server di runtime Interact.

Valore predefinito

SQLServer

Valori validi

SQLServer | DB2 | ORACLE

aliasPrefix

Descrizione

La proprietà `AliasPrefix` specifica il modo in cui Interact forma il nome alias che Interact crea automaticamente quando utilizza una tabella dimensionale e scrive in una nuova tabella nelle tabelle cliente a cui accedono i server di runtime Interact.

Tenere presente che ciascun database ha una lunghezza massima per l'identificativo; consultare la documentazione relativa al database utilizzato per accertarsi che il valore impostato non superi la lunghezza massima dell'identificativo per il database.

Valore predefinito

A

connectionRetryPeriod

Descrizione

La proprietà `ConnectionRetryPeriod` specifica il periodo di tempo, in secondi, durante il quale Interact ù ripete automaticamente la richiesta di connessione al database, quando non riesce, per le tabelle cliente di runtime. Interact tenta automaticamente di riconnettersi al database per questo periodo di tempo, prima di riportare un errore o malfunzionamento del database. Se il valore è impostato su 0, Interact riproverà per un tempo indefinito; se il valore è impostato su -1, non verranno effettuati tentativi.

Valore predefinito

-1

connectionRetryDelay

Descrizione

La proprietà `ConnectionRetryDelay` specifica il periodo di tempo, in secondi, durante il quale Interact rimane in attesa, prima di riprovare a connettersi al database dopo un errore, per le tabelle cliente di runtime di Interact. Se il valore è impostato su -1, non verranno effettuati tentativi.

Valore predefinito

-1

schema

Descrizione

Il nome dello schema che contiene le tabelle di dati del profilo. Interact inserisce il valore di questa proprietà prima di tutti i nomi tabella, ad esempio `UACI_IntChannel` diventa `schema.UACI_IntChannel`.

Non è necessario definire uno schema. Se non si definisce uno schema, Interact presuppone che il proprietario delle tabelle sia lo stesso dello schema. Si consiglia di impostare questo valore per eliminare l'ambiguità.

Quando si utilizza un database DB2, il nome schema deve essere in maiuscolo.

Valore predefinito

Non è specificato alcun valore predefinito.

Interact | general | systemTablesDataSource

Queste proprietà di configurazione definiscono le impostazioni dell'origine dati per le tabelle di sistema per l'ambiente di runtime. È necessario definire questa origine dati.

jndiName

Descrizione

Utilizzare la proprietà jndiName per identificare l'origine dati JNDI (Java Naming and Directory Interface), definita nel server delle applicazioni (Websphere o WebLogic) per le tabelle dell'ambiente di runtime.

Il database dell'ambiente di runtime è il database in cui gli script dll aci_runtime e aci_populate_runtime inseriscono i dati, ad esempio, contiene le seguenti tabelle (tra le altre): UACI_CHOfferAttrib e UACI_DefaultedStat.

Valore predefinito

Non è specificato alcun valore predefinito.

type

Descrizione

Il tipo di database per le tabelle di sistema dell'ambiente di runtime.

Il database dell'ambiente di runtime è il database in cui gli script dll aci_runtime e aci_populate_runtime inseriscono i dati, ad esempio, contiene le seguenti tabelle (tra le altre): UACI_CHOfferAttrib e UACI_DefaultedStat.

Valore predefinito

SQLServer

Valori validi

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descrizione

La proprietà ConnectionRetryPeriod specifica il periodo di tempo, in secondi, durante il quale Interact ripete automaticamente la richiesta di connessione al database, quando non riesce, per le tabelle di sistema di runtime. Interact tenta automaticamente di riconnettersi al database per questo periodo di tempo, prima di riportare un errore o malfunzionamento del database. Se il valore è impostato su 0, Interact riproverà per un tempo indefinito; se il valore è impostato su -1, non verranno effettuati tentativi.

Il database dell'ambiente di runtime è il database in cui gli script dll aci_runtime e aci_populate_runtime inseriscono i dati, ad esempio, contiene le seguenti tabelle (tra le altre): UACI_CHOfferAttrib e UACI_DefaultedStat.

Valore predefinito

-1

connectionRetryDelay

Descrizione

La proprietà ConnectionRetryDelay specifica il periodo di tempo, in secondi, durante il quale Interact rimane in attesa, prima di riprovare a connettersi al database dopo un errore, per le tabelle di sistema di runtime di Interact. Se il valore è impostato su -1, non verranno effettuati tentativi.

Il database dell'ambiente di runtime è il database in cui gli script dll `aci_runtime` e `aci_populate_runtime` inseriscono i dati, ad esempio, contiene le seguenti tabelle (tra le altre): `UACI_CHOfferAttrib` e `UACI_DefaultedStat`.

Valore predefinito

-1

schema

Descrizione

Il nome dello schema che contiene le tabelle per l'ambiente di runtime. Interact inserisce il valore di questa proprietà prima di tutti i nomi tabella, ad esempio `UACI_IntChannel` diventa `schema.UACI_IntChannel`.

Non è necessario definire uno schema. Se non si definisce uno schema, Interact presuppone che il proprietario delle tabelle sia lo stesso dello schema. Si consiglia di impostare questo valore per eliminare l'ambiguità.

Valore predefinito

Non è specificato alcun valore predefinito.

Interact | general | systemTablesDataSource | loaderProperties

Queste proprietà di configurazione definiscono le impostazioni di un programma di utilità di caricamento del database per le tabelle di sistema per un ambiente di runtime. È necessario definire queste proprietà, se si sta utilizzando solo un programma di utilità di caricamento del database.

databaseName

Descrizione

Il nome del database a cui si connette il programma di utilità di caricamento del database.

Valore predefinito

Non è specificato alcun valore predefinito.

LoaderCommandForAppend

Descrizione

Il parametro `LoaderCommandForAppend` specifica il comando immesso per richiamare il programma di utilità per il caricamento del database per aggiungere i record alle tabelle database di staging della cronologia dei contatti e delle risposte in Interact. È necessario impostare questo parametro, per abilitare il programma di utilità di caricamento del database a gestire i dati della cronologia dei contatti e delle risposte.

Questo parametro viene specificato come nome di percorso completo per l'eseguibile del programma di utilità per il caricamento del database o per uno script che avvia il programma di utilità per il caricamento del database. L'utilizzo di uno script consente di eseguire ulteriori operazioni di configurazione, prima di richiamare il programma di utilità per il caricamento.

La maggior parte dei programmi di utilità per il caricamento del database richiedono diversi argomenti per essere avviati in modo corretto. Essi possono includere la specifica del file di dati e del file di controllo da cui

effettuare il caricamento e il database e la tabella in cui effettuare il caricamento. I token vengono sostituiti dagli elementi specificati quando viene eseguito il comando.

Per la sintassi corretta da utilizzare, quando si richiama il programma di utilità per il caricamento del database, consultare la documentazione relativa a tale programma.

Questo parametro non è definito per impostazione predefinita.

I token disponibili per `LoaderCommandForAppend` sono descritti nella seguente tabella.

Token	**Descrizione**
<CONTROLFILE>	Questo token viene sostituito con il nome file e il percorso completo del file di controllo temporaneo che Interact genera, in base al modello specificato nel parametro `LoaderControlFileTemplate`.
<DATABASE>	Questo token viene sostituito con il nome dell'origine dati in cui Interact sta caricando i dati. Si tratta dello stesso nome origine dati utilizzato nel nome categoria per questa origine dati.
<DATAFILE>	Questo token viene sostituito con il percorso completo e il nome file del file di dati temporaneo creato da Interact durante il processo di caricamento. Questo file si trova nella directory Temp di Interact, `UNICA_ACTMPDIR`.
<DBCOLUMNNUMBER>	Questo token viene sostituito con il numero ordinale della colonna del database.
<FIELDLENGTH>	Questo token viene sostituito con la lunghezza del campo caricato nel database.
<FIELDNAME>	Questo token viene sostituito con il nome del campo caricato nel database.
<FIELDNUMBER>	Questo token viene sostituito con il numero del campo caricato nel database.
<FIELDTYPE>	Questo token è sostituito dalla costante letterale "CHAR( )". La lunghezza di questo campo è specificata tra le parentesi (). Se il database non riconosce il tipo di campo, CHAR, è possibile specificare manualmente il testo appropriato per il tipo di campo ed utilizzare il token <FIELDLENGTH>. Ad esempio, per SQLSVR ed SQL2000 si dovrebbe utilizzare "SQLCHAR(<FIELDLENGTH>)"

Token	Descrizione
<NATIVETYPE>	Questo token viene sostituito con il tipo di database in cui il campo viene caricato.
<NUMFIELDS>	Questo token viene sostituito con il numero di campi contenuti nella tabella.
<PASSWORD>	Questo token viene sostituito con la password del database utilizzata per la connessione del diagramma di flusso corrente all'origine dati.
<TABLENAME>	Questo token viene sostituito con il nome della tabella database in cui Interact sta caricando i dati.
<USER>	Questo token viene sostituito con l'utente database utilizzato per la connessione del diagramma di flusso corrente all'origine dati.

Valore predefinito

Non è specificato alcun valore predefinito.

LoaderControlFileTemplateForAppend

Descrizione

La proprietà `LoaderControlFileTemplateForAppend` specifica il percorso completo e il nome file del modello del file di controllo configurato precedentemente in Interact. Quando questo parametro è impostato, Interact crea dinamicamente un file di controllo temporaneo, in base al modello specificato qui. Il percorso e il nome di questo file di controllo temporaneo sono disponibili per il token `<CONTROLFILE>` che è, a sua volta, disponibile per la proprietà `LoaderCommandForAppend`.

Prima di utilizzare Interactin modalità programma di utilità di caricamento del database, è necessario configurare il modello del file di controllo specificato da questo parametro. Il modello del file di controllo supporta i seguenti token, che vengono sostituiti dinamicamente quando il file di controllo temporaneo viene creato da Interact.

Per la sintassi corretta, richiesta per il file di controllo, consultare la documentazione del programma di utilità di caricamento del database. I token disponibili per il modello del file di controllo sono gli stessi della proprietà `LoaderControlFileTemplate`.

Questo parametro non è definito per impostazione predefinita.

Valore predefinito

Non è specificato alcun valore predefinito.

LoaderDelimiterForAppend

Descrizione

La proprietà LoaderDelimiterForAppend specifica se il file di dati temporaneo di Interact è un file flat a larghezza fissa o delimitato e. se delimitato, specifica il carattere o la serie di caratteri utilizzati come delimitatori.

Se il valore non è definito, Interact crea il file di dati temporaneo come file flat a larghezza fissa.

Se si specifica un valore, viene utilizzato quando il programma di caricamento viene richiamato per inserire dati in una tabella riconosciuta come vuota. Interact crea il file di dati temporaneo come file flat delimitato, utilizzando il valore di questa proprietà come delimitatore.

Questa proprietà non è definita per impostazione predefinita.

Valore predefinito

Valori validi

Caratteri, che possono essere racchiusi tra doppi apici, se lo si desidera.

LoaderDelimiterAtEndForAppend

Descrizione

Alcuni programmi di utilità per il caricamento esterni richiedono che il file di dati sia delimitato e che ogni riga termini con un delimitatore. Per soddisfare questo requisito, impostare il valore di LoaderDelimiterAtEndForAppend su TRUE, in modo che, quando il programma di caricamento viene richiamato per inserire dati in una tabella riconosciuta come vuota, Interact utilizzi i delimitatori alla fine di ogni riga.

Valore predefinito

FALSE

Valori validi

TRUE | FALSE

LoaderUseLocaleDP

Descrizione

La proprietà LoaderUseLocaleDP specifica, quando Interact scrive valori numerici in file che devono essere caricati da un programma di utilità per il caricamento del database, se viene utilizzato come segno decimale, il simbolo specifico per la locale.

Impostare questo valore su FALSE, per specificare che il punto (.) viene utilizzato come segno decimale.

Impostare questo valore su TRUE per specificare l'utilizzo del simbolo del segno decimale appropriato alla locale.

Valore predefinito

FALSE

Valori validi

TRUE | FALSE

Interact | general | testRunDataSource

Queste proprietà di configurazione definiscono le impostazioni dell'origine dati per le tabelle di esecuzione di test per l'ambiente di progettazione di Interact. È necessario definire questa origine dati per almeno uno degli ambienti di runtime. Queste sono le tabelle utilizzate durante un'esecuzione di test del diagramma di flusso interattivo.

jndiName

Descrizione

Utilizzare la proprietà `jndiName` per identificare l'origine dati JNDI (Java Naming and Directory Interface), definita nel server delle applicazioni (Websphere o WebLogic) per le tabelle cliente, a cui accede l'ambiente di progettazione nel corso delle esecuzioni di test dei diagrammi di flusso interattivi.

Valore predefinito

Non è specificato alcun valore predefinito.

type

Descrizione

Il tipo di database per le tabelle cliente a cui accede l'ambiente di progettazione durante le esecuzioni di test dei diagrammi di flusso interattivi.

Valore predefinito

SQLServer

Valori validi

SQLServer | DB2 | ORACLE

aliasPrefix

Descrizione

La proprietà `AliasPrefix` specifica il modo in cui Interact forma il nome alias che Interact crea automaticamente quando utilizza una tabella dimensionale e scrive in una nuova tabella, per le tabelle cliente a cui accede l'ambiente di progettazione durante le esecuzioni di test dei diagrammi di flusso interattivi.

Tenere presente che ciascun database ha una lunghezza massima per l'identificativo; consultare la documentazione relativa al database utilizzato per accertarsi che il valore impostato non superi la lunghezza massima dell'identificativo per il database.

Valore predefinito

A

connectionRetryPeriod

Descrizione

La proprietà `ConnectionRetryPeriod` specifica il periodo di tempo, in secondi, durante il quale Interact ripete automaticamente la richiesta di connessione al database, quando non riesce, per le tabelle di esecuzione di test. Interact tenta automaticamente di riconnettersi al database per questo

periodo di tempo, prima di riportare un errore o malfunzionamento del database. Se il valore è impostato su 0, Interact riproverà per un tempo indefinito; se il valore è impostato su -1, non verranno effettuati tentativi.

Valore predefinito

-1

connectionRetryDelay

Descrizione

La proprietà `ConnectionRetryDelay` specifica il periodo di tempo, in secondi, durante il quale Interact rimane in attesa, prima di riprovare a connettersi al database dopo un errore, per le tabelle di esecuzione di test. Se il valore è impostato su -1, non verranno effettuati tentativi.

Valore predefinito

-1

schema

Descrizione

Il nome dello schema che contiene le tabelle per le esecuzioni di test dei diagrammi di flusso interattivo. Interact inserisce il valore di questa proprietà prima di tutti i nomi tabella, ad esempio `UACI_IntChannel` diventa `schema.UACI_IntChannel`.

Non è necessario definire uno schema. Se non si definisce uno schema, Interact presuppone che il proprietario delle tabelle sia lo stesso dello schema. Si consiglia di impostare questo valore per eliminare l'ambiguità.

Valore predefinito

Non è specificato alcun valore predefinito.

Interact | general | contactAndResponseHistoryDataSource

Queste proprietà di configurazione definiscono le impostazioni di connessione per l'origine dati della cronologia dei contatti e delle risposte, richiesta per il tracciamento della risposta delle sessioni incrociate di Interact. Queste impostazioni non sono relative al modulo della cronologia dei contatti e delle risposte.

jndiName

Descrizione

Utilizzare la proprietà `jndiName` per identificare l'origine dati JNDI (Java Naming and Directory Interface) definita nel server delle applicazioni (WebSphere o WebLogic) per l'origine dati della cronologia dei contatti e delle risposte, richiesta per il tracciamento della risposta delle sessioni incrociate di Interact.

Valore predefinito

type

Descrizione

Il tipo di database per l'origine dati, utilizzato dall'origine dati della cronologia dei contatti e delle risposte, richiesta per il tracciamento della risposta delle sessioni incrociate di Interact.

Valore predefinito

SQLServer

Valori validi

SQLServer | DB2 | ORACLE

connectionRetryPeriod

Descrizione

La proprietà `ConnectionRetryPeriod` specifica il periodo di tempo, in secondi, durante il quale Interact ritenta automaticamente la richiesta di connessione al database in seguito a un errore, per il tracciamento della risposta delle sessioni incrociate di Interact. Interact tenta automaticamente di riconnettersi al database per questo periodo di tempo, prima di riportare un errore o malfunzionamento del database. Se il valore è impostato su 0, Interact riproverà per un tempo indefinito; se il valore è impostato su -1, non verranno effettuati tentativi.

Valore predefinito

-1

connectionRetryDelay

Descrizione

La proprietà `ConnectionRetryDelay` specifica il periodo di tempo, in secondi, durante il quale Interact rimane in attesa, prima di tentare di riconnettersi al database in seguito a un errore, per il tracciamento della risposta delle sessioni incrociate di Interact. Se il valore è impostato su -1, non verranno effettuati tentativi.

Valore predefinito

-1

schema

Descrizione

Il nome dello schema contenente le tabelle per il tracciamento della risposta delle sessioni incrociate di Interact. Interact inserisce il valore di questa proprietà prima di tutti i nomi tabella, ad esempio `UACI_IntChannel` diventa `schema.UACI_IntChannel`.

Non è necessario definire uno schema. Se non si definisce uno schema, Interact presuppone che il proprietario delle tabelle sia lo stesso dello schema. Si consiglia di impostare questo valore per eliminare l'ambiguità.

Valore predefinito

Non è specificato alcun valore predefinito.

Interact | general | idsByType

Queste proprietà di configurazione definiscono le impostazioni per i numeri ID utilizzati dal modulo della cronologia dei contatti e delle risposte.

initialValue

Descrizione

Il valore ID iniziale utilizzato durante la generazione degli ID tramite la tabella UACI_IDsByType.

Valore predefinito

1

Valori validi

Qualsiasi valore superiore a 0.

retries

Descrizione

Il numero di tentativi prima di emettere un'eccezione, quando si generano gli ID tramite la tabella UACI_IDsByType.

Valore predefinito

20

Valori validi

Qualsiasi numero intero superiore a 0.

Interact | flowchart

Questa sezione definisce le impostazioni della configurazione per diagrammi di flusso interattivi.

defaultDateFormat

Descrizione

Il formato predefinito della data utilizzato da Interact per convertire il formato Date in String e il formato String in Date.

Valore predefinito

MM/gg/aa

idleFlowchartThreadTimeoutInMinutes

Descrizione

Il numero di minuti di inattività che Interact consente a un thread dedicato a un diagramma di flusso interattivo, prima di rilasciare tale thread.

Valore predefinito

5

idleProcessBoxThreadTimeoutInMinutes

Descrizione

Il numero di minuti di inattività che Interact consente a un thread dedicato a un processo del diagramma di flusso interattivo, prima di rilasciare tale thread.

Valore predefinito

5

maxSizeOfFlowchartEngineInboundQueue

Descrizione

Il numero massimo di richieste di esecuzione del diagramma di flusso che Interact mantiene in coda. Se viene raggiunto questo numero di richieste, Interact interromperà l'acquisizione delle richieste.

Valore predefinito

1000

maxNumberOfFlowchartThreads

Descrizione

Il numero massimo di thread dedicati alle richieste del diagramma di flusso interattivo.

Valore predefinito

25

maxNumberOfProcessBoxThreads

Descrizione

Il numero massimo di thread dedicati ai processi del diagramma di flusso interattivo.

Valore predefinito

50

maxNumberOfProcessBoxThreadsPerFlowchart

Descrizione

Il numero massimo di thread dedicati ai processi del diagramma di flusso interattivo, per istanza del diagramma di flusso.

Valore predefinito

3

minNumberOfFlowchartThreads

Descrizione

Il numero minimo di thread dedicati alle richieste del diagramma di flusso interattivo.

Valore predefinito

10

minNumberOfProcessBoxThreads

Descrizione

Il numero minimo di thread dedicati ai processi del diagramma di flusso interattivo.

Valore predefinito

20

sessionVarPrefix

Descrizione

Il prefisso per le variabili di sessione.

Valore predefinito

SessionVar

Interact | flowchart | ExternalCallouts | [ExternalCalloutName]

Questa sezione definisce le impostazioni della classe per i callout esterni personalizzati, scritti con la relativa API di callout esterno.

class

Descrizione

Il nome della classe Java rappresentata da questo callout esterno.

Si tratta della classe Java a cui si può accedere mediante la macro IBM `EXTERNALCALLOUT`.

Valore predefinito

Non è specificato alcun valore predefinito.

classpath

Descrizione

Il percorso classi per la classe Java rappresentata da questo callout esterno. Il percorso classi deve fare riferimento ai file jar sul server dell'ambiente di runtime. Se si sta utilizzando un gruppo di server e tutti i server di runtime stanno utilizzando lo stesso componente Marketing Platform, ogni server deve avere una copia del file jar nella stessa ubicazione. Il percorso classi deve essere formato dalle ubicazioni assolute dei file jar, separate dal delimitatore di percorso del sistema operativo del server dell'ambiente di runtime, ad esempio un punto e virgola (;) nei sistemi Windows e due punti (:) nei sistemi UNIX. Non sono accettate directory contenenti file di classe. Ad esempio, su un sistema Unix: `/path1/file1.jar:/path2/file2.jar`.

Questo percorso classi non deve superare i 1024 caratteri di lunghezza. È possibile utilizzare il file manifest in un file .jar per specificare altri file .jar, in modo che sia presente un solo file .jar nel percorso classi

Si tratta della classe Java a cui si può accedere mediante la macro IBM `EXTERNALCALLOUT`.

Valore predefinito

Non è specificato alcun valore predefinito.

Interact | flowchart | ExternalCallouts | [ExternalCalloutName] | Parameter Data | [parameterName]

Questa sezione definisce le impostazioni di parametro per un callout esterno personalizzato scritto con la relativa API di callout esterno.

value

Descrizione

Il valore di un qualsiasi parametro richiesto dalla classe per il callout esterno.

Valore predefinito

Non è specificato alcun valore predefinito.

Esempio

Se il callout esterno richiede un nome host di un server esterno, creare una categoria di parametri denominata `host` e definire la proprietà `value` come nome del server.

Interact | monitoring

Questa serie di proprietà di configurazione consente di definire le impostazioni per il monitoraggio JMX. È necessario configurare queste proprietà solo se si sta utilizzando il monitoraggio JMX. Vi sono proprietà di monitoraggio JMX separate da definire per il modulo della cronologia dei contatti e delle risposte nelle proprietà di configurazione per l'ambiente di progettazione Interact.

protocol

Descrizione

Definire il protocollo per il servizio di messaggistica di Interact.

Se si sceglie JMXMP è necessario includere il seguenti file JAR nel proprio percorso classi, nel seguente ordine:

`Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar`

Valore predefinito

JMXMP

Valori validi

JMXMP | RMI

port

Descrizione

Il numero porta per il servizio di messaggistica.

Valore predefinito

9998

enableSecurity

Descrizione

Un valore booleano che abilita o disabilita la sicurezza del servizio di messaggistica JMXMP per il server di runtime Interact. Se questa proprietà è impostata su `true`, è necessario fornire un nome utente e una password per accedere al servizio JMX di runtime Interact. Queste credenziali utente vengono autenticate da Marketing Platform per il server di runtime. Jconsole non consente l'accesso con password vuota.

Questa proprietà non ha effetto se il protocollo è RMI. Questa proprietà non ha alcun effetto su JMX per Campaign (la fase di progettazione di Interact).

Valore predefinito

True

Valori validi

True | False

Interact | monitoring | activitySubscribers

Questa serie di proprietà di configurazione abilita il nodo root alle impostazioni correlate ai sottoscrittori remoti che possono ricevere aggiornamenti periodici su dati sulle prestazioni di base nell'ambiente di runtime di Interact.

heartbeatPeriodInSecs

Descrizione

L'intervallo in secondi in base al quale l'istanza di runtime invia un aggiornamento ai sottoscrittori.

Valore predefinito

60

Interact | monitoring | activitySubscribers | (obiettivo)

(obiettivo)

Descrizione

Il nodo root per le impostazioni di un sottoscrittore.

URL

Descrizione

L'URL di questo sottoscrittore. Questo endpoint deve essere in grado di accettare messaggi JSON trasferiti attraverso HTTP.

continuousErrorsForAbort

Descrizione

Il numero di aggiornamenti continui non riusciti prima che l'istanza di runtime smetta di inviare altri aggiornamenti al sottoscrittore.

Valore predefinito

5

timeoutInMillis

Descrizione

Il valore in millisecond dopo il quale il processo di invio scade durante l'invio di aggiornamenti a questo sottoscrittore.

Valore predefinito

1000

Valori validi

Abilitato

Descrizione

Determina se questo sottoscrittore è abilitato o disabilitato.

Valore predefinito

`True`

Valori validi

`True` o `False`

type

Descrizione

Il tipo di questo archivio dati. Quando questa opzione è selezionata, deve essere aggiunto il parametro **`className`** con il valore del nome completo di questa classe di implementazione. È necessario aggiungere **`classPath`** con l'URI del file JAR se non è presente nel percorso classi del runtime Interact.

Valore predefinito

`InteractLog`

Valori validi

`InteractLog, RelationalDB e Custom`

jmxInclusionCycles

Descrizione

L'intervallo nel multiplier di **`heartbeatPeriodInSecs`** in base a cui le statistiche JMX vengono inviate a questo sottoscrittore.

Valore predefinito

`5`

Valori validi

Interact | profile

Questa serie di proprietà di configurazione controlla diverse delle funzioni di presentazione dell'offerta facoltative, incluse la soppressione dell'offerta e la sovrascrittura del punteggio.

enableScoreOverrideLookup

Descrizione

Se questa proprietà è impostata su `True`, Interact carica i dati di sovrascrittura del punteggio da `scoreOverrideTable`, al momento della creazione di una sessione. Se è impostata su `False`, Interact non carica i dati di sovrascrittura del punteggio di marketing, durante la creazione di una sessione.

Se questa proprietà è impostata su true, è necessario configurare anche la proprietà `Interact | profile | Audience Levels | (Audience Level) | scoreOverrideTable`. È necessario definire la proprietà `scoreOverrideTable` solo per i livelli destinatario richiesti. Non specificando alcun valore di `scoreOverrideTable` per un livello destinatario si disabilita la tabella di sovrascrittura punteggio per quel livello destinatario.

Valore predefinito

`False`

Valori validi

True | False

enableOfferSuppressionLookup

Descrizione

Se questa proprietà è impostata su `True`, Interact carica i dati di soppressione dell'offerta da `offerSuppressionTable`, al momento della creazione di una sessione. Se è impostata su `False`, Interact non carica i dati di soppressione dell'offerta, durante la creazione di una sessione.

Se questa proprietà è impostata su true, è necessario configurare anche la proprietà `Interact | profile | Audience Levels | (Audience Level) | offerSuppressionTable`. È necessario definire la proprietà `enableOfferSuppressionLookup` solo per i livelli destinatario richiesti.

Valore predefinito

`False`

Valori validi

True | False

enableProfileLookup

Descrizione

In una nuova installazione di Interact, questa proprietà è obsoleta. In un'installazione aggiornata di Interact, questa proprietà è valida fino alla prima distribuzione.

Il comportamento di caricamento per una tabella utilizzata in un diagramma di flusso interattivo ma non associato nel canale interattivo. Se questa proprietà è impostata su `True`, Interact carica i dati del profilo da `profileTable`, al momento della creazione di una sessione.

Se questa proprietà è impostata su true, è necessario configurare anche la proprietà `Interact | profile | Audience Levels | (Audience Level) | profileTable`.

L'impostazione **Carica questi dati in memoria all'avvio di una sessione visite** nella procedura guidata del mapping della tabella del canale interattivo sovrascrive questa proprietà di configurazione.

Valore predefinito

`False`

Valori validi

True | False

defaultOfferUpdatePollPeriod

Descrizione

Il numero di secondi che il sistema attende prima di aggiornare le offerte predefinite nella cache dalla tabella delle offerte predefinite. Se questa proprietà è impostata su -1, il sistema non aggiorna le offerte predefinite nella cache una volta caricato nella cache l'elenco iniziale, all'avvio del server di runtime.

Valore predefinito

-1

Interact | profile | Audience Levels | [AudienceLevelName]

Questa serie di proprietà di configurazione consente di definire i nomi tabella richiesti per le funzioni Interact aggiuntive. La definizione del nome tabella è richiesta solo se si sta utilizzando la funzione associata.

Nuovo nome categoria

Descrizione

Il nome del livello destinatario.

scoreOverrideTable

Descrizione

Il nome della tabella che contiene le informazioni di sovrascrittura del punteggio per questo livello destinatario. Questa proprietà è applicabile, se `enableScoreOverrideLookup` è stato impostato su true. È necessario definire questa proprietà per i livelli destinatario per cui si desidera abilitare una tabella di sovrascrittura punteggio. Se non si dispone di una tabella di sovrascrittura punteggio per questo livello destinatario, è possibile lasciare questa proprietà non definita, anche se `enableScoreOverrideLookup` è impostato su true.

Interact cerca questa tabella nelle tabelle cliente a cui accedono i server di runtime Interact, definite dalle proprietà `prodUserDataSource`.

Se è stata definita la proprietà `schema` per questa origine dati, Interact antepone lo schema a questo nome tabella, ad esempio, `schema.UACI_ScoreOverride`. Se si immette un nome completo, ad esempio, `mySchema.UACI_ScoreOverride`, Interact non antepone il nome schema.

Valore predefinito

UACI_ScoreOverride

offerSuppressionTable

Descrizione

Il nome della tabella contenente le informazioni sulle soppressioni delle offerte per questo livello destinatario. È necessario definire questa proprietà per i livelli destinatario per cui si desidera abilitare una tabella di soppressioni offerta. Se non esiste alcuna tabella di soppressioni offerta per questo livello destinatario, è possibile lasciare non definita questa proprietà. Se la proprietà `enableOfferSuppressionLookup` è impostata su true, è necessario impostare questa proprietà su una tabella valida.

Interact cerca questa tabella nelle tabelle cliente a cui accedono i server di runtime, definite dalle proprietà `prodUserDataSource`.

Valore predefinito

UACI_BlackList

contactHistoryTable

Descrizione

Il nome della tabella di staging per i dati della cronologia dei contatti, per questo livello destinatario.

Questa tabella viene archiviata nelle tabelle dell'ambiente di runtime (systemTablesDataSource).

Se è stata definita la proprietà schema per questa origine dati, Interact antepone lo schema a questo nome tabella, ad esempio, schema.UACI_CHStaging. Se si immette un nome completo, ad esempio, mySchema.UACI_CHStaging, Interact non antepone il nome schema.

Se la registrazione della cronologia dei contatti è disabilitata, non è necessario impostare questa proprietà.

Valore predefinito

UACI_CHStaging

chOfferAttribTable

Descrizione

Il nome della tabella di attributi dell'offerta della cronologia dei contatti, per questo livello destinatario.

Questa tabella viene archiviata nelle tabelle dell'ambiente di runtime (systemTablesDataSource).

Se è stata definita la proprietà schema per questa origine dati, Interact antepone lo schema a questo nome tabella, ad esempio, schema.UACI_CHOfferAttrib. Se si immette un nome completo, ad esempio, mySchema.UACI_CHOfferAttrib, Interact non antepone il nome schema.

Se la registrazione della cronologia dei contatti è disabilitata, non è necessario impostare questa proprietà.

Valore predefinito

UACI_CHOfferAttrib

responseHistoryTable

Descrizione

Il nome della tabella di staging della cronologia delle risposte, per questo livello destinatario.

Questa tabella viene archiviata nelle tabelle dell'ambiente di runtime (systemTablesDataSource).

Se è stata definita la proprietà schema per questa origine dati, Interact antepone lo schema a questo nome tabella, ad esempio, schema.UACI_RHStaging. Se si immette un nome completo, ad esempio, mySchema.UACI_RHStaging, Interact non antepone il nome schema.

Se la registrazione della cronologia delle risposte è disabilitata, non è necessario impostare questa proprietà.

Valore predefinito

UACI_RHStaging

crossSessionResponseTable

Descrizione

Il nome della tabella per questo livello destinatario, richiesta per tracciamento della risposta delle sessioni incrociate nelle tabelle della cronologia dei contatti e delle risposte accessibili per la funzione di tracciamento delle risposte.

Se è stata definita la proprietà schema per questa origine dati, Interact antepone lo schema a questo nome tabella, ad esempio, schema.UACI_XSessResponse. Se si immette un nome completo, ad esempio, mySchema.UACI_XSessResponse, Interact non antepone il nome schema.

Se la registrazione della risposta delle sessioni incrociate è disabilitata, non è necessario impostare questa proprietà.

Valore predefinito

UACI_XSessResponse

userEventLoggingTable

Descrizione

Questo è il nome della tabella database utilizzata per registrare attività evento definite dall'utente. Gli eventi definiti dagli utenti nella scheda Eventi delle pagine di riepilogo del canale interattivo, nell'interfaccia Interact. La tabella database specificata in questo contesto contiene informazioni quali, ad esempio, l'ID evento, il nome, la frequenza con cui si è verificato l'evento, per questo livello destinatario, dall'ultima volta in cui è stata svuotata la cache delle attività evento e così via.

Se è stata definita la proprietà schema per questa origine dati, Interact antepone lo schema a questo nome tabella, ad esempio, schema.UACI_UserEventActivity. Se si immette un nome completo, ad esempio, mySchema.UACI_UserEventActivity, Interact non antepone il nome schema.

Valore predefinito

UACI_UserEventActivity

patternStateTable

Descrizione

Questo è il nome della tabella database utilizzata per la registrazione degli stati del pattern di evento, ad esempio se è stata soddisfatta o meno la condizione specificata per il pattern, se il pattern è scaduto oppure è stato disabilitato e così via.

Se è stata definita la proprietà schema per questa origine dati, Interact antepone lo schema a questo nome tabella, ad esempio, schema.UACI_EventPatternState. Se si immette un nome completo, ad esempio, mySchema.UACI_EventPatternState, Interact non antepone il nome schema.

Una patternStateTable è richiesta per ogni livello destinatario anche se non si utilizzano pattern di evento. La patternStateTable si basa sul ddl del valore UACI_EventPatternState incluso. Quello che segue, è un esempio in cui l'ID destinatario ha due componenti; ComponentNum e ComponentStr.

```
CREATE TABLE UACI_EventPatternState_Composite
(
   UpdateTime bigint NOT NULL,
   State varbinary(4000),
   ComponentNum bigint NOT NULL,
```

```
    ComponentStr nvarchar(50) NOT NULL,
    CONSTRAINT PK_CustomerPatternState_Composite PRIMARY KEY
    (ComponentNum,ComponentStr,UpdateTime)
)
```

Valore predefinito

UACI_EventPatternState

Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL

Questa serie di proprietà di configurazione consente di definire i nomi tabella richiesti per le funzioni Interact aggiuntive. La definizione del nome tabella è richiesta solo se si sta utilizzando la funzione associata.

enableOffersByRawSQL

Descrizione

Se questa proprietà è impostata su True, Interact abilita la funzione offersBySQL per questo livello destinatario, che consente di configurare il codice SQL da eseguire per creare una serie desiderata di offerte candidate in fase di runtime. Se è impostata su False, Interact non utilizzerà la funzione offersBySQL.

Se si imposta questa proprietà su true, è possibile configurare anche la proprietà Interact | profile | Audience Levels | (Audience Level) | Offers by Raw SQL | SQL Template per definire uno o più modelli SQL.

Valore predefinito

False

Valori validi

True | False

cacheSize

Descrizione

La dimensione della cache utilizzata per memorizzare i risultati delle query OfferBySQL. Si tenga presente che l'utilizzo di una cache può avere un impatto negativo, se i risultati delle query sono univoci per la maggior parte delle sessioni.

Valore predefinito

-1 (disattivato)

Valori validi

-1 | Valore

cacheLifeInMinutes

Descrizione

Se la cache è abilitata, questo valore indica il numero di minuti prima che il sistema cancelli il contenuto della cache per evitare l'obsolescenza.

Valore predefinito

-1 (disattivato)

Valori validi

-1 | Valore

defaultSQLTemplate

Descrizione

Il nome del modello SQL da utilizzare, se non ne viene specificato uno tramite chiamate all'API.

Valore predefinito

`Nessuno`

Valori validi

Nome modello SQL

dominio

Categoria configurazione

`Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL | (SQL Templates)`

Descrizione

Il nome che si desidera assegnare a questo modello query SQL. Immettere un nome descrittivo, che avrà significato quando si utilizza questo modello SQL nelle chiamate alle API. Si tenga presente che, se qui si utilizza un nome *identico* al nome definito nella casella del processo Elenco interazioni per un trattamento offerBySQL, verrà adottato l'SQL nella casella del processo, piuttosto che quello immesso qui.

Valore predefinito

Nessuno

SQL

Categoria configurazione

`Interact | profile | Audience Levels | [AudienceLevelName] | Offers by Raw SQL | (SQL Templates)`

Descrizione

Contiene la query SQL che questo modello dovrà richiamare. La query SQL può contenere riferimenti ai nomi di variabile che fanno parte dei dati di sessione del visitatore (profilo). Ad esempio, `select * from MyOffers where category = ${preferredCategory}` farebbe riferimento alla sessione contenente una variabile denominata `preferredCategory`.

Si dovrebbe configurare l'SQL in modo da interrogare tabelle di offerte specifiche, create durante la fase di progettazione per essere utilizzate da questa funzione. Si tenga presente che, in questo contesto, non sono supportate procedure memorizzate.

Valore predefinito

Nessuno

Interact | profile | Audience Levels | [AudienceLevelName] | SQL Template

Queste proprietà di configurazione consentono di definire uno o più modelli query SQL utilizzati dalla funzione offersBySQL di Interact.

dominio

Descrizione

Il nome che si desidera assegnare a questo modello query SQL. Immettere un nome descrittivo, che avrà significato quando si utilizza questo modello SQL nelle chiamate alle API. Si tenga presente che, se qui si utilizza un nome *identico* al nome definito nella casella del processo Elenco interazioni per un trattamento offerBySQL, verrà adottato l'SQL nella casella del processo, piuttosto che quello immesso qui.

Valore predefinito

Nessuno

SQL

Descrizione

Contiene la query SQL che questo modello dovrà richiamare. La query SQL può contenere riferimenti ai nomi di variabile che fanno parte dei dati di sessione del visitatore (profilo). Ad esempio, `select * from MyOffers where category = ${preferredCategory}` farebbe riferimento alla sessione contenente una variabile denominata `preferredCategory`.

Si dovrebbe configurare l'SQL in modo da interrogare tabelle di offerte specifiche, create durante la fase di progettazione per essere utilizzate da questa funzione. Si tenga presente che, in questo contesto, non sono supportate procedure memorizzate.

Valore predefinito

Nessuno

Interact | profile | Audience Levels | [AudienceLevelName | Profile Data Services | [DataSource]

Questa serie di proprietà di configurazione consente di definire i nomi tabella richiesti per le funzioni Interact aggiuntive. La definizione del nome tabella è richiesta solo se si sta utilizzando la funzione associata. La categoria Profile Data Services fornisce informazioni su un'origine dati integrata (denominata Database), creata per tutti i livelli destinatario e che è pre-configurata con una priorità di 100. Tuttavia, si può decidere di modificarla o disabilitarla. Questa categoria contiene anche un modello per ulteriori origini dati esterne. Quando si fa clic sul modello, denominato **External Data Services**, è possibile completare le impostazioni di configurazione descritte qui.

Nuovo nome categoria

Descrizione

(Non disponibile per la voce Database predefinita.) Il nome dell'origine dati che si sta definendo. Il nome immesso in questo campo deve essere univoco tra le origini dati per lo stesso livello destinatario.

Valore predefinito

Nessuno

Valori validi

È consentita qualsiasi stringa di testo.

enabled

Descrizione

Se la proprietà è impostata su `True`, questa origine dati è abilitata per il livello destinatario a cui è assegnata. Se è impostata su `False`, Interact non utilizza tale origine dati per questo livello destinatario.

Valore predefinito

`True`

Valori validi

True | False

className

Descrizione

(Non disponibile per la voce Database predefinita.) Il nome completo della classe origine dati che implementa `IInteractProfileDataService`.

Valore predefinito

Nessuno.

Valori validi

Una stringa che riporta un nome classe completo.

classPath

Descrizione

(Non disponibile per la voce Database predefinita.) Un'impostazione di configurazione facoltativa, che fornisce il percorso per caricare la classe di implementazione di questa origine dati. Se non si specifica questa proprietà, verrà utilizzato per impostazione predefinita il percorso classi del server delle applicazioni contenitore.

Valore predefinito

Non visualizzato, ma, se non si specifica alcun valore per questo campo, verrà utilizzato per impostazione predefinita il percorso classi del server delle applicazioni contenitore.

Valori validi

Una stringa che riporta il percorso classi.

priority

Descrizione

La priorità di questa origine dati nell'ambito di questo livello destinatario. Deve trattarsi di un valore univoco tra tutte le origini dati per ogni livello destinatario. (Cioè, se priority è impostata su 100 per un'origine dati, nessun altra origine dati nello stesso livello destinatario potrà avere un valore priority di 100.)

Valore predefinito

`100` per il Database predefinito, `200` per un'origine dati definita dall'utente.

Valori validi

È consentito qualsiasi numero intero non negativo.

Interact | offerserving

Queste proprietà di configurazione definiscono le proprietà di configurazione di apprendimento generiche. Se si sta utilizzando l'apprendimento integrato, per ottimizzare l'implementazione di apprendimento adottare le proprietà di configurazione per l'ambiente di progettazione.

offerTieBreakMethod

Descrizione

La proprietà `offerTieBreakMethod` definisce il comportamento di presentazione di un'offerta, nel caso due offerte abbiano punteggi equivalenti (collegati). Se si imposta questa proprietà sul relativo valore predefinito, cioè Random, Interact presenta una scelta casuale tra le offerte con punteggi equivalenti. Se si imposta questa configurazione su Newer Offer, Interact presenta l'offerta più recente (cioè, quella con l'ID offerta più elevato) prima di quella meno recente (con ID offerta inferiore), nel caso in cui i punteggi delle offerte siano equivalenti.

Nota:

Interact ha una funzione facoltativa che consente all'amministratore di configurare il sistema in modo che restituisca le offerte in ordine casuale, indipendentemente dal punteggio, impostando l'opzione `percentRandomSelection` (`Campaign | partitions | [partition_number] | Interact | learning | percentRandomSelection`). La proprietà `offerTieBreakMethod` qui descritta viene utilizzata solo quando `percentRandomSelection` è impostata su zero (disabilitata).

Valore predefinito

Random

Valori validi

Random | Newer Offer

optimizationType

Descrizione

La proprietà `optimizationType` definisce la possibilità che Interact utilizzi un motore di apprendimento come supporto alle assegnazioni di offerta. Se è impostata su `NoLearning`, Interact non utilizza l'apprendimento. Se è impostata su `BuiltInLearning`, Interact utilizza il motore di apprendimento Bayesiano sviluppato con Interact. Se è impostata su `ExternalLearning`, Interact utilizza un motore di apprendimento fornito dall'utente. Se si seleziona `ExternalLearning`, è necessario definire le proprietà `externalLearningClass` ed `externalLearningClassPath`.

Valore predefinito

NoLearning

Valori validi

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

Descrizione

Il numero massimo di millisecondi che il server di runtime attende per il completamento di un diagramma di flusso interattivo, prima di richiamare le offerte.

Valore predefinito

5000

treatmentCodePrefix

Descrizione

Il prefisso anteposto ai codici trattamento.

Valore predefinito

Non è specificato alcun valore predefinito.

effectiveDateBehavior

Descrizione

Stabilisce se Interact dovrebbe attenersi alla data di validità di un'offerta per filtrare le offerte presentate ad un visitatore. I valori includono:

- -1 indica ad Interact di ignorare la data di validità di un'offerta.

  0 indica ad Interact di utilizzare la data di validità per filtrare l'offerta, in modo che, se la data di validità dell'offerta è precedente o corrispondente alla data corrente, l'offerta verrà presentata ai visitatori.

  Se è impostato un valore **effectiveDateGracePeriod**, ci si baserà anche sul periodo di tolleranza per stabilire se presentare l'offerta.
- Qualsiasi numero intero positivo indica a Interact di utilizzare la data corrente più il valore specificato per questa proprietà, per stabilire se presentare l'offerta ai visitatori, in modo tale che, se la data di validità dell'offerta è precedente rispetto alla data corrente più il valore di questa proprietà, l'offerta viene presentata ai visitatori.

  Se è impostato un valore **effectiveDateGracePeriod**, ci si baserà anche sul periodo di tolleranza per stabilire se presentare l'offerta.

Valore predefinito

-1

effectiveDateGracePeriodOfferAttr

Descrizione

Specifica il nome dell'attributo personalizzato in una definizione di offerta che indica il periodo di tolleranza della data di validità. Ad esempio, si potrebbe configurare tale proprietà con un valore di `AltGracePeriod`. Si dovrebbero, quindi, definire le offerte con un attributo personalizzato, denominato `AltGracePeriod`, che verrà utilizzato per specificare il numero di giorni da calcolare come periodo di tolleranza con la proprietà **effectiveDateBehavior**.

Supponiamo che venga creato un nuovo modello dell'offerta con una data di validità di 10 giorni a partire dalla data corrente e che sia incluso un attributo personalizzato, denominato `AltGracePeriod`. Quando si crea un'offerta applicando questo modello, se si imposta il valore di `AltGracePeriod` su 14 giorni, l'offerta dovrebbe essere presentata ai visitatori, in quanto la data corrente rientra nel periodo di tolleranza della data di validità dell'offerta.

Valore predefinito

Vuoto

alwaysLogLearningAttributes

Descrizione

Indica se Interact dovrebbe scrivere informazioni relative agli attributi visitatore utilizzati dal modulo di apprendimento nei file di log. Si tenga presente che, impostando questo valore su true, si potrebbero influenzare le prestazioni di apprendimento e le dimensioni dei file di log.

Valore predefinito

False

Interact | offerserving | Built-in Learning Config

Queste proprietà di configurazione definiscono le impostazioni di scrittura nel database per l'apprendimento integrato. Per ottimizzare l'implementazione di apprendimento, utilizzare le proprietà di configurazione per l'ambiente di progettazione.

version

Descrizione

È possibile selezionare 1 o 2. Versione 1 è la versione della configurazione di base, che non utilizza parametri per impostare limiti per thread e record. Versione 2 è la versione della configurazione avanzata, che consente di impostare parametri relativi a thread e record, per migliorare le prestazioni. Questi parametri eseguono l'aggregazione e la cancellazione quando vengono raggiunti i limiti per tali parametri.

Valore predefinito

1

insertRawStatsIntervalInMinutes

Descrizione

Il numero di minuti che il modulo di apprendimento Interact attende, prima di inserire altre righe nelle tabelle di staging di apprendimento. Potrebbe essere necessario modificare questo periodo in base alla quantità di dati elaborati dal modulo di apprendimento nell'ambiente.

Valore predefinito

5

Valori validi

Un numero intero positivo

aggregateStatsIntervalInMinutes

Descrizione

Il numero di minuti che il modulo di apprendimento Interact attende tra le aggregazioni di dati nelle tabelle stats di apprendimento. Potrebbe essere necessario modificare questo periodo in base alla quantità di dati elaborati dal modulo di apprendimento nell'ambiente.

Valore predefinito

15

Valori validi

Un numero intero superiore a zero.

autoAdjustPercentage

Descrizione

Il valore che determina la percentuale di dati che l'esecuzione dell'aggregazione tenta di elaborare sulla base delle metriche della precedente esecuzione. Per impostazione predefinita, questo valore è impostato su zero, che indica che il programma di aggregazione elabora tutti i record di staging e che questa funzione di aggiustamento automatico è disabilitata.

Valore predefinito

0

Valori validi

Un numero compreso tra 0 e 100.

enableObservationModeOnly

Descrizione

Se impostato su `True`, consente di una modalità di apprendimento in cui Interact raccoglie i dati per l'apprendimento senza utilizzare tali dati per raccomandazioni o per un arbitrato di offerta. Questo consente di effettuare l'apprendimento automatico in modalità di avvio, fino a quando non si stabilisce che sono stati raccolti dati sufficienti per le raccomandazioni.

Valore predefinito

False

Valori validi

True | False

excludeAbnormalAttribute

Descrizione

L'impostazione che determina se contrassegnare tali attributi come non validi. Se viene impostata su `IncludeAttribute`, gli attributi anomali sono inclusi e non sono contrassegnati come non validi. Se viene impostata su `ExcludeAttribute`, gli attributi anomali sono esclusi e sono contrassegnati come non validi.

Valore predefinito

IncludeAttribute

Valori validi

IncludeAttribute | ExcludeAttribute

Interact | offerserving | Built-in Learning Config | Parameter Data | [parameterName]

Queste proprietà di configurazione definiscono qualsiasi parametro per il modulo di apprendimento esterno.

numberOfThreads

Descrizione

Il numero massimo di thread che il programma di aggregazione di apprendimento utilizza per elaborare i dati. Un valore valido è un numero intero positivo e non dovrebbe superare il numero massimo connessioni configurate nell'origine dati di apprendimento. Questo parametro viene utilizzato solo dal programma di aggregazione versione 2.

Valore predefinito

10

maxLogTimeSpanInMin

Descrizione

Se viene selezionato il programma di aggregazione versione 1, è possibile elaborare i record di staging in iterazioni, onde evitare batch di database eccessivamente grandi. In questo caso, tali record di staging vengono elaborati in blocchi; iterazione per iterazione in un singolo ciclo di aggregazione. Il valore di questo parametro specifica il massimo intervallo di tempo dei record di staging che il programma di aggregazione tenta di elaborare in ciascuna iterazione. Questo intervallo di tempo si basa sul campo LogTime, associato a ciascun record di staging e vengono elaborati solo i record il cui LogTime rientra nella prima finestra di tempo. Un valore valido è un numero intero non negativo. Se il valore è 0, non vi è alcun limite, il che significa che tutti i record di staging vengono elaborati in una singola iterazione.

Valore predefinito

0

maxRecords

Descrizione

Se viene selezionato il programma di aggregazione versione 2, è possibile elaborare i record di staging in iterazioni, onde evitare batch di database eccessivamente grandi. In questo caso, tali record di staging vengono elaborati in blocchi; iterazione per iterazione in un singolo ciclo di aggregazione. Il valore di questo parametro specifica il numero massimo di record di staging che il programma di aggregazione tenta di elaborare in ciascuna iterazione. Un valore valido è un numero intero non negativo. Se il valore è 0, non vi è alcun limite, il che significa che tutti i record di staging vengono elaborati in una singola iterazione.

Valore predefinito

0

value

Descrizione

Il valore per qualsiasi parametro, richiesto dalla classe per un modulo di apprendimento integrato.

Valore predefinito

Non è specificato alcun valore predefinito.

Interact | offerserving | External Learning Config

Queste proprietà di configurazione definiscono le impostazioni della classe per un modulo di apprendimento esterno scritto dall'utente utilizzando l'API di apprendimento.

class

Descrizione

Se la proprietà `optimizationType` è impostata su `ExternalLearning`, impostare `externalLearningClass` sul nome classe del motore di apprendimento esterno.

Valore predefinito

Non è specificato alcun valore predefinito.

Disponibilità

Questa proprietà è applicabile solo se la proprietà `optimizationType` è impostata su `ExternalLearning`.

classPath

Descrizione

Se la proprietà `optimizationType` è impostata su `ExternalLearning`, impostare `externalLearningClass` sul percorso classi del motore di apprendimento esterno.

Il percorso classi deve fare riferimento ai file jar sul server dell'ambiente di runtime. Se si sta utilizzando un gruppo di server e tutti i server di runtime stanno utilizzando lo stesso componente Marketing Platform, ogni server deve avere una copia del file jar nella stessa ubicazione. Il percorso classi deve essere formato dalle ubicazioni assolute dei file jar, separate dal delimitatore di percorso del sistema operativo del server dell'ambiente di runtime, ad esempio un punto e virgola (;) nei sistemi Windows e due punti (:) nei sistemi UNIX. Non sono accettate directory contenenti file di classe. Ad esempio, su un sistema Unix: `/path1/file1.jar:/path2/file2.jar`.

Questo percorso classi non deve superare i 1024 caratteri di lunghezza. È possibile utilizzare il file manifest in un file .jar per specificare altri file .jar, in modo che sia presente un solo file .jar nel percorso classi

Valore predefinito

Non è specificato alcun valore predefinito.

Disponibilità

Questa proprietà è applicabile solo se la proprietà `optimizationType` è impostata su `ExternalLearning`.

Interact | offerserving | External Learning Config | Parameter Data | [parameterName]

Queste proprietà di configurazione definiscono qualsiasi parametro per il modulo di apprendimento esterno.

value

Descrizione

Il valore per qualsiasi parametro richiesto dalla classe per un modulo di apprendimento esterno.

Valore predefinito

Non è specificato alcun valore predefinito.

Esempio

Se il modulo di apprendimento esterno richiede un percorso ad un'applicazione di risoluzione di algoritmi, creare una categoria di parametri chiamata `solverPath` e definire la proprietà `value` come percorso all'applicazione.

Interact | offerserving | Constraints

Queste proprietà di configurazione definiscono le restrizioni sul processo che fornisce l'offerta.

maxOfferAllocationInMemoryPerInstance

Descrizione

La dimensione di un blocco di offerte. Interact trattiene un pool di offerte in memoria così il sistema non deve interrogare il database ogni volta che viene restituita un'offerta. Ogni volta che viene restituita un'offerta, il pool viene regolato di conseguenza. Quando il è vuoto, Interact prende un altro blocco di offerte per riempirlo.

Valore predefinito

1000

Valori validi

Un numero intero maggiore di 0.

maxDistributionPerIntervalPerInstanceFactor

Descrizione

La percentuale di restrizione per una data allocazione di offerta perché un server di runtime supporti la distribuzione sui server di runtime.

Valore predefinito

100

Valori validi

Un numero intero compreso tra 0 e 100.

constraintCleanupIntervalInDays

Descrizione

La frequena di ripulitura dei conteggi di disabilitazione dalla tabella UACI_OfferCount. Un valore inferiore a 1 disabilita questa funzione.

Valore predefinito

7

Valori validi

Un numero intero maggiore di 0.

Interact | services

Le proprietà di configurazione di questa categoria definiscono le impostazioni per tutti i servizi che gestiscono la raccolta dei dati e delle statistiche della cronologia dei contatti e delle risposte per il reporting e la scrittura nelle tabelle di sistema dell'ambiente di runtime.

externalLoaderStagingDirectory

Descrizione

Questa proprietà definisce l'ubicazione della directory di staging per un programma di utilità per il caricamento del database.

Valore predefinito

Non è specificato alcun valore predefinito.

Valori validi

Un percorso relativo alla directory di installazione di Interact o un percorso assoluto ad una directory di staging.

Se si abilita un programma di utilità per il caricamento del database, è necessario impostare la proprietà `cacheType` nelle categorie `contactHist` e `responstHist` su `External Loader File`.

Interact | services | contactHist

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il servizio che raccoglie i dati per le tabelle di staging della cronologia dei contatti.

enableLog

Descrizione

Se la proprietà è impostata su `true`, abilita il servizio che raccoglie i dati per la registrazione dei dati della cronologia dei contatti. Se il valore è `false`, non viene raccolto alcun dato.

Valore predefinito

True

Valori validi

True | False

cacheType

Descrizione

Definisce se i dati raccolti per la cronologia dei contatti vengono conservati in memoria (`Memory Cache`) o in un file (`External Loader File`). È possibile

utilizzare `External Loader File` solo se Interact è stato configurato per utilizzare un programma di utilità di caricamento del database.

Se si seleziona `Memory Cache`, utilizzare le impostazioni della categoria `cache`. Se si seleziona `External Loader File`, utilizzare le impostazioni della categoria `fileCache`.

Valore predefinito

Memory Cache

Valori validi

Memory Cache | External Loader File

Interact | services | contactHist | cache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie i dati per la tabella di staging della cronologia dei contatti.

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database i dati della cronologia dei contatti raccolti.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | contactHist | fileCache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie i dati della cronologia dei contatti se si utilizza un programma di utilità di caricamento del database.

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database i dati della cronologia dei contatti raccolti.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | defaultedStats

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il servizio che raccoglie le statistiche relative al numero di volte in cui è stata utilizzata la stringa predefinita per il punto di interazione.

enableLog

Descrizione

Se la proprietà è impostata su `true`, abilita il servizio che raccoglie le statistiche relative al numero di volte in cui è stata utilizzata la stringa predefinita per il punto di interazione, per la tabella `UACI_DefaultedStat`. Se il valore è `false`, non vengono raccolte statistiche relative alla stringa predefinita.

Se non si sta utilizzando il reporting di IBM, è possibile impostare questa proprietà su `false`, dato che la raccolta dei dati non è richiesta.

Valore predefinito

True

Valori validi

True | False

Interact | services | defaultedStats | cache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie le statistiche relative al numero di volte in cui è stata utilizzata la stringa predefinita per il punto di interazione.

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database le statistiche relative alla stringa predefinita raccolte.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | eligOpsStats

Le proprietà di configurazione in questa categoria definiscono le impostazioni per il servizio che scrive le statistiche per le offerte idonee.

enableLog

Descrizione

Se la proprietà è impostata su `true`, abilita il servizio che raccoglie le statistiche per le offerte idonee. se il valore è `false`, nnon vengono raccolte statistiche per le offerte idonee.

Se non si sta utilizzando il reporting di IBM, è possibile impostare questa proprietà su `false`, dato che la raccolta dei dati non è richiesta.

Valore predefinito

True

Valori validi

True | False

Interact | services | eligOpsStats | cache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie le statistiche per le offerte idonee.

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database le statistiche relative all'offerta idonea raccolte.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | eventActivity

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il servizio che raccoglie le statistiche per l'attività evento.

enableLog

Descrizione

Se la proprietà è impostata su `true`, abilita il servizio che raccoglie le statistiche per l'attività evento. Se il valore è `false`, non vengono raccolte statistiche relative all'evento.

Se non si sta utilizzando il reporting di IBM, è possibile impostare questa proprietà su `false`, dato che la raccolta dei dati non è richiesta.

Valore predefinito

True

Valori validi

True | False

Interact | services | eventActivity | cache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della per il servizio che raccoglie le statistiche per l'attività evento.

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database le statistiche relative all'attività evento raccolte.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | eventPattern

Le proprietà di configurazione nella categoria `eventPattern` definiscono le impostazioni per il servizio che raccoglie le statistiche sull'attività del pattern di evento.

persistUnknownUserStates

Descrizione

Determina se gli stati del pattern di evento per un ID del destinatario sconosciuto (visitatore) vengono conservati nel database. Per impostazione predefinita, quando termina una sessione, gli stati di tutti i pattern di evento aggiornati, che sono associati all'ID del destinatario del visitatore vengono memorizzati nel database, se l'ID del destinatario è noto (ossia è possibile trovare il profilo del visitatore nell'origine dati del profilo).

La proprietà `persistUnknownUserStates` determina cosa accade se l'ID del destinatario non è noto. Per impostazione predefinita, questa proprietà è impostata su `False` e, per ID del destinatario sconosciuti, gli stati del pattern di evento vengono eliminati alla fine della sessione.

Se si imposta questa proprietà su `True`, gli stati del pattern di evento di utenti sconosciuti (il cui profilo non è reperibile nel servizio dati del profilo configurato) verranno resi persistenti.

Valore predefinito

False

Valori validi

True | False

mergeUnknowUserInSessionStates

Descrizione

Determina in che modo vengono conservati gli stati del pattern di evento per ID del destinatario sconosciuti (visitatori). Se l'ID del destinatario viene

commutato nel mezzo di una sessione, Interact tenta di caricare gli stati del pattern di evento salvati per il nuovo ID del destinatario dalla tabella del database. Quando l'ID del destinatario era precedentemente sconosciuto e la proprietà `mergeUnknowUserInSessionStates` viene impostata su `True`, le attività evento dell'utente appartenenti al precedente ID del destinatario nella stessa sessione verranno unite nel nuovo ID del destinatari.

Valore predefinito

False

Valori validi

True | False

enableUserEventLog

Descrizione

Determina se le attività evento dell'utente vengono registrate nel database.

Valore predefinito

False

Valori validi

True | False

Interact | services | eventPattern | userEventCache

Le proprietà di configurazione nella categoria `userEventCache` definiscono le impostazioni che stabiliscono quando l'attività evento viene spostata dalla cache per la memorizzazione permanente nel database.

threshold

Descrizione

Stabilisce il numero massimo di stati del pattern di evento che può essere memorizzato nella cache degli stati del pattern di evento. Quando si raggiunge il limite, gli stati utilizzati meno di recente vengono cancellati dalla cache.

Valore predefinito

100

Valori validi

Il numero desiderato di stati del pattern di evento da conservare nella cache.

insertPeriodInSecs

Descrizione

Stabilisce la durata massima, in secondi, dell'accodamento in memoria delle attività evento dell'utente. Quando viene raggiunto il limite di tempo specificato da questa proprietà, tali attività vengono memorizzate in modo permanente nel database.

Valore predefinito

3600 (60 minuti)

Valori validi

Il numero di secondi desiderato.

Interact | services | eventPattern | advancedPatterns

Le proprietà di configurazione in questa categoria controllano se è abilitata l'integrazione con Interact Advanced Patterns e definiscono gli intervalli di timeout per le connessioni con Interact Advanced Patterns.

enableAdvancedPatterns

Descrizione

Se questa proprietà è impostata su `true`, abilita l'integrazione con Interact Advanced Patterns. Se il valore è `false`, l'integrazione non è abilitata. Se l'integrazione era precedentemente abilitata, Interact utilizzerà i più recenti stati del pattern ricevuti da Interact Advanced Patterns.

Valore predefinito

True

Valori validi

True | False

connectionTimeoutInMilliseconds

Descrizione

Il tempo massimo che si può impiegare per stabilire una connessione HTTP dall'ambiente in tempo reale Interact a Interact Advanced Patterns. Se la richiesta scade, Interact utilizza gli ultimi dati salvati, ricavati dai pattern.

Valore predefinito

30

readTimeoutInMilliseconds

Descrizione

Una volta stabilita una connessione HTTP tra l'ambiente in tempo reale Interact e Interact Advanced Patterns e una volta inviata una richiesta a Interact Advanced Patterns per richiamare lo stato di un pattern di evento, il tempo massimo che si può impiegare per ricevere i dati. Se la richiesta scade, Interact utilizza gli ultimi dati salvati, ricavati dai pattern.

Valore predefinito

100

connectionPoolSize

Descrizione

La dimensione del pool di connessioni HTTP per la comunicazione tra l'ambiente in tempo reale Interact e Interact Advanced Patterns.

Valore predefinito

10

Interact | services | eventPattern | advancedPatterns | autoReconnect

Le proprietà di configurazione in questa categoria specificano parametri per la funzione di riconnessione automatica nell'integrazione con Interact Advanced Patterns.

abilita

Descrizione

Determina se il sistema si riconnette automaticamente, in caso di problemi di connessione tra l'ambiente in tempo reale Interact real e Interact Advanced Patterns. Il valore predefinito **True** abilita questa funzione.

Valore predefinito

True

Valori validi

True | False

durationInMinutes

Descrizione

Questa proprietà specifica l'intervallo di tempo, in minuti, durante il quale il sistema valuta i problemi di connessione ripetuti tra l'ambiente in tempo reale Interact e Interact Advanced Patterns.

Valore predefinito

10

numberOfFailuresBeforeDisconnect

Descrizione

Questa proprietà specifica il numero di errori di connessione consentiti durante il periodo di tempo specificato, prima che il sistema si disconnetta automaticamente da Interact Advanced Patterns.

Valore predefinito

3

consecutiveFailuresBeforeDisconnect

Descrizione

Determina se la funzione di riconnessione automatica valuterà esclusivamente gli errori consecutivi di connessione tra l'ambiente in tempo reale Interact e Interact Advanced Patterns. Se si imposta questo valore su **False**, verranno valutati tutti gli errori entro l'intervallo di tempo specificato.

Valore predefinito

True

sleepBeforeReconnectDurationInMinutes

Descrizione

Il sistema attende il numero di minuti specificato in questa proprietà prima di riconnettersi, dopo che si è disconnesso in seguito ad errori ripetuti, come specificato nelle altre proprietà di questa categoria.

Valore predefinito

5

sendNotificationAfterDisconnect

Descrizione

Questa proprietà determina se il sistema invierà una notifica e-mail al verificarsi di un errore di connessione. Il messaggio di notifica include il nome dell'istanza in tempo reale di Interact per cui si è verificato l'errore e la quantità di tempo prima che avvenga la riconnessione, come specificato nella proprietà **sleepBeforeReconnectDurationInMinutes**. Il valore predefinito **True** abilita l'invio delle notifiche.

Valore predefinito

True

Interact | services | customLogger

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il servizio che raccoglie i dati personalizzati da scrivere in una tabella (un evento che utilizza il parametro evento `UACICustomLoggerTableName`).

enableLog

Descrizione

Se la proprietà è impostata su `true`, abilita la funzione log personalizzato nella tabella. Se il valore è `false`, il parametro evento `UACICustomLoggerTableName` non ha effetto.

Valore predefinito

True

Valori validi

True | False

Interact | services | customLogger | cache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie i dati personalizzati in una tabella (un evento che utilizza il parametro evento `UACICustomLoggerTableName`).

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database i dati personalizzati raccolti.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | responseHist

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il servizio che scrive nelle tabelle di staging della cronologia delle risposte.

enableLog

Descrizione

Se la proprietà è impostata su `true`, abilita il servizio che scrive nelle tabelle di staging della cronologia delle risposte. Se il valore è `false`, non viene scritto alcun dato nelle tabelle di staging della cronologia delle risposte.

La tabella di staging della cronologia delle risposte è definita dalla proprietà `responseHistoryTable` per il livello destinatario. Il valore predefinito è `UACI_RHStaging`.

Valore predefinito

True

Valori validi

True | False

cacheType

Descrizione

Definisce se la cache è conservata in memoria o in un file. È possibile utilizzare `External Loader File` solo se Interact è stato configurato per utilizzare un programma di utilità di caricamento del database.

Se si seleziona `Memory Cache`, utilizzare le impostazioni della categoria `cache`. Se si seleziona `External Loader File`, utilizzare le impostazioni della categoria `fileCache`.

Valore predefinito

Memory Cache

Valori validi

Memory Cache | External Loader File

actionOnOrphan

Descrizione

Questa impostazione determina cosa fare con eventi risposta per cui non esistono eventi contatto corrispondenti. Se la proprietà è impostata su `NoAction`, l'evento risposta viene elaborato come se l'evento contatto corrispondente fosse stato inviato. Se la proprietà è impostata su `Warning`, l'evento risposta viene elaborato come se l'evento contatto corrispondente fosse stato inviato, ma viene scritto un messaggio di avvertenza in

`interact.log`. Se la proprietà è impostata su `Skip`, l'evento risposta non viene elaborato ed un messaggio di errore viene scritto in `interact.log`. L'impostazione selezionata qui è attiva indipendentemente dal fatto che sia abilitata la cronologia delle risposte.

Valore predefinito

NoAction

Valori validi

NoAction | Warning | Skip

Interact | services | responseHist | cache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie i dati della cronologia delle risposte.

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database i dati della cronologia delle risposte raccolti.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | response Hist | responseTypeCodes

Le proprietà di configurazione in questa categoria definiscono le impostazioni per il servizio di cronologia delle risposte.

Nuovo nome categoria

Descrizione

Il nome del codice tipo di risposta.

code

Descrizione

Il codice personalizzato per il tipo di risposta.

Valore predefinito

Il codice personalizzato aggiunto nella tabella UA_UsrResponseType.

action

Descrizione

L'azione corrispondente al codice tipo di risposta personalizzato.

L'azione definita per l'evento a cui questo èinviato sovrascrive l'azione definita qui. Pertanto, se viene inviato un evento logAccept senza

responseTypeCode, questo evento viene trattato come un evento di accettazione. Se viene inviato un evento logAccept con un responseTypeCode esistente in questa configurazione, viene utilizzata l'azione configurata per determinare se questo è un evento di accettazione. Se viene inviato un evento logAccept con un responseTypeCode non esistente in questa configurazione, questo evento non viene trattato come un evento di accettazione. Quando un evento viene trattato come evento di accettazione, le statistiche di apprendimento vengono aggiornate di conseguenza, se l'apprendimento è abilitato. Le regole di espressione dell'offerta, se presenti, vengono valutate in base all'accettazione di questa offerta.

Valore predefinito

Nessuno

Valori validi

LogAccept | LogReject | None

Interact | services | responseHist | fileCache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie i dati della cronologia delle risposte se si utilizza un programma di utilità di caricamento del database.

threshold

Descrizione

Il numero di record accumulati prima che Interact li scriva nel database.

`responseHist` - La tabella definita dalla proprietà `responseHistoryTable` per il livello destinatario. Il valore predefinito è `UACI_RHStaging`.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nel database.

Valore predefinito

3600

Interact | services | crossSessionResponse

Le proprietà di configurazione di questa categoria definiscono le impostazioni generali per il servizio crossSessionResponse ed il processo xsession. È necessario configurare queste impostazioni solo se si utilizza il tracciamento della risposta delle sessioni incrociate di Interact.

enableLog

Descrizione

Se la proprietà è impostata su `true`, abilita il servizio crossSessionResponse e Interact scrive i dati nelle tabelle di staging del tracciamento della risposta delle sessioni incrociate. Se il valore è `false`, disabilita il servizio crossSessionResponse.

Valore predefinito

False

xsessionProcessIntervalInSecs

Descrizione

Il numero di secondi tra le esecuzioni del processo xsession. Questo processo sposta i dati dalle tabelle di staging del tracciamento della risposta delle sessioni incrociate alla tabella di staging della cronologia delle risposte e al modulo di apprendimento integrato.

Valore predefinito

180

Valori validi

Un numero intero superiore a zero.

purgeOrphanResponseThresholdInMinutes

Descrizione

Il numero di minuti che il servizio crossSessionResponse attende prima di contrassegnare le eventuali risposte che non corrispondono ai contatti delle tabelle della cronologia dei contatti e delle risposte.

Se una risposta non ha corrispondenza nelle tabelle della cronologia dei contatti e delle risposte, dopo un numero di minuti corrispondente a `purgeOrphanResponseThresholdInMinutes`, Interact contrassegna la risposta con un valore di -1 nella colonna `Segna` della tabella di staging xSessResponse. È possibile, quindi, mettere in corrispondenza manualmente o eliminare queste risposte.

Valore predefinito

180

Interact | services | crossSessionResponse | cache

Le proprietà di configurazione di questa categoria definiscono le impostazioni della cache per il servizio che raccoglie i dati delle risposte delle sessioni incrociate.

threshold

Descrizione

Il numero di record accumulati prima che il servizio flushCacheToDB scriva nel database i dati delle risposte delle sessioni incrociate raccolti.

Valore predefinito

100

insertPeriodInSecs

Descrizione

Il numero di secondi tra le scritture forzate nella tabella `XSessResponse`.

Valore predefinito

3600

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

Le proprietà di questa sezione definiscono in che modo il tracciamento della risposta delle sessioni incrociate mette in corrispondenza i codici trattamento con la cronologia dei contatti e delle risposte.

SQL

Descrizione

Questa proprietà definisce se Interact utilizza SQL generato dal sistema o SQL personalizzato, definito nella proprietà `OverrideSQL`.

Valore predefinito

Use System Generated SQL

Valori validi

Use System Generated SQL | Override SQL

OverrideSQL

Descrizione

Se non si utilizza il comando SQL predefinito per mettere in corrispondenza il codice trattamento con la cronologia dei contatti e delle risposte, immettere qui l'SQL o la procedura memorizzata.

Questo valore viene ignorato se la proprietà `SQL` è impostata su `Use System Generated SQL`.

Valore predefinito

useStoredProcedure

Descrizione

Se la proprietà è impostata su true, `OverrideSQL` deve contenere un riferimento ad una procedura memorizzata che mette in corrispondenza il codice trattamento con la cronologia dei contatti e delle risposte.

Se impostata su false, la proprietà `OverrideSQL`, se utilizzata, deve essere una query SQL.

Valore predefinito

false

Valori validi

true | false

Tipo

Descrizione

Il TrackingCodeType associato, definito nella tabella `UACI_TrackingType` nelle tabelle dell'ambiente di runtime. A meno che non venga revisionata la tabella `UACI_TrackingType`, il valore di `Type` deve essere 1.

Valore predefinito

1

Valori validi

Un numero intero definito nella tabella UACI_TrackingType.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

Le proprietà di questa sezione definiscono in che modo il tracciamento della risposta delle sessioni incrociate mette in corrispondenza i codici offerta con la cronologia dei contatti e delle risposte.

SQL

Descrizione

Questa proprietà definisce se Interact utilizza SQL generato dal sistema o SQL personalizzato, definito nella proprietà OverrideSQL.

Valore predefinito

Use System Generated SQL

Valori validi

Use System Generated SQL | Override SQL

OverrideSQL

Descrizione

Se non si utilizza il comando SQL predefinito per mettere in corrispondenza il codice offerta con la cronologia dei contatti e delle risposte, immettere qui l'SQL o la procedura memorizzata.

Questo valore viene ignorato se la proprietà SQL è impostata su Use System Generated SQL.

Valore predefinito

useStoredProcedure

Descrizione

Se questa proprietà è impostata su true, OverrideSQL deve contenere un riferimento ad una procedura memorizzata che mette in corrispondenza il codice offerta con la cronologia dei contatti e delle risposte.

Se impostata su false, la proprietà OverrideSQL, se utilizzata, deve essere una query SQL.

Valore predefinito

false

Valori validi

true | false

Tipo

Descrizione

Il TrackingCodeType associato, definito nella tabella UACI_TrackingType nelle tabelle dell'ambiente di runtime. A meno che non si revisioni la tabella UACI_TrackingType, il valore di Type deve essere 2.

Valore predefinito

2

Valori validi

Un numero intero definito nella tabella UACI_TrackingType.

Interact | services | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

Le proprietà di questa sezione definiscono in che modo il tracciamento della risposta delle sessioni incrociate mette in corrispondenza un codice alternativo definito dall'utente con la cronologia dei contatti e delle risposte.

Nome

Descrizione

Questa proprietà definisce il nome del codice alternativo. Deve corrispondere al valore Nome nella tabella UACI_TrackingType nelle tabelle dell'ambiente di runtime.

Valore predefinito

OverrideSQL

Descrizione

Il comando SQL o la procedura memorizzata per mettere in corrispondenza il codice alternativo con la cronologia dei contatti e delle risposte in base al codice offerta o al codice trattamento.

Valore predefinito

useStoredProcedure

Descrizione

Se impostata su true, la proprietà OverrideSQL deve contenere un riferimento ad una procedura memorizzata che mette in corrispondenza il codice alternativo con la cronologia dei contatti e delle risposte.

Se impostata su false, la proprietà OverrideSQL, se utilizzata, deve essere una query SQL.

Valore predefinito

false

Valori validi

true | false

Tipo

Descrizione

Il TrackingCodeType associato, definito nella tabella UACI_TrackingType nelle tabelle dell'ambiente di runtime.

Valore predefinito

3

Valori validi

Un numero intero definito nella tabella UACI_TrackingType.

Interact | services | threadManagement | contactAndResponseHist

Le proprietà di configurazione di questa categoria definiscono le impostazioni di gestione dei thread per i servizi che raccolgono i dati per le tabelle di staging della cronologia dei contatti e delle risposte.

corePoolSize

Descrizione

Il numero di thread da conservare nel pool, anche se sono inattivi, per la raccolta dei dati della cronologia dei contatti e delle risposte.

Valore predefinito

5

maxPoolSize

Descrizione

Il numero massimo di thread da conservare nel pool per la raccolta dei dati della cronologia dei contatti e delle risposte.

Valore predefinito

5

keepAliveTimeSecs

Descrizione

Quando il numero di thread è maggiore del core, questo è il tempo massimo in cui i thread inattivi, in eccesso, attenderanno nuove attività prima di terminare, per la raccolta dei dati della cronologia dei contatti e delle risposte.

Valore predefinito

5

queueCapacity

Descrizione

La dimensione della coda utilizzata dal pool di thread per la raccolta dei dati della cronologia dei contatti e delle risposte.

Valore predefinito

1000

termWaitSecs

Descrizione

Alla chiusura del server di runtime, questo è il numero di secondi di attesa che i thread dei servizi completino la raccolta dei dati della cronologia dei contatti e delle risposte.

Valore predefinito

5

Interact | services | threadManagement | allOtherServices

Le proprietà di configurazione di questa categoria definiscono le impostazioni della gestione thread per i servizi che raccolgono le statistiche di idoneità delle offerte, le statistiche delle attività evento, le statistiche di utilizzo della stringa predefinita e il log personalizzato per i dati della tabella.

corePoolSize

Descrizione

Il numero di thread da conservare nel pool, anche se sono inattivi, per i servizi che raccolgono le statistiche di idoneità delle offerte, le statistiche delle attività evento, le statistiche di utilizzo della stringa predefinita e il log personalizzato per i dati della tabella.

Valore predefinito

5

maxPoolSize

Descrizione

Il numero massimo di thread da conservare nel pool, per i servizi che raccolgono le statistiche di idoneità delle offerte, le statistiche delle attività evento, le statistiche di utilizzo della stringa predefinita e il log personalizzato per i dati della tabella.

Valore predefinito

5

keepAliveTimeSecs

Descrizione

Quando il numero di thread è maggiore del core, questo è il periodo di tempo massimo in cui i thread inattivi, in eccesso, attendono nuove attività prima di terminare, per i servizi che raccolgono le statistiche di idoneità delle offerte, le statistiche delle attività evento, le statistiche di utilizzo della stringa predefinita e il log personalizzato per i dati della tabella.

Valore predefinito

5

queueCapacity

Descrizione

La dimensione della coda utilizzata dal pool di thread per i servizi che raccolgono le statistiche di idoneità delle offerte, le statistiche delle attività evento, le statistiche di utilizzo della stringa predefinita e il log personalizzato per i dati della tabella.

Valore predefinito

1000

termWaitSecs

Descrizione

Alla chiusura del server di runtime, questo è il numero di secondi di attesa per il completamento dei thread di servizio, per i servizi che raccolgono le statistiche di idoneità delle offerte, le statistiche delle attività evento, le statistiche di utilizzo della stringa predefinita e il log personalizzato per i dati della tabella.

Valore predefinito

5

Interact | services | threadManagement | flushCacheToDB

Le proprietà di configurazione di questa categoria definiscono le impostazioni di gestione dei thread, per i thread che scrivono i dati raccolti nella cache, per le tabelle database dell'ambiente di runtime.

corePoolSize

Descrizione

Il numero di thread da conservare nel pool per i thread pianificati che scrivono i dati memorizzati nella cache nell'archivio dati.

Valore predefinito

5

maxPoolSize

Descrizione

Il numero massimo di thread da conservare nel pool per i thread pianificati che scrivono i dati memorizzati nella cache nell'archivio dati.

Valore predefinito

5

keepAliveTimeSecs

Descrizione

Quando il numero di thread è maggiore del core, questo è il periodo di tempo massimo in cui i thread inattivi, in eccesso, attenderanno nuove attività prima di terminare, per i thread pianificati che scrivono i dati memorizzati nella cache nell'archivio dati.

Valore predefinito

5

queueCapacity

Descrizione

La dimensione della coda utilizzata dal pool di thread per i thread pianificati che scrivono i dati memorizzati nella cache nell'archivio dati.

Valore predefinito

1000

termWaitSecs

Descrizione

Alla chiusura del server di runtime, questo è il numero di secondi di attesa che i thread dei servizi vengano completati, per i thread pianificati che scrivono i dati memorizzati nella cache nell'archivio dati.

Valore predefinito

5

Interact | services | threadManagement | eventHandling

Le proprietà di configurazione di questa categoria definiscono le impostazioni di gestione dei thread per i servizi che raccolgono i dati per la gestione dell'evento.

corePoolSize

Descrizione

Il numero di thread da conservare nel pool, anche se sono inattivi, per la raccolta dei dati di gestione dell'evento.

Valore predefinito

1

maxPoolSize

Descrizione

Il numero massimo di thread da conservare nel pool per i servizi che raccolgono i dati di gestione dell'evento.

Valore predefinito

5

keepAliveTimeSecs

Descrizione

Quando il numero di thread è maggiore del core, questo è il tempo massimo in cui i thread inattivi in eccesso attenderanno nuove attività prima di terminare, per la raccolta dei dati di gestione dell'evento.

Valore predefinito

5

queueCapacity

Descrizione

La dimensione della coda utilizzata dal pool di thread per la raccolta dei dati di gestione dell'evento.

Valore predefinito

1000

termWaitSecs

Descrizione

Alla chiusura del server di runtime, questo è il numero di secondi in cui attendere che i thread dei servizi si completino per i servizi che raccolgono i dati di gestione dell'evento.

Valore predefinito

5

Interact | services | configurationMonitor

Le proprietà di configurazione in questa categoria consentono di abilitare o disabilitare l'integrazione con Interact Advanced Patterns senza dover riavviare l'ambiente in tempo reale Interact e definiscono l'intervallo per interrogare il valore della proprietà che abilita l'integrazione.

abilita

Descrizione

Se la proprietà è impostata su `true`, abilita il servizio che aggiorna il valore della proprietà **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**. Se il valore è `false`, è necessario riavviare l'ambiente in tempo reale Interact quando si modifica il valore della proprietà **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**.

Valore predefinito

False

Valori validi

True | False

refreshIntervalInMinutes

Descrizione

Definisce l'intervallo di tempo per interrogare il valore della proprietà **Interact | services | eventPattern | advancedPatterns enableAdvancedPatterns**.

Valore predefinito

5

Interact | cacheManagement

Questa serie di proprietà di configurazione definisce le impostazioni per la selezione e la configurazione di ognuno dei gestori cache supportati, che è possibile utilizzare per migliorare le prestazioni di Interact, ad esempio EHCache, che è integrato nella propria installazione Interact, la memorizzazione nella cache WebSphere eXtreme Scale, che è un componente aggiuntivo facoltativo o un altro sistema esterno di memorizzazione nella cache.

Utilizzare le proprietà di configurazione **Interact | cacheManagement | Cache Managers** per configurare il gestore cache che si desidera utilizzare. Utilizzare le proprietà di configurazione **Interact | cacheManagement | caches** per specificare quale gestore cache dovrebbe essere utilizzato da Interact per migliorare le prestazioni.

Interact | cacheManagement | Cache Managers

La categoria Cache Managers specifica i parametri per le soluzioni di gestione cache che si intende utilizzare con Interact.

Interact | cacheManagement | Cache Managers | EHCache

La categoria EHCache specifica i parametri per la soluzione di gestione cache EHCache, in modo da poterla personalizzare, per migliorare le prestazioni di Interact.

Interact | Cache Managers | EHCache | Parameter Data

Le proprietà di configurazione in questa categoria controllano in che modo funziona il sistema di gestione della cache EHCache, per migliorare le prestazioni di Interact.

cacheType

Descrizione

È possibile configurare i server di runtime Interact in un gruppo di server in modo che utilizzino un indirizzo multicast per la condivisione dei dati della cache. Questa configurazione è definita *cache distribuita*. Il parametro cacheType specifica se si sta utilizzando il meccanismo di memorizzazione nella cache EHCache integrato in modalità **local** (autonoma) o **distributed** (ad esempio con un gruppo dei server di runtime).

Nota:

Se si seleziona **Distributed** per cacheType, tutti i server che condividono la cache devono fare parte di un unico, medesimo, gruppo di server. È necessario anche abilitare il multicast in modo che funzioni tra tutti i membri di un gruppo di server.

Valore predefinito

Local

Valori validi

Local | Distributed

multicastIPAddress

Descrizione

Se per il parametro **cacheType** si specifica il valore "distributed," si sta configurando la cache in modo che operi tramite multicast tra tutti i membri di un gruppo dei server di runtime di Interact. Il valore multicastIPAddress corrisponde all'indirizzo IP utilizzato da tutti i server Interact del gruppo di server per l'ascolto.

L'indirizzo IP deve essere univoco per i gruppi di server.

Valore predefinito

230.0.0.1

multicastPort

Descrizione

Se si specific il valore "distributed" per il parametro **cacheType**, il parametro **multicastPort** indica la porta utilizzata da tutti i server Interact del gruppo di server per l'ascolto.

Valore predefinito

6363

overflowToDisk

Descrizione

Il gestore cache EHCache gestisce le informazioni sulla sessione utilizzando la memoria disponibile. Per ambienti in cui la sessione ha grandi dimensioni a causa di un profilo di grandi dimensioni, il numero di sessioni da supportare nella memoria è possibile che non sia sufficiente per supportare lo scenario del cliente. Per questo tipo di situazioni, EHCache dispone di una funzione facoltativa, per consentire che informazioni sulla cache, in quantità maggiore rispetto a quella che è possibile conservare in memoria, invece vengano temporaneamente scritte sul disco fisso.

Se si imposta la proprietà **overflowToDisk** su "yes," ogni JVM (Java virtual machine) potrà gestire un numero maggiore di sessioni simultanee rispetto a quello che la sola memoria avrebbe consentito.

Valore predefinito

No

Valori validi

No | Yes

diskStore

Descrizione

Quando la proprietà di configurazione **overflowToDisk** è impostata su `Yes`, questa proprietà di configurazione specifica la directory del disco che conterrà le voci della cache in eccesso per la memoria. Se questa proprietà di configurazione non esiste o il relativo valore non è valido, la directory del disco viene creata automaticamente nella directory temporanea predefinita del sistema operativo.

Valore predefinito

Nessuno

Valori validi

Una directory sulla quale l'applicazione web che ospita il runtime di Interact abbia privilegi di scrittura.

(Parameter)

Descrizione

Un modello che è possibile utilizzare per creare un parametro personalizzato da utilizzare con il gestore della cache. È possibile impostare qualsiasi nome parametro, e il valore da assegnargli.

Per creare un parametro personalizzato, fare clic su ***(Parameter)*** e compilare il nome e il valore che si desidera assegnare al parametro in questione. Quando si fa clic su **Salva modifiche**, il parametro creato viene aggiunto all'elenco nella categoria Parameter Data.

Valore predefinito

Nessuno

Interact | cacheManagement | Cache Managers | Extreme Scale

La categoria Extreme Scale specifica i parametri per fare sì che l'adattatore utilizzi la soluzione di gestione della cache WebSphere eXtreme Scale, in modo da poterla personalizzare, per migliorare le prestazioni di Interact.

ClassName

Descrizione

Il nome completo della classe che connette Interact al server WebSphere eXtreme Scale. Deve essere `com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`.

Valore predefinito

`com.unicacorp.interact.cache.extremescale.ExtremeScaleCacheManager`

ClassPath

Descrizione

L'URI del percorso del file `interact_wxs_adapter.jar`, ad esempio `file:///IBM/IMS/Interact/lib/interact_wxs_adapter.jar` o `file:///C:/IBM/IMS/Interact/lib/interact_wxs_adapter.jar`. Tuttavia, se questo file jar fosse già incluso nel percorso classi del server delle applicazioni host, il campo dovrebbe rimanere vuoto.

Valore predefinito

Vuoto

Interact | Cache Managers | Extreme Scale | Parameter Data

Le proprietà di configurazione in questa categoria controllano l'adattatore WebSphere eXtreme Scale, incluso a titolo facoltativo con l'installazione di Interact Queste impostazioni devono essere configurate per ogni server di runtime Interact che funga da client per la griglia server eXtreme Scale.

catalogPropertyFile

Descrizione

L'URI dell'ubicazione del file delle proprietà utilizzato per avviare il server di catalogo WebSphere eXtreme Scale. Se, per avviare il server di catalogo, viene utilizzato l'adattatore Extreme Scale, sarà necessario impostare questa proprietà. Altrimenti, non verrà utilizzato.

Valore predefinito

`file:///C:/depot/Interact/dev/main/extremescale/config/catalogServer.props`

containerPropertyFile

Descrizione

L'URI dell'ubicazione del file delle proprietà utilizzato per avviare le istanze contenitore WebSphere eXtreme Scale. Se, per avviare i server contenitore WebSphere eXtreme Scale, viene utilizzato il componente server incluso, sarà necessario impostare questa proprietà. Altrimenti non viene utilizzato.

Valore predefinito

file:///C:/depot/Interact/dev/main/extremescale/config/containerServer.props

deploymentPolicyFile

Descrizione

L'URI dell'ubicazione del file della politica di distribuzione utilizzato per avviare il server di catalogo WebSphere eXtreme Scale. Se, per avviare il server di catalogo WebSphere eXtreme Scale, viene utilizzato il componente server incluso, sarà necessario impostare questa proprietà. Altrimenti non viene utilizzato.

Valore predefinito

file:///C:/depot/Interact/dev/main/extremescale/config/deployment.xml

objectGridConfigFile

Descrizione

L'URI dell'ubicazione del file di configurazione della griglia di oggetti utilizzato per avviare il server di catalogo WebSphere eXtreme Scale e anche il componente near cache in esecuzione con il server di runtime Interact nella stessa JVM (Java Virtual Machine).

Valore predefinito

file:///C:/depot/Interact/dev/main/extremescale/config/objectgrid.xml

gridName

Descrizione

Il nome della griglia WebSphere eXtreme Scale che contiene tutte le cache Interact.

Valore predefinito

InteractGrid

catalogURLs

Descrizione

Un URL che contiene il nome host o l'indirizzo IP e la porta sulla quale il server di catalogo WebSphere eXtreme Scale resta in ascolto delle connessioni.

Valore predefinito

Nessuno

(Parameter)

Descrizione

Un modello che è possibile utilizzare per creare un parametro personalizzato da utilizzare con il gestore della cache. È possibile impostare qualsiasi nome parametro, e il valore da assegnargli.

Per creare un parametro personalizzato, fare clic su ***(Parameter)*** e compilare il nome e il valore che si desidera assegnare al parametro in

questione. Quando si fa clic su **Salva modifiche**, il parametro creato viene aggiunto all'elenco nella categoria Parameter Data.

Valore predefinito

Nessuno

Interact | caches

Utilizzare questa serie di proprietà di configurazione per specificare quale gestore cache supportato si desideri utilizzare per migliorare le prestazioni di Interact, ad esempio la memorizzazione nella cache Ehcache o WebSphere eXtreme Scale e per configurare proprietà della cache specifiche per il server di runtime che si sta configurando.

Questo include le cache per la memorizzazione dei dati di sessione, degli stati del pattern di evento e dei risultati della segmentazione. Assegnando i valori a queste impostazioni, è possibile specificare quale soluzione cache utilizzare per ogni tipo di cache e anche le singole impostazioni per controllare la modalità di funzionamento della cache.

Interact | cacheManagement | caches | InteractCache

La categoria InteractCache configura la memorizzazione nella cache per tutti gli oggetti della sessione, inclusi i dati del profilo, i risultati della segmentazione, i trattamenti erogati più di recente, i parametri trasmessi tramite metodi API ed altri oggetti utilizzati dal runtime Interact .

La categoria InteractCache è obbligatoria per il corretto funzionamento di Interact.

La categoria InteractCache può anche essere configurata tramite una configurazione EHCache esterna, per impostazioni non supportate in **Interact | cacheManagement | Caches**. Se si utilizza EHCache, è necessario accertarsi che InteractCache sia configurata correttamente.

CacheManagerName

Descrizione

Il nome del gestore cache che gestisce la cache Interact. Il valore qui immesso deve corrispondere ad uno dei gestori cache definiti nelle proprietà di configurazione **Interact | cacheManagement | Cache Managers**, ad esempio `EHCache` o `Extreme Scale`.

Valore predefinito

EHCache

Valori validi

Qualsiasi gestore cache definito nella proprietà di configurazione **Interact | cacheManagement | Cache Managers**.

maxEntriesInCache

Descrizione

Il numero massimo di oggetti dati di sessione da memorizzare in questa cache. Quando il numero massimo di oggetti dati di sessione è stato raggiunto e devono essere memorizzati i dati per una ulteriore sessione, viene cancellato l'oggetto utilizzato meno di recente.

Valore predefinito

100000

Valori validi

Numero intero superiore a 0.

timeoutInSecs

Descrizione

Il tempo, in secondi, trascorso da quando un oggetto dati di sessione è stato utilizzato o aggiornato, valore che viene utilizzato per determinare quando l'oggetto verrà rimosso dalla cache.

Nota: se è stato effettuato l'aggiornamento da una versione precedente alla 9.1, è necessario riconfigurare la proprietà `timeoutInSecs` perché è stata modificata.

Valore predefinito

300

Valori validi

Numero intero superiore a 0.

Interact | Caches | Interact Cache | Parameter Data

Le proprietà di configurazione in questa categoria controllano la cache Interact automaticamente utilizzata dalla propria installazione Interact. Queste impostazioni devono essere configurate singolarmente per ogni server di runtime Interact.

asyncIntervalMillis

Descrizione

Il tempo di attesa, in millisecondi, da parte del gestore cache EHCache prima di replicare eventuali modifiche ad altre istanze di runtime Interact. Se il valore non è positivo, tali modifiche verranno replicate in modo sincrono.

Questa proprietà di configurazione non viene creata per impostazione predefinita. Se si crea questa proprietà, essa verrà utilizzata solo quando EHCache è specificato come gestore cache e quando la proprietà **cacheType** per ehCache è impostata su `distributed`.

Valore predefinito

Nessuno.

(Parameter)

Descrizione

Un modello che è possibile utilizzare per creare un parametro personalizzato da utilizzare con la cache Intearct. È possibile impostare qualsiasi nome parametro, e il valore da assegnargli.

Per creare un parametro personalizzato, fare clic su ***(Parameter)*** e compilare il nome e il valore che si desidera assegnare al parametro in questione. Quando si fa clic su **Salva modifiche**, il parametro creato viene aggiunto all'elenco nella categoria Parameter Data.

Valore predefinito

Nessuno

Interact | cacheManagement | caches | PatternStateCache

La categoria PatternStateCache viene utilizzata per contenere gli stati dei pattern di evento e le regole di soppressione dell'offerta in tempo reale. Per impostazione predefinita, questa cache è configurata come cache read-through e write-through, in modo che Interact tenti di utilizzare i primi dati relativi al pattern di evento e alla soppressione dell'offerta nella cache. Se la voce richiesta non esiste nella cache, l'implementazione della cache la carica dall'origine dati, tramite la configurazione JNDI oppure utilizzando direttamente una connessione JDBC.

Per utilizzare una connessione JNDI, Interact si connette ad un provider dell'origine dati esistente, che sia stato definito tramite il server specificato, utilizzando il nome JNDI, l'URL e così via. Per una connessione JDBC, è necessario fornire una serie di impostazioni JDBC, che includono il nome classe driver JDBC, l'URL del database e le informazioni di autenticazione.

Si noti che, se si definiscono più origini JNDI e JDBC, verrà utilizzata la prima origine JNDI abilitata e, se non esiste alcuna origine JNDI abilitata, verrà utilizzata la prima origine JDBC abilitata.

La categoria PatternStateCache è obbligatoria per il corretto funzionamento di Interact.

La categoria PatternStateCache può anche essere configurata tramite una configurazione EHCache esterna, per impostazioni non supportate in **Interact | cacheManagement | Caches**. Se si utilizza EHCache, è necessario accertarsi che PatternStateCache sia configurata correttamente.

CacheManagerName

Descrizione

Il nome del gestore cache che gestisce la cache degli stati pattern di Interact. Il valore qui immesso deve corrispondere ad uno dei gestori cache definiti nelle proprietà di configurazione **Interact | cacheManagement | Cache Managers**, ad esempio `EHCache` o `Extreme Scale`.

Valore predefinito

EHCache

Valori validi

Qualsiasi gestore cache definito nella proprietà di configurazione **Interact | cacheManagement | Cache Managers**.

maxEntriesInCache

Descrizione

Il numero massimo di stati del pattern di evento da memorizzare in questa cache. Quando il numero massimo di stati pattern di evento è stato raggiunto e devono essere memorizzati i dati per un ulteriore stato di pattern di evento, viene cancellato l'oggetto utilizzato meno di recente.

Valore predefinito

100000

Valori validi

Numero intero superiore a 0.

timeoutInSecs

Descrizione

Specifica il periodo di tempo, in secondi, dopo il quale un oggetto stato del pattern di evento scade nella cache degli stati del pattern di evento. Quando tale oggetto stato è rimasto inattivo nella cache per un numero di secondi pari al valore di `timeoutInSecs`, potrebbe essere rimosso dalla cache in base alla regola dell'oggetto utilizzato meno di recente. Si noti che il valore di questa proprietà dovrebbe essere maggiore di quello definito nella proprietà `sessionTimeoutInSecs`.

Nota: se è stato effettuato l'aggiornamento da una versione precedente alla 9.1, è necessario riconfigurare la proprietà `timeoutInSecs` perché è stata modificata.

Valore predefinito

300

Valori validi

Numero intero superiore a 0.

Interact | Caches | PatternStateCache | Parameter Data:

Le proprietà di configurazione in questa categoria controllano la cache degli stati del pattern utilizzata per ospitare gli stati dei pattern di evento e le regole di soppressione dell'offerta in tempo reale.

(Parameter)

Descrizione

Un modello che è possibile utilizzare per creare un parametro personalizzato da utilizzare con la cache degli stati del pattern. È possibile impostare qualsiasi nome parametro, e il valore da assegnargli.

Per creare un parametro personalizzato, fare clic su ***(Parameter)*** e compilare il nome e il valore che si desidera assegnare al parametro in questione. Quando si fa clic su **Salva modifiche**, il parametro creato viene aggiunto all'elenco nella categoria Parameter Data.

Valore predefinito

Nessuno

Interact | cacheManagement | caches | PatternStateCache | loaderWriter:

La categoria **loaderWriter** contiene la configurazione del programma di caricamento che interagisce con depositi esterni per il richiamo e la permanenza di pattern di evento.

className

Descrizione

Il nome classe per questo programma di caricamento. Questa classe deve essere conforme con il requisito del gestore cache scelto.

Valore predefinito

```
com.unicacorp.interact.cache.ehcache.loaderwriter.
PatternStateEHCacheLoaderWriter
```

Valori validi

Un nome classe completo.

classPath

Descrizione

Il percorso per il file della classe del programma di caricamento. Se non si specifica questo valore o la voce non è valida, verrà utilizzato il percorso classi indicato per l'esecuzione di Interact.

Valore predefinito

Nessuno

Valori validi

Un percorso classi valido.

writeMode

Descrizione

Specifica la modalità in cui il programma di scrittura può rendere permanenti gli stati del pattern di evento, nuovi o aggiornati, nella cache. Opzioni valide sono:

- WRITE_THROUGH. Ogni volta che si rileva una nuova voce o che una voce esistente viene aggiornata, tale voce viene immediatamente scritta nei depositi.
- WRITE_BEHIND. Il gestore cache attende del tempo al fine di raccogliere un certo numero di modifiche e, quindi, le rende permanenti nei depositi in un batch.

Valore predefinito

WRITE_THROUGH

Valori validi

WRITE_THROUGH o WRITE_BEHIND.

batchSize

Descrizione

Il numero massimo di oggetti di stato del pattern di evento che il programma di scrittura renderà permanenti in un batch. Questa proprietà viene utilizzata solo quando la proprietà **writeMode** è impostata sul valore `WRITE_BEHIND`.

Valore predefinito

100

Valori validi

Un valore intero.

maxDelayInSecs

Descrizione

Il tempo massimo, in secondi, che il gestore cache attende prima che un oggetto stato del pattern di evento venga reso permanente. Questa proprietà viene utilizzata solo quando la proprietà **writeMode** è impostata sul valore `WRITE_BEHIND`.

Valore predefinito

5

Valori validi

Un valore intero.

Interact | Caches | PatternStateCache | loaderWriter | Parameter Data:

Le proprietà di configurazione in questa categoria controllano il programma di caricamento della cache degli stati del pattern.

(Parameter)

Descrizione

Un modello che è possibile utilizzare per creare un parametro personalizzato da utilizzare con il programma di caricamento della cache degli stati del pattern. È possibile impostare qualsiasi nome parametro, e il valore da assegnargli.

Per creare un parametro personalizzato, fare clic su ***(Parameter)*** e compilare il nome e il valore che si desidera assegnare al parametro in questione. Quando si fa clic su **Salva modifiche**, il parametro creato viene aggiunto all'elenco nella categoria Parameter Data.

Valore predefinito

Nessuno

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jndiSettings:

La categoria **jndiSettings** contiene la configurazione per l'origine dati JNDI che il programma di caricamento utilizzerà per comunicare con il database di supporto. Per creare una nuova serie di impostazioni JNDI, espandere la categoria **jdniSettings** e fare clic sulla proprietà ***(jndiSetting)***.

(jndiSettings)

Nota: quando si utilizza WebSphere Application Server, loaderWriter non viene connesso con **jndiSettings**.

Descrizione

Quando si fa clic su questa categoria, viene visualizzato un modulo. Per definire un'origine dati JNDI, compilare i seguenti valori:

- **Nuovo nome categoria** è il nome che si vuole utilizzare per identificare questa connessione JNDI.
- **enabled** consente all'utente di indicare se desidera che questa connessione JNDI sia disponibile o meno per l'uso. Impostare questa voce su `True` per nuove connessioni.
- **jdniName** è il nome JNDI che è già stato definito nell'origine dati quando è stata configurata.

- **providerUrl** è l'URL per trovare questa origine dati JNDI. Se non si compila questo campo, verrà utilizzato l'URL dell'applicazione Web che ospita il runtime Interact.
- **Factory di contesto iniziale** è il nome classe completo della classe Factory di contesto iniziale per la connessione al provider JNDI. Se per **providerUrl** è stata utilizzata l'applicazione Web che ospita il runtime Interact, lasciare vuoto questo campo.

Valore predefinito

Nessuno.

Interact | cacheManagement | caches | PatternStateCache | loaderWriter | jdbcSettings:

La categoria **jdbcSettings** contiene la configurazione per le connessioni JDBC che il programma di caricamento utilizzerà per comunicare con il database di supporto. Per creare una nuova serie di impostazioni JDBC, espandere la categoria **jdbcSettings** e fare clic sulla proprietà ***(jdbcSetting)***.

(jdbcSettings)

Descrizione

Quando si fa clic su questa categoria, viene visualizzato un modulo. Per definire un'origine dati JDBC, compilare i seguenti valori:

- **Nuovo nome categoria** è il nome che si vuole utilizzare per identificare questa connessione JDBC.
- **enabled** consente all'utente di indicare se desidera che questa connessione JDBC sia disponibile o meno per l'uso. Impostare questa voce su `True` per nuove connessioni.
- **driverClassName** è il nome classe completo del driver JDBC. Questa classe deve esistere nel percorso classi configurato per l'avvio del server cache host.
- **databaseUrl** è l'URL per trovare questa origine dati JDBC.
- **asmUser** è il nome dell'utente IBM Marketing Software che è stato configurato con le credenziali per la connessione al database in questa connessione JDBC.
- **asmDataSource** è il nome dell'origine dati IBM Marketing Software che è stata configurata con le credenziali per la connessione al database in questa connessione JDBC.
- **maxConnection** è il numero massimo di connessioni simultanee consentite per il database in questa connessione JDBC.

Valore predefinito

Nessuno.

Interact | triggeredMessage

Le proprietà di configurazione di questa categoria definiscono le impostazioni per la consegna tramite canale dell'oferta e i messaggi attivati.

backendProcessIntervalMin

Descrizione

Questa proprietà definisce il periodo di tempo in minuti durante il quale il thread di backend carica ed elabora le consegne di offerte ritardate. Questo valore deve essere un numero intero. Se il valore è zero o negativo, il processo di backend è disabilitato.

Valori validi

Un numero intero positivo

autoLogContactAfterDelivery

Descrizione

Se questa proprietà è impostata su true, un evento contatto viene inviato automaticamente non appena viene distribuita questa offerta o questa offerta viene accodata per la consegna ritardata. Se questa proprietà è impostata su false, non viene automaticamente inviato un evento contatto per le offerte in uscita. Questo è il funzionamento predefinito.

Nota:

- Se si desidera catturare altri attributi nella cronologia dei contatti quando viene attivato il messaggio in uscita, è possibile aggiungere gli altri attributi personalizzati come colonne nella cronologia dei contatti. Mentre si pubblica un evento, che attiverà il messaggio in uscita, è possibile passare i valori per l'attributo nel metodo `postEvent` come parametri del valore nome.
- Per parametrizzare un'offerta su un canale in uscita, è possibile assegnare offerte nella strategia associata, distribuire il canale, personalizzare l'offerta e nel messaggio attivato scegliere **Seleziona automaticamente la migliore offerta successiva**.

Valori validi

True | False

waitForFlowchart

Descrizione

Questa proprietà determina se il diagramma di flusso deve attendere il completamento della segmentazione attualmente in esecuzione, e il comportamento se l'attesa scade.

DoNotWait: l'elaborazione di un messaggio attivato viene avviata indipendentemente dal fatto che la segmentazione è attualmente in esecuzione o meno. Tuttavia, se i segmenti vengono utilizzati nella regola di idoneità e/o è selezionato NextBestOffer come metodo di selezione dell'offerta, l'esecuzione TM attende comunque.

OptionalWait: l'elaborazione di un messaggio attivato attende fino a quando la segmentazione attualmente in esecuzione termina o scade. Se l'attesa scade, viene registrato un avviso e l'elaborazione di questo messaggio attivato continua. Questo è il valore predefinito.

MandatoryWait: l'elaborazione di un messaggio attivato attende fino a quando la segmentazione attualmente in esecuzione termina o scade. Se l'attesa scade, viene registrato un errore e l'elaborazione di questo messaggio attivato si interrompe.

Valori validi

DoNotWait | OptionalWait | MandatoryWait

Interact | triggeredMessage | offerSelection

Le proprietà di configurazione di questa categoria definiscono le impostazioni per la selezione offerta nei messaggi attivati.

maxCandidateOffers

Descrizione

Questa proprietà definisce il numero massimo di offerte idonee che il motore restituisce per ottenere la migliore offerta per la consegna. C'è la possibilità che nessuna di queste offerte idonee restituite possa essere inviata in base al canale selezionato. Più offerte candidate ci sono, meno si verifica questo caso. Tuttavia, più offerte candidate possono aumentare il tempo di elaborazione.

Valori validi

Un numero intero positivo

defaultCellCode

Descrizione

Se l'offerta consegnata è il risultato della valutazione di una regola strategica o di un record diretto dalla tabella, esiste una cella obiettivo associata ad essa e le informazioni di questa cella vengono utilizzate in tutta la registrazione correlata. Tuttavia, se viene utilizzato un elenco di offerte specifiche come input per la selezione dell'offerta, non è disponibile alcuna cella obiettivo. In questo caso, viene utilizzato il valore di questa impostazione di configurazione. È necessario assicurarsi che questa cella obiettivo e la relativa campagna siano comprese nella distribuzione. Il metodo più semplice per raggiungere tale obiettivo è aggiungere la cella in una strategia distribuita.

Interact | triggeredMessage | dispatchers

Le proprietà di configurazione di questa categoria definiscono le impostazioni per tutti i dispatcher nei messaggi attivati.

dispatchingThreads

Descrizione

Questa proprietà definisce il numero di thread che il motore utilizza per richiamare in modo asincrono i dispatcher. Se il valore è 0 o un numero negativo, il richiamo dei dispatcher è sincrono. Il valore predefinito è 0.

Valori validi

Un numero intero

Interact | triggeredMessage | dispatchers | <dispatcherName>

Le proprietà di configurazione di questa categoria definiscono le impostazioni per un dispatcher specifico nei messaggi attivati.

nome categoria

Descrizione

Questa proprietà definisce il nome di questo dispatcher. Il nome deve essere univoco tra tutti i dispatcher.

type

Descrizione

Questa proprietà definisce il tipo di disptacher.

Valori validi

InMemoryQueue | JMSQueue | Custom

Nota: Se si utilizza JMSQueue o Custom, per integrare Interact con IBM MQ, il runtime di Interact deve essere sul server delle applicazioni con JDK 1.7. Per WebSphere e WebLogic, si consiglia di utilizzare l'ultima versione del fix pack JDK fornito.

JMSQueue supporta solo WebLogic. Non è possibile utilizzare JMSQueue se si utilizza WebSphere Application Server.

className

Descrizione

Questa proprietà definisce il nome classe completo di questa implementazione del dispatcher. Se il tipo è InMemoryQueue il valore deve essere vuoto. Se il tipo è custom, questa impostazione deve avere il valore `com.unicacorp.interact.eventhandler.triggeredmessage.dispatchers.IBMMQDispatcher`.

classPath

Descrizione

Questa proprietà definisce l'URL per il file JAR che include l'implementazione di questo dispatcher.

Se il tipo è custom, questa impostazione ha il valore `file://<Interact_HOME>/lib/interact_ibmmqdispatcher.jar;file://<Interact_HOME>/lib/com.ibm.mq.allclient.jar;file://<Interact_HOME>/lib/jms.jar`

Interact | triggeredMessage | dispatchers | <dispatcherName> | Parameter Data

Le proprietà di configurazione di questa categoria definiscono i parametri per un dispatcher specifico nei messaggi attivati.

È possibile scegliere tra tre tipi di dispatcher. InMemoryQueue è il dispatcher interno per Interact. Custom viene utilizzato per IBM MQ. JMSQueue viene utilizzato per la connessione a un provider JMS tramite JNDI.

nome categoria

Descrizione

Questa proprietà definisce il nome di questo parametro. Il nome deve essere univoco tra tutti i parametri per quel dispatcher.

value

Descrizione

Questa proprietà definisce i parametri, nel formato di coppie nome-valore, necessari per questo dispatcher.

Nota: Tutti i parametri per i messaggi attivati sono sensibili al maiuscolo/minuscolo e devono essere immessi come mostrato qui.

Se il tipo è InMemoryQueue, il seguente parametro è supportato.

- queueCapacity: facoltativo. Il numero massimo di offerte che possono essere in attesa nella coda per essere distribuite. Quando specificata, questa proprietà deve essere un numero intero positivo. Se non specificata o non valida, viene utilizzato il valore predefinito (1000).

Se il tipo è Custom, i seguenti parametri sono supportati.

- providerUrl: <hostname>:port (sensibile al maiuscolo/minuscolo)
- queueManager: il nome del gestore code che è stato creato sul server IBM MQ.
- messageQueueName: il nome della coda messaggi che è stata creata sul server IBM MQ.
- enableConsumer: questa proprietà deve essere impostata su true.
- asmUserforMQAuth: il nome utente per l'accesso al server. È richiesto quando il server applica l'autenticazione. Altrimenti, non deve essere specificato.
- authDS: la password associata al nome utente per l'accesso al server. È richiesto quando il server applica l'autenticazione. Altrimenti, non deve essere specificato.

Se il tipo è JMSQueue, il seguente parametro è supportato.

- providerUrl: L'URL del provider JNDI (sensibile al maiuscolo/minuscolo).
- connectionFactoryJNDI: il nome JNDI del factory di connessioni JMS.
- messageQueueJNDI: il nome JNDI della coda JMS a cui vengono inviati e da cui vengono richiamati i messaggi attivati.
- enableConsumer: se un utente di quei messaggi attivati deve essere avviato in Interact. Questa proprietà deve essere impostata su true. Se non specificato, viene utilizzato il valore predefinito (false).
- initialContextFactory: il nome completo della classe factory di contesto iniziale JNDI. Se si utilizza WebLogic, il valore di questo parametro deve essere `weblogic.jndi.WLInitialContextFactory`.

Interact | triggeredMessage | gateways | <gatewayName>

Le proprietà di configurazione di questa categoria definiscono le impostazioni per un gateway specifico nei messaggi attivati.

Interact non supporta più istanze dello stesso gateway. Tutti i file di configurazione gateway devono essere accessibili da ogni nodo del runtime Interact. Nel caso di una configurazione distribuita, assicurarsi che i file gateway si trovino in un'ubicazione condivisa.

nome categoria

Descrizione

Questa proprietà definisce il nome di questo gateway. Deve essere univoco tra tutti i gateway.

className

Descrizione

Questa proprietà definisce il nome classe completo di questa implementazione del gateway.

classPath

Descrizione

Questa proprietà definisce l'URI del file JAR che include l'implementazione di questo gateway. Se lasciato vuoto, viene utilizzato il percorso classe dell'applicazione Interact ospitante.

Ad esempio, in un sistema windows, se il file JAR del gateway è disponibile nella directory `C:\IBM\EMM\EmailGateway\IBM_Interact_OMO_OutboundGateway_Silverpop_1.0\lib\OMO_OutboundGateway_Silverpop.jar`, il percorso classi sarà `file:///C:/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`. In un sistema unixse il file JAR del gateway è disponibile nella directory `/opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`, il percorso classi sarà `file:///opt/IBM/EMM/EmailGateway/IBM_Interact_OMO_OutboundGateway_Silverpop_1.0/lib/OMO_OutboundGateway_Silverpop.jar`.

Interact | triggeredMessage | gateways | <gatewayName> | Parameter Data

Le proprietà di configurazione di questa categoria definiscono i parametri per un gateway specifico nei messaggi attivati.

nome categoria

Descrizione

Questa proprietà definisce il nome di questo parametro. Il nome deve essere univoco tra tutti i parametri per quel gateway.

value

Descrizione

Questa proprietà definisce i parametri, nel formato di coppie nome-valore, necessari per questo gateway. Per tutti i gateway, sono supportati i seguenti parametri.

Nota: Tutti i parametri per i messaggi attivati sono sensibili al maiuscolo/minuscolo e devono essere immessi come mostrato qui.

- validationTimeoutMillis: il periodo di tempo in millisecondi dopo il quale la convalida di un'offerta attraverso questo gateway scade. Il valore predefinito è 500.
- deliveryTimeoutMillis: il periodo di tempo in millisecondi dopo il quale la consegna di un'offerta tramite questo gateway scade. Il valore predefinito è 1000.

Interact | triggeredMessage | channels

Le proprietà di configurazione di questa categoria definiscono le impostazioni per tutti i canali nei messaggi attivati.

type

Descrizione

Questa proprietà definisce il nodo root per le impostazioni relative ad un gateway specifico. Default corrisponde al selettore canale incorporato, che si basa sull'elenco di canali definiti nella IU dei messaggi attivati. Se viene selezionato Default, i valori className e classPath devono essere lasciati vuoti. Custom corrisponde all'implementazione di IChannelSelector del cliente.

Valori validi

Default | Custom

className

Descrizione

Questa proprietà definisce il nome classe completo dell'implementazione da parte del cliente del selettore canale. Questa impostazione è obbligatoria se il tipo è Custom.

classPath

Descrizione

Questa proprietà definisce l'URL per il file JAR che include l'implementazione da parte del cliente del selettore canale. Se lasciato vuoto, viene utilizzato il percorso classe dell'applicazione Interact ospitante.

Interact | triggeredMessage | channels | Parameter Data

Le proprietà di configurazione di questa categoria definiscono i parametri per un canale specifico nei messaggi attivati.

nome categoria

Descrizione

Questa proprietà definisce il nome di questo parametro. Il nome deve essere univoco tra tutti i parametri per quel canale.

value

Descrizione

Questa proprietà definisce i parametri, nel formato di coppie nome-valore, necessari per il selettore canale.

Se si utilizza **Canale preferito cliente** per il canale, è necessario creare

Interact | triggeredMessage | channels | <channelName>

Le proprietà di configurazione di questa categoria definiscono le impostazioni per un canale specifico nei messaggi attivati.

nome categoria

Descrizione

Questa proprietà definisce il nome del canale tramite il quale vengono inviate le offerte. Deve corrispondere a quelli definiti nella fase di progettazione in **Campaign | partitions | <partition[N]> | Interact | outboundChannels**.

Interact | triggeredMessage | channels | <channelName> | <handlerName>

Le proprietà di configurazione di questa categoria definiscono le impostazioni per un gestore specifico nei messaggi attivati.

nome categoria

Descrizione

Questa proprietà definisce il nome del gestore che il canale utilizzerà per inviare le offerte.

dispatcher

Descrizione

Questa proprietà definisce il nome del dispatcher attraverso il quale questo gestore invia le offerte al gateway. Deve essere uno di quelli definiti in **interact | triggeredMessage | dispatchers**.

gateway

Descrizione

Questa proprietà definisce il nome del gateway a cui questo gestore invia le offerte alla fine. Deve essere uno di quelli definiti in **interact | triggeredMessage | gateways**.

mode

Descrizione

Questa proprietà definisce la modalità di utilizzo di questo gestore. Se è selezionato Failover, questo gestore viene utilizzato solo quando tutti i gestori con priorità più elevate definiti in questo canale non sono riusciti a inviare le offerte. Se è selezionato Componente aggiuntivo, questo gestore viene utilizzato indipendentemente dal fatto che altri gestori hanno correttamente inviato offerte.

priority

Descrizione

Questa proprietà definisce la priorità di questo gestore. Il primo motore tenta di utilizzare il gestore con la priorità più alta per l'invio delle offerte.

Valori validi

Qualsiasi numero intero

Impostazione predefinita

100

Interact | activityOrchestrator

La categoria orchestrator attività specifica i ricevitori e i gateway per l'attività del gateway in entrata Interact.

Utilizzare le proprietà di configurazione **Interact | activityOrchestrator | receivers** per configurare i ricevitori di Interact. Utilizzare le proprietà di configurazione **Interact | activityOrchestrator | gateways** per configurare i gateway da utilizzare in Interact.

Interact | activityOrchestrator | receivers

La categoria ricevitori dell'orchestrator attività specifica i ricevitori evento per l'attività del gateway in entrata Interact.

Nome categoria

Descrizione

Il nome del ricevitore.

Tipo

Descrizione

Il tipo di ricevitore. È possibile scegliere tra `IBM MQ` e `Personalizzato`. `Personalizzato` richiede che si utilizzi un'implementazione di `iReceiver`.

Abilitato

Descrizione

Selezionare `True` per abilitare il ricevitore o `false` per disabilitarlo.

className

Descrizione

Questa proprietà definisce il nome classe completo di questa implementazione del ricevitore. È utilizzata solo quando il tipo è `Personalizzato`.

classPath

Descrizione

Questa proprietà definisce l'URI per il file JAR che include l'implementazione di questo ricevitore. Se lasciato vuoto, viene utilizzato il percorso classe dell'applicazione Interact ospitante. È utilizzata solo quando il tipo è `Personalizzato`.

Interact | activityOrchestrator | receivers | Parameter Data

È possibile aggiungere parametri ricevitore, come queueManager e messageQueueName, per definire la coda del ricevitore.

Interact | activityOrchestrator | gateways

La categoria gateway dell'orchestrator attività specifica i gateway per l'attività del gateway in entrata Interact.

Nome categoria

Descrizione

Il nome del gateway.

className

Descrizione

Questa proprietà definisce il nome classe completo di questa implementazione del gateway.

classPath

Descrizione

Questa proprietà definisce l'URI per il file JAR che include l'implementazione di questo gateway. Se lasciato vuoto, viene utilizzato il percorso classe dell'applicazione Interact ospitante. È utilizzata solo quando il tipo è `Personalizzato`.

Interact | activityOrchestrator | gateways | Parameter Data

È possibile aggiungere i parametri gateway per i file di configurazione gateway, ad esempio `OMO-conf_inbound_UBX_interactEventNameMapping` e `OMO-conf_inbound_UBX_interactEventPayloadMapping`.

Interact | ETL | patternStateETL

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il processo ETL.

Nuovo nome categoria

Descrizione

Fornire un nome che identifichi in modo univoco questa configurazione. Si noti che è necessario fornire il nome esatto, quando si esegue il processo ETL in modalità autonoma. Per comodità, quando si specifica questo nome sulla riga di comando, sarebbe opportuno evitare un nome contenente spazi o punteggiatura, ad esempio `ETLProfile1`.

runOnceADay

Descrizione

Stabilisce se il processo ETL in modalità autonoma, in questa configurazione, dovrebbe essere eseguito una volta al giorno. Risposte valide sono **Yes** o **No**. Se si risponde **No** per questa voce, il valore **processSleepIntervalInMinutes** determina la pianificazione dell'esecuzione per il processo.

preferredStartTime

Descrizione

L'ora preferita in cui il processo ETL in modalità autonoma deve iniziare. Specificare l'ora nel formato HH:MM:SS AM/PM, come in `01:00:00 AM`.

preferredEndTime

Descrizione

L'ora preferita in cui il processo ETL in modalità autonoma deve arrestarsi. Specificare l'ora nel formato HH:MM:SS AM/PM, come in `08:00:00 AM`.

processSleepIntervalInMinutes

Descrizione

Se il processo ETL in modalità autonoma non è stato configurato per l'esecuzione una volta ala giorno (come specificato nella proprietà **runOnceADay**), con questa proprietà si specifica l'intervallo tra le esecuzioni del processo ETL. Ad esempio, se si specifica `15` per questa

proprietà, il processo ETL in modalità autonoma attenderà 15 minuti dopo aver arrestato l'esecuzione, prima di avviarsi nuovamente.

maxJDBCInsertBatchSize

Descrizione

Il numero massimo di record di un batch JDBC, prima di eseguire il commit della query. Per impostazione predefinita, questa proprietà è impostata su 5000. Si noti che questo non è il numero massimo di record che l'ETL elabora in un'iterazione. Durante ogni iterazione, l'ETL elabora tutti i record disponibili dalla tabella UACI_EVENTPATTERNSTATE. Tuttavia, tutti quei record vengono suddivisi in blocchi in base al valore **maxJDBCInsertSize**.

maxJDBCFetchBatchSize

Descrizione

Il numero massimo di record di un batch JDBC da estrarre dal database di staging.

Potrebbe essere necessario aumentare questo valore per ottimizzare le prestazioni dell'ETL.

communicationPort

Descrizione

La porta di rete su cui il processo ETL in modalità autonoma resta in ascolto di una richiesta di arresto. In circostanze normali, non dovrebbe esistere alcun motivo per cambiare questo valore dal valore predefinito.

queueLength

Descrizione

Un valore utilizzato per l'ottimizzazione delle prestazioni. Le raccolte di dati sullo stato del pattern vengono estratte e trasformate in oggetti, i quali vengono aggiunti ad una coda per essere elaborati e scritti nel database. Questa proprietà controlla la dimensione della coda.

completionNotificationScript

Descrizione

Specifica il percorso assoluto per uno script da eseguire una volta completato il processo ETL. Se si specifica uno script, vengono passati tre argomenti allo script di notifica del completamento: ora di inizio, ora di fine e numero totale di record del pattern di evento elaborati. L'ora di inizio e l'ora di fine sono valori numerici che rappresentano il numero di millisecondi trascorsi dal 1970.

Interact | ETL | patternStateETL | <patternStateETLName> | RuntimeDS

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il DS di runtime ETL.

type

Descrizione

Un elenco dei tipi di database supportati per l'origine dati che si sta definendo.

dsname

Descrizione

Il nome JNDI dell'origine dati. Questo nome deve essere utilizzato anche nella configurazione dell'origine dati dell'utente, per garantire che l'utente abbia accesso alle origini dati di runtime e di destinazione.

driver

Descrizione

Il nome del driver JDBC da utilizzare, ad esempio, uno dei seguenti:

Oracle: `oracle.jdbc.OracleDriver`

Microsoft SQL Server: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

IBM DB2: `com.ibm.db2.jcc.DB2Driver`

serverURL

Descrizione

L'URL dell'origine dati, ad esempio, uno dei seguenti:

Oracle: `jdbc:oracle:thin:@ <your_db_host>:<your_db_port>:<your_db_service_name>`

Microsoft SQL Server: `jdbc:sqlserver:// <your_db_host>:<your_db_port> ;databaseName= <your_db_name>`

IBM DB2: `jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>`

connectionpoolSize

Descrizione

Un valore che indica la dimensione del pool di connessioni, fornito per l'ottimizzazione delle prestazioni. I dati sullo stato del pattern vengono letti e trasformati contemporaneamente, in base alle connessioni al database disponibili. L'aumento della dimensione del pool di connessioni consente un numero maggiore di connessioni al database simultanee, vincolate dalle limitazioni di memoria e dalle funzioni di lettura/scrittura del database. Ad esempio, se questo valore è impostato su 4, sarà possibile l'esecuzione simultanea di quattro job. Se si dispone di una gran quantità di dati, potrebbe risultare necessario l'aumento di questo valore fino a 10 o 20, fintanto che sono disponibili memoria e prestazioni del database sufficienti.

schema

Descrizione

Il nome dello schema del database a cui questa configurazione si sta connettendo.

connectionRetryPeriod

Descrizione

La proprietà ConnectionRetryPeriod specifica il periodo di tempo, in secondi, durante il quale Interact ripete automaticamente la richiesta di connessione al database in caso di esito negativo. Interact tenta automaticamente di riconnettersi al database per questo periodo di tempo, prima di riportare un errore o malfunzionamento del database. Se il valore è impostato su 0, Interact ripete il tentativo per un tempo indefinito; se il valore è impostato su -1, non verrà effettuato alcun tentativo.

connectionRetryDelay

Descrizione

La proprietà ConnectionRetryDelay specifica il periodo di tempo, in secondi, durante il quale Interact rimane in attesa, prima di riprovare a connettersi al database dopo un errore. Se il valore è impostato su -1, non verranno effettuati tentativi.

Interact | ETL | patternStateETL | <patternStateETLName> | TargetDS

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il DS di destinazione ETL.

type

Descrizione

Un elenco dei tipi di database supportati per l'origine dati che si sta definendo.

dsname

Descrizione

Il nome JNDI dell'origine dati. Questo nome deve essere utilizzato anche nella configurazione dell'origine dati dell'utente, per garantire che l'utente abbia accesso alle origini dati di runtime e di destinazione.

driver

Descrizione

Il nome del driver JDBC da utilizzare, ad esempio, uno dei seguenti:

Oracle: oracle.jdbc.OracleDriver

Microsoft SQL Server: com.microsoft.sqlserver.jdbc.SQLServerDriver

IBM DB2: com.ibm.db2.jcc.DB2Driver

serverURL

Descrizione

L'URL dell'origine dati, ad esempio, uno dei seguenti:

Oracle: jdbc:oracle:thin:@ <your_db_host>:<your_db_port>:<your_db_service_name>

Microsoft SQL Server: jdbc:sqlserver:// <your_db_host>:<your_db_port> ;databaseName= <your_db_name>

IBM DB2: jdbc:db2:// <your_db_host>:<your_db_port>/<your_db_name>

connectionpoolSize

Descrizione

Un valore che indica la dimensione del pool di connessioni, fornito per l'ottimizzazione delle prestazioni. I dati sullo stato del pattern vengono letti e trasformati contemporaneamente, in base alle connessioni al database disponibili. L'aumento della dimensione del pool di connessioni consente un numero maggiore di connessioni al database simultanee, vincolate dalle limitazioni di memoria e dalle funzioni di lettura/scrittura del database. Ad esempio, se questo valore è impostato su 4, sarà possibile l'esecuzione simultanea di quattro job. Se si dispone di una gran quantità di dati, potrebbe risultare necessario l'aumento di questo valore fino a 10 o 20, fintanto che sono disponibili memoria e prestazioni del database sufficienti.

schema

Descrizione

Il nome dello schema del database a cui questa configurazione si sta connettendo.

connectionRetryPeriod

Descrizione

La proprietà `ConnectionRetryPeriod` specifica il periodo di tempo, in secondi, durante il quale Interact ripete automaticamente la richiesta di connessione al database in caso di esito negativo. Interact tenta automaticamente di riconnettersi al database per questo periodo di tempo, prima di riportare un errore o malfunzionamento del database. Se il valore è impostato su 0, Interact ripete il tentativo per un tempo indefinito; se il valore è impostato su -1, non verrà effettuato alcun tentativo.

connectionRetryDelay

Descrizione

La proprietà `ConnectionRetryDelay` specifica il periodo di tempo, in secondi, durante il quale Interact rimane in attesa, prima di riprovare a connettersi al database dopo un errore. Se il valore è impostato su -1, non verranno effettuati tentativi.

Interact | ETL | patternStateETL | <patternStateETLName> | Report

Le proprietà di configurazione di questa categoria definiscono le impostazioni per il processo di aggregazione di report ETL.

abilita

Descrizione

Abilitare o disabilitare l'integrazione report con ETL. Questa proprietà è impostata su disable per impostazione predefinita.

Se impostata su `disable`, questa proprietà disabilita gli aggiornamenti nella tabella UARI_DELTA_PATTERNS. Non disabilita completamente la funzione di reporting .

Nota: per disabilitare l'integrazione del report con ETL, è necessario modificare il valore del trigger TR_AGGREGATE_DELTA_PATTERNS in disable nella tabella di staging UACI_ETLPATTERNSTATERUN.

retryAttemptsIfAggregationRunning

Descrizione

Il numero di volte in cui ETL tenta di verificare se l'aggregazione di report si è completata, nel caso sia impostato l'indicatore di blocco. Questa proprietà è impostata su 3 per impostazione predefinita.

sleepBeforeRetryDurationInMinutes

Descrizione

Il periodo di sospensione, in minuti, tra tentativi consecutivi. Questa proprietà è impostata su 5 minuti per impostazione predefinita.

aggregationRunningCheckSql

Descrizione

Questa proprietà consente di definire un'istruzione SQL personalizzata, che sarà possibile eseguire per verificare se è impostato l'indicatore di blocco dell'aggregazione di report. Per impostazione predefinita, per questa proprietà non è specificato alcun valore.

Quando questa proprietà non è impostata, l'ETL esegue la seguente istruzione SQL per richiamare l'indicatore di blocco.

```
select count(1) AS ACTIVERUNS from uari_pattern_lock where islock='Y'
=> If ACTIVERUNS is > 0, lock is set
```

aggregationRunningCheck

Descrizione

Abilitare o disabilitare la verifica che l'esecuzione dell'aggregazione di report sia avvenuta prima dell'esecuzione dell'ETL. Questa proprietà è impostata su enable per impostazione predefinita.

Capitolo 14. Proprietà di configurazione dell'ambiente di progettazione di Interact

Questa sezione descrive tutte le proprietà di configurazione per l'ambiente di progettazione di Interact.

Campaign | partitions | partition[n] | reports

La proprietà **Campaign | partitions | partition[n] | reports** definisce i tipi differenti di cartelle per i report.

offerAnalysisTabCachedFolder

Descrizione

La proprietà `offerAnalysisTabCachedFolder` specifica l'ubicazione della cartella che contiene la specifica per i report dell'offerta esplosi (espansi), elencati nella scheda Analisi, quando la si raggiunge facendo clic sul link Analisi nel riquadro di navigazione. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

Descrizione

La proprietà `segmentAnalysisTabOnDemandFolder` specifica l'ubicazione della cartella che contiene i report del segmento elencati nella scheda Analisi di un segmento. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

Descrizione

La proprietà `offerAnalysisTabOnDemandFolder` specifica l'ubicazione della cartella che contiene i report dell'offerta elencati nella scheda Analisi di un'offerta. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

segmentAnalysisTabCachedFolder

Descrizione

La proprietà `segmentAnalysisTabCachedFolder` specifica l'ubicazione della cartella che contiene la specifica per i report del segmento esplosi (espansi),

elencati nella scheda Analisi, quando la si raggiunge facendo clic sul link Analisi nel riquadro di navigazione. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']
```

analysisSectionFolder

Descrizione

La proprietà `analysisSectionFolder` specifica l'ubicazione della cartella root in cui vengono archiviate le specifiche dei report. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign']
```

campaignAnalysisTabOnDemandFolder

Descrizione

La proprietà `campaignAnalysisTabOnDemandFolder` specifica l'ubicazione della cartella che contiene i report sulla campagna elencati nella scheda Analisi di una campagna. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']
```

campaignAnalysisTabCachedFolder

Descrizione

La proprietà `campaignAnalysisTabCachedFolder` specifica l'ubicazione della cartella che contiene la specifica dei report sulla campagna esplosi (espansi), elencati nella scheda Analisi, quando la si raggiunge facendo clic sul link Analisi nel riquadro di navigazione. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']/folder[@name='cached']
```

campaignAnalysisTabEmessageOnDemandFolder

Descrizione

La proprietà `campaignAnalysisTabEmessageOnDemandFolder` specifica l'ubicazione della cartella che contiene i report eMessage elencati nella scheda Analisi di una campagna. Il percorso viene specificato utilizzando la notazione XPath.

Valore predefinito

```
/content/folder[@name='Affinium Campaign']/folder[@name='eMessage
Reports']
```

campaignAnalysisTabInteractOnDemandFolder

Descrizione

La stringa della cartella del server di report per i report Interact.

Valore predefinito

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

Disponibilità

Questa proprietà è disponibile solo se si installa Interact.

interactiveChannelAnalysisTabOnDemandFolder

Descrizione

La stringa della cartella del server di report per i report della scheda dell'analisi Canale interattivo.

Valore predefinito

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

Disponibilità

Questa proprietà è disponibile solo se si installa Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking

Queste proprietà di configurazione definiscono le impostazioni per il modulo della cronologia dei contatti e delle risposte di Interact.

isEnabled

Descrizione

Se impostata su `yes`, abilita il modulo della cronologia dei contatti e delle risposte di Interact che copia la cronologia dei contatti e delle risposte di Interact dalle tabelle di staging durante il runtime di Interact alle tabelle della cronologia dei contatti e delle risposte di Campaign. È necessario che anche la proprietà `interactInstalled` sia impostata su `yes`.

Valore predefinito

no

Valori validi

yes | no

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

runOnceADay

Descrizione

Specifica se eseguire l'ETL della cronologia dei contatti e delle risposte una volta al giorno. Se si imposta questa proprietà su `Yes`, l'ETL viene eseguito

durante l'intervallo pianificato, specificato mediante i valori `preferredStartTime` e `preferredEndTime`.

Se l'esecuzione dell'ETL impiega più di 24 ore e, quindi, non rispetta l'ora di inizio del giorno successivo, quel giorno verrà ignorato e l'esecuzione avrà luogo all'ora pianificata del giorno successivo. Ad esempio, se l'esecuzione dell'ETL è configurata tra l'1.00 e le 3.00 ed il processo parte il lunedì all'1.00 e viene completato alle 2.00 di martedì, l'esecuzione successiva, inizialmente pianificata per l'1.00 di martedì, verrà ignorata e la prossima esecuzione dell'ETL verrà avviata all'1.00 di mercoledì.

La pianificazione dell'ETL non tiene conto delle modifiche dell'ora legale. Ad esempio, se l'esecuzione dell'ETL è pianificata tra l'1.00 e le 3.00, l'esecuzione si potrebbe avviare alle 24.00 o alle 2.00, quando si applica il cambiamento dell'ora legale.

Valore predefinito

No

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

processSleepIntervalInMinutes

Descrizione

Il numero di minuti di attesa del modulo della cronologia dei contatti e delle risposte di Interact tra la copia dei dati dalle tabelle di staging del runtime di Interact alle tabelle della cronologia dei contatti e delle risposte di Campaign.

Valore predefinito

60

Valori validi

Qualsiasi numero intero superiore a zero.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

preferredStartTime

Descrizione

L'ora preferita per avviare il processo ETL quotidiano. Questa proprietà, quando viene utilizzata insieme alla proprietà preferredEndTime, configura l'intervallo di tempo preferito durante il quale si desidera venga eseguito l'ETL. L'ETL si avvierà durante l'intervallo di tempo specificato ed elaborerà, al massimo, il numero di record specificati tramite il valore `maxJDBCFetchBatchSize`. Il formato è HH:mm:ss AM o PM, quando si utilizza un orologio a 12 ore.

Valore predefinito

12:00:00 AM

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

preferredEndTime

Descrizione

L'ora preferita per completare il processo ETL quotidiano. Questa proprietà, quando viene utilizzata insieme alla proprietà preferredStartTime, configura l'intervallo di tempo preferito durante il quale si desidera venga eseguito l'ETL. L'ETL si avvierà durante l'intervallo di tempo specificato ed elaborerà, al massimo, il numero di record specificati tramite il valore `maxJDBCFetchBatchSize`. Il formato è HH:mm:ss AM o PM, quando si utilizza un orologio a 12 ore.

Valore predefinito

2:00:00 AM

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

purgeOrphanResponseThresholdInMinutes

Descrizione

Il numero di minuti di attesa del modulo della cronologia dei contatti e delle risposte di Interact, prima di cancellare le risposte che non hanno un contatto corrispondente. In questo modo si evita la registrazione delle risposte senza quella dei contatti.

Valore predefinito

180

Valori validi

Qualsiasi numero intero superiore a zero.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

maxJDBCInsertBatchSize

Descrizione

Il numero massimo di record di un batch JDBC, prima di eseguire il commit della query. Non si tratta del numero massimo di record che il modulo della cronologia dei contatti e delle risposte di Interact elabora in un'iterazione. Durante ogni iterazione, il modulo della cronologia dei contatti e delle risposte di Interact elabora tutti i record disponibili dalle tabelle di staging. Tuttavia, tutti quei record vengono suddivisi in blocchi in base al valore `maxJDBCInsertSize`.

Valore predefinito

1000

Valori validi

Qualsiasi numero intero superiore a zero.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

maxJDBCFetchBatchSize

Descrizione

Il numero massimo di record di un batch JDBC da estrarre dal database di staging. Potrebbe essere necessario aumentare questo valore per ottimizzare le prestazioni del modulo della cronologia dei contatti e delle risposte.

Ad esempio, per elaborare 2.5 milioni di registrazioni della cronologia di contatti al giorno, impostare `maxJDBCFetchBatchSize` su un numero superiore a 2.5M, in modo che tutti i record di un giorno vengano elaborati.

Si potrebbero, quindi, impostare le proprietà `maxJDBCFetchChunkSize` e `maxJDBCInsertBatchSize` su valori inferiori (in questo esempio, magari 50.000 e 10.000, rispettivamente). È possibile che vengano elaborati anche alcuni record pianificati per il giorno successivo, ma verranno conservati fino a quel giorno.

Valore predefinito

1000

Valori validi

Qualsiasi numero intero superiore a zero.

maxJDBCFetchChunkSize

Descrizione

Il valore massimo della dimensione di un blocco JDBC di dati letti durante l'ETL (extract, transform, load). In alcuni casi, una dimensione di blocco superiore alla dimensione di inserimento può aumentare la velocità del processo ETL.

Valore predefinito

1000

Valori validi

Qualsiasi numero intero superiore a zero.

deleteProcessedRecords

Descrizione

Specifica se conservare i record della cronologia dei contatti e delle risposte dopo la loro elaborazione.

Valore predefinito

Sì

completionNotificationScript

Descrizione

Specifica il percorso assoluto per uno script da eseguire al completamento dell'ETL. Se si specifica uno script, vengono trasmessi cinque argomenti allo script di notifica del completamento: ora di inizio, ora di fine, numero totale dei record CH elaborati, numero totale di record RH elaborati e stato. L'ora di inizio e l'ora di fine sono valori numerici che rappresentano il numero di millisecondi trascorsi dal 1970. L'argomento stato indica se il

job ETL ha avuto esito positivo o negativo. 0 indica un job ETL riuscito. 1 indica la non riuscita e che ci sono errori nel job ETL.

Valore predefinito

Nessuno

fetchSize

Descrizione

Consente di impostare il valore fetchSize di JDBC, quando si richiamano i record dalle tabelle di staging.

Soprattutto nei database Oracle, adattare l'impostazione al numero di record che JDBC dovrebbe recuperare con ciascun passaggio in rete. Per batch di grandi dimensioni, ovvero 100K o più, provare ad impostare il valore 10000. Fare attenzione a non utilizzare un valore troppo grande, perché questo avrà un impatto sull'utilizzo della memoria e i vantaggi diventerebbero trascurabili o potrebbe addirittura essere dannoso.

Valore predefinito

Nessuno

daysBackInHistoryToLookupContact

Descrizione

Limita i record che è possibile ricercare durante le query della cronologia delle risposte a quelli registrati entro il numero specificato di giorni precedenti. Per database con un gran numero di record della cronologia delle risposte, in questo modo si riesce a ridurre il tempo di elaborazione durante le query, limitando il periodo di ricerca al numero di giorni specificato.

Il valore predefinito 0 indica che la ricerca è estesa a tutti i record.

Valore predefinito

0 (zero)

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

Queste proprietà di configurazione definiscono l'origine dati per il modulo della cronologia dei contatti e delle risposte di Interact.

jndiName

Descrizione

Utilizzare la proprietà `systemTablesDataSource` per identificare l'origine dati JNDI (Java Naming and Directory Interface) definita nel server delle applicazioni (Websphere o WebLogic) per le tabelle di Interact.

Il database di runtime Interact è il database è il database in cui gli script dll `aci_runtime` e `aci_populate_runtime` inseriscono i dati e, ad esempio, contiene le seguenti tabelle (tra le altre): `UACI_CHOfferAttrib` e `UACI_DefaultedStat`.

Valore predefinito

Non è specificato alcun valore predefinito.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

databaseType

Descrizione

Tipo di database per l'origine dati di runtime di Interact.

Valore predefinito

SQLServer

Valori validi

SQLServer | Oracle | DB2

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

schemaName

Descrizione

Il nome dello schema contenente le tabelle di staging del modulo della cronologia dei contatti e delle risposte. Dovrebbe corrispondere al nome dello schema contenente le tabelle dell'ambiente di runtime.

Non è necessario definire uno schema.

Valore predefinito

Non è specificato alcun valore predefinito.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

Queste proprietà di configurazione definiscono il tipo di contatto dalla campagna che si associa ad un 'contatto' per scopi di report o di apprendimento.

contacted

Descrizione

Il valore assegnato alla colonna `ContactStatusID` della tabella `UA_DtlContactHist` nelle tabelle di sistema Campaign per un contatto di offerta. Il valore deve essere una voce valida nella tabella `UA_ContactStatus`. Consultare il manuale *Campaign Administrator's Guide* per dettagli sull'aggiunta dei tipi di contatto.

Valore predefinito

2

Valori validi

Un numero intero superiore a zero.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

Queste proprietà di configurazione definiscono le risposte relative all'accettazione o al rifiuto per il reporting e l'apprendimento.

accept

Descrizione

Il valore assegnato alla colonna `ResponseTypeID` della tabella `UA_ResponseHistory` nelle tabelle di sistema Campaign per un'offerta accettata. Il valore deve essere una voce valida nella tabella `UA_UsrResponseType`. È necessario assegnare alla colonna `CountsAsResponse` il valore `1`, una risposta.

Consultare il manuale *Campaign Administrator's Guide* per dettagli sull'aggiunta dei tipi di risposta.

Valore predefinito

3

Valori validi

Un numero intero superiore a zero.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

reject

Descrizione

Il valore assegnato alla colonna `ResponseTypeID` della tabella `UA_ResponseHistory` nelle tabelle di sistema Campaign per un'offerta rifiutata. Il valore deve essere una voce valida nella tabella `UA_UsrResponseType`. È necessario assegnare alla colonna `CountsAsResponse` il valore `2`, un rifiuto. Consultare il manuale *Campaign Administrator's Guide* per dettagli sull'aggiunta dei tipi di risposta.

Valore predefinito

8

Valori validi

Qualsiasi numero intero superiore a zero.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | report

Queste proprietà di configurazione definiscono i nomi dei report quando si integrano con Cognos.

interactiveCellPerformanceByOfferReportName

Descrizione

Nome per il report Prestazioni cella interattiva per offerta. Questo nome deve corrispondere al nome di tale report nel server Cognos.

Valore predefinito

Prestazioni cella interattiva per offerta

treatmentRuleInventoryReportName

Descrizione

Nome per il report Inventario regola di trattamento. Questo nome deve corrispondere al nome di tale report nel server Cognos.

Valore predefinito

Inventario regola di trattamento canale

deploymentHistoryReportName

Descrizione

Nome per il report Cronologia di distribuzione. Questo nome deve corrispondere al nome di tale report nel server Cognos.

Valore predefinito

Cronologia distribuzione canale

Campaign | partitions | partition[n] | Interact | learning

Queste proprietà di configurazione consentono di regolare il modulo di apprendimento integrato.

confidenceLevel

Descrizione

Un valore percentuale che indica il livello di funzionamento desiderato per il programma di utilità di apprendimento, prima di passare dall'esplorazione all'utilizzo. Un valore 0 elimina di fatto l'esplorazione.

Questa proprietà è disponibile solo se la proprietà `Interact > offerserving > optimizationType` per il runtime Interact è impostata su `BuiltInLearning`.

Valore predefinito

95

Valori validi

Un numero intero compreso tra 0 e 95 ,divisibile per 5 o 99.

validateonDeployment

Descrizione

Se impostato su `No`, Interact non convalida il modulo di apprendimento durante la distribuzione. Se impostato su `Sì`, Interact convalida il modulo di apprendimento durante la distribuzione.

Valore predefinito

`No`

Valori validi

Yes | No

maxAttributeNames

Descrizione

I numero massimo di attributi di apprendimento monitorati dal programma di utilità di apprendimento di Interact.

Questa proprietà è disponibile solo se la proprietà `Interact > offerserving > optimizationType` per il runtime Interact è impostata su `BuiltInLearning`.

Valore predefinito

`10`

Valori validi

Qualsiasi numero intero.

maxAttributeValues

Descrizione

Il numero massimo di valori di cui il modulo di apprendimento di Interact tiene traccia per ciascun attributo di apprendimento.

Questa proprietà è disponibile solo se la proprietà `Interact > offerserving > optimizationType` per il runtime Interact è impostata su `BuiltInLearning`.

Valore predefinito

5

otherAttributeValue

Descrizione

Il nome predefinito per il valore dell'attributo utilizzato per rappresentare tutti i valori degli attributi oltre `maxAttributeValues`.

Questa proprietà è disponibile solo se la proprietà `Interact > offerserving > optimizationType` per il runtime Interact è impostata su `BuiltInLearning`.

Valore predefinito

`Other`

Valori validi

Una stringa o un numero.

Esempio

Se maxAttributeValues è impostato su 3 e otherAttributeValue è impostato su other, il modulo di apprendimento tiene traccia dei primi tre valori. Tutti gli altri valori vengono assegnati alla categoria other. Ad esempio, se si sta tenendo traccia dell'attributo visitatore relativo al colore dei capelli e i primi cinque visitatori hanno colore di capelli nero, castano, biondo, rosso e grigio, il programma di utilità di apprendimento tiene traccia dei colori di capelli nero, castano e biondo. I colori rosso e grigio vengono raggruppati in `otherAttributeValue`, other.

percentRandomSelection

Descrizione

La percentuale di volte in cui un modulo di apprendimento presenta un'offerta casuale. Ad esempio, impostare percentRandomSelection su 5 indica che il 5% delle volte (5 ogni 100 raccomandazioni), il modulo di apprendimento presenta un'offerta casuale, indipendentemente dal punteggio. L'abilitazione di `percentRandomSelection` sovrascrive la proprietà di configurazione `offerTieBreakMethod`. Quando `percentRandomSelection` è abilitato, questa proprietà è impostata indipendentemente dal fatto che l'apprendimento è attivo o disattivo o se è utilizzato l'apprendimento esterno o integrato.

Valore predefinito

5

Valori validi

Qualsiasi numero intero, da 0 (che disabilita la funzione `percentRandomSelection`) fino ad un massimo di 100.

recencyWeightingFactor

Descrizione

La rappresentazione decimale di un valore percentuale della serie di dati definita dalla proprietà `recencyWeightingPeriod`. Ad esempio, il valore predefinito .15 indica che il 15% dei dati utilizzati dal programma di utilità di apprendimento deriva dalla proprietà `recencyWeightingPeriod`.

Questa proprietà è disponibile solo se la proprietà `Interact > offerserving > optimizationType` per il runtime Interact è impostata su `BuiltInLearning`.

Valore predefinito

0.15

Valori validi

Un valore decimale inferiore a 1.

recencyWeightingPeriod

Descrizione

La dimensione in ore dei dati, con percentuale di peso `recencyWeightingFactor`, utilizzati dal modulo di apprendimento. Ad esempio, il valore predefinito di 120 indica che una quantità di dati corrispondente al valore `recencyWeightingFactor`, utilizzata dal modulo di apprendimento, è stata ricavata dalle ultime 120 ore.

Questa proprietà è disponibile solo se `optimizationType` è impostata su `builtInLearning`.

Valore predefinito

120

minPresentCountThreshold

Descrizione

Il numero minimo di volte in cui un'offerta deve essere presentata, prima che i relativi dati vengano utilizzati nei calcoli e il modulo di apprendimento entri in modalità di esplorazione.

Valore predefinito

0

Valori validi

Un numero intero superiore o uguale a zero.

enablePruning

Descrizione

Se impostata su `Yes`, il modulo di apprendimento di Interact determina algoritmicamente quando un attributo di apprendimento (standard o dinamico) non è di previsione. Se un attributo di apprendimento non è di previsione, il modulo di apprendimento non considererà quell'attributo durante la determinazione del peso di un'offerta. Questo continua fino a quando il modulo di apprendimento non aggrega i dati di apprendimento.

Se impostata su `No`, il modulo di apprendimento utilizza sempre tutti gli attributi di apprendimento. Non eliminando gli attributi non predittivi, il modulo di apprendimento potrebbe non essere accurato come dovrebbe.

Valore predefinito

Sì

Valori validi

Yes | No

Campaign | partitions | partition[n] | Interact | learning | learningAttributes | [learningAttribute]

Queste proprietà di configurazione definiscono gli attributi di apprendimento.

attributeName

Descrizione

Ogni `attributeName` è il nome di un attributo visitatore che si desidera venga monitorato dal modulo di apprendimento. Deve corrispondere al nome di una coppia nome-valore nei dati della sessione.

Questa proprietà è disponibile solo se la proprietà `Interact > offerserving > optimizationType` per il runtime Interact è impostata su `BuiltInLearning`.

Valore predefinito

Non è specificato alcun valore predefinito.

Campaign | partitions | partition[n] | Interact | deployment

Queste proprietà di configurazione definiscono le impostazioni di distribuzione.

chunkSize

Descrizione

La dimensione massima della frammentazione in KB per ciascun package di distribuzione di Interact.

Valore predefinito

500

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup]

Queste proprietà di configurazione definiscono le impostazioni del gruppo di server.

serverGroupName

Descrizione

Il nome del gruppo dei server di runtime di Interact. Questo è il nome visualizzato nella scheda di riepilogo del canale interattivo.

Valore predefinito

Non è specificato alcun valore predefinito.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

Queste proprietà di configurazione definiscono i server di runtime Interact.

instanceURL

Descrizione

L'URL del server di runtime Interact. Un gruppo di server può contenere diversi server di runtime Interact; tuttavia, ogni server deve essere creato sotto una nuova categoria.

Valore predefinito

Non è specificato alcun valore predefinito.

Esempio

`http://server:porta/interact`

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | flowchart

Queste proprietà di configurazione definiscono l'ambiente di runtime di Interact utilizzato per le esecuzioni di test di diagrammi di flusso interattivi.

serverGroup

Descrizione

Il nome del gruppo di server Interact che Campaign utilizza per eseguire un'esecuzione di test. Questo nome deve corrispondere al nome categoria creato in `serverGroups`.

Valore predefinito

Non è specificato alcun valore predefinito.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

dataSource

Descrizione

Utilizzare la proprietà `dataSource` per identificare l'origine dei dati fisica che Campaign può utilizzare durante lo svolgimento delle esecuzioni di test di diagrammi di flusso interattivi. Questa proprietà dovrebbe corrispondere all'origine dei dati definita mediante la proprietà `Campaign > partitions > partitionN > dataSources` per l'origine dei dati dell'esecuzione di test definita per la fase di progettazione di Interact.

Valore predefinito

Non è specificato alcun valore predefinito.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

eventPatternPrefix

Descrizione

La proprietà `eventPatternPrefix` è un valore di stringa che viene anteposto a nomi di pattern di evento, per consentirne l'utilizzo in espressioni nei processi Seleziona o Decisione, nell'ambito dei diagrammi di flusso interattivi.

Si tenga presente che, in caso di modifica di questo valore, sarà necessario distribuire modifiche globali nel canale interattivo per rendere effettiva questa configurazione aggiornata.

Valore predefinito

`EventPattern`

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

Queste proprietà di configurazione definiscono il codice cella predefinito per la tabella delle offerte predefinita. È necessario configurare queste proprietà solo se si stanno definendo assegnazioni di offerta globali.

DefaultCellCode

Descrizione

Il codice cella predefinito che Interact utilizza se non si definisce un codice cella nella tabella delle offerte predefinita.

Valore predefinito

Non è specificato alcun valore predefinito.

Valori validi

Una stringa che corrisponde al formato del codice cella definito in Campaign

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

Queste proprietà di configurazione definiscono il codice cella predefinito per la tabella offersBySQL. È necessario configurare queste proprietà solo se si utilizzano query SQL per ottenere una serie desiderata di offerte candidate.

DefaultCellCode

Descrizione

Il codice cella predefinito che Interact utilizza per qualsiasi offerta nelle tabelle OffersBySQL con valore null nella colonna del codice cella (o se la colonna del codice cella manca completamente). Tale valore deve essere un codice cella valido.

Valore predefinito

Non è specificato alcun valore predefinito.

Valori validi

Una stringa che corrisponde al formato del codice cella definito in Campaign

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

Queste proprietà di configurazione definiscono il codice cella predefinito per la tabella di sovrascrittura dei punteggi. È necessario configurare queste proprietà solo se si definiscono singole assegnazioni dell'offerta.

DefaultCellCode

Descrizione

Il codice cella predefinito utilizzato da Interact, se non si definisce un codice cella nella tabella di sovrascrittura dei punteggi.

Valore predefinito

Non è specificato alcun valore predefinito.

Valori validi

Una stringa che corrisponde al formato del codice cella definito in Campaign

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | server | internal

Le proprietà di questa categoria specificano le impostazioni di integrazione e i limiti degli ID interni per la partizione Campaign selezionata. Se l'installazione Campaign ha più partizioni, impostare queste proprietà per ciascuna partizione su cui si desidera vengano applicate.

internalIdLowerLimit

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Le proprietà internalIdUpperLimit e internalIdLowerLimit vincolano gli ID interni di Campaign a rientrare nell'intervallo specificato. Si tenga presente che i valori sono inclusivi: vale a dire, Campaign può utilizzare sia il limite inferiore che quello superiore.

Valore predefinito

0 (zero)

internalIdUpperLimit

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Le proprietà internalIdUpperLimit e internalIdLowerLimit vincolano gli ID interni di Campaign a rientrare nell'intervallo specificato. I valori sono inclusivi: vale a dire, Campaign può utilizzare sia il limite inferiore che quello superiore. Se è installato Distributed Marketing, impostare il valore su 2147483647.

Valore predefinito

4294967295

eMessageInstalled

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Indica che eMessage è installato. Quando si seleziona Yes, le funzioni eMessage sono disponibili nell'interfaccia Campaign.

Il programma di installazione di IBM imposta questa proprietà su Yes per la partizione predefinita nella propria installazione di eMessage. Per partizioni aggiuntive nell'ubicazione di installazione di eMessage, è necessario configurare manualmente questa proprietà.

Valore predefinito

No

Valori validi

Yes | No

interactInstalled

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Dopo aver installato l'ambiente di progettazione di Interact, questa proprietà di configurazione dovrebbe essere impostata su Yes, per abilitare l'ambiente di progettazione di Interact in Campaign.

Se Interact non è installato, impostarla su No. Impostando questa proprietà su No non si rimuovono menu e opzioni di Interact dall'interfaccia utente. Per rimuovere i menu e le opzioni, è necessario annullare manualmente la registrazione di Interact mediante il programma di utilità configTool.

Valore predefinito

No

Valori validi

Yes | No

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

MO_UC_integration

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Abilita l'integrazione con Marketing Operations per questa partizione, se l'integrazione è abilitata nelle impostazioni della configurazione di **Platform**. Per ulteriori informazioni, consultare il manuale *IBM Marketing Operations and Campaign Integration Guide*.

Valore predefinito

No

Valori validi

Yes | No

MO_UC_BottomUpTargetCells

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Per questa partizione, consente celle bottom-up per i fogli di calcolo delle celle obiettivo, se è abilitata la proprietà **MO_UC_integration**. Quando la proprietà è impostata su Yes, sono visibili le celle obiettivo sia top-down che bottom-up, ma quelle bottom-up sono di sola lettura. Per ulteriori informazioni, consultare il manuale *IBM Marketing Operations and Campaign Integration Guide*.

Valore predefinito

No

Valori validi

Yes | No

Legacy_campaigns

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Per questa partizione, consente l'accesso a campagne create prima che Marketing Operations e Campaign fossero integrati. si applica solo se la proprietà **MO_UC_integration** è impostata su Yes. Le campagne eredità includono anche campagne create in Campaign 7.x e collegate a progetti Plan 7.x. Per ulteriori informazioni, consultare *IBM Marketing Operations e Campaign Guida di integrazione*.

Valore predefinito

No

Valori validi

Yes | No

IBM Marketing Operations - Integrazione offerta

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Consente di utilizzare Marketing Operations per eseguire attività di gestione del ciclo di vita dell'offerta su questa partizione, se per tale partizione è abilitata la proprietà **MO_UC_integration**. L'integrazione dell'offerta deve essere abilitata nelle proprie impostazioni della configurazione di **Platform**. Per ulteriori informazioni, consultare *IBM Marketing Operations e Campaign Guida di integrazione*.

Valore predefinito

No

Valori validi

Yes | No

UC_CM_integration

Categoria configurazione

Campaign|partitions|partition[n]|server|internal

Descrizione

Consente l'integrazione del segmento in linea Digital Analytics per una partizione Campaign. Se si imposta questo valore su Yes, la casella del processo Seleziona in un diagramma di flusso fornisce l'opzione per selezionare **Segmenti Digital Analytics** come input. Per configurare l'integrazione di Digital Analytics per ciascuna partizione, selezionare **Impostazioni > Configurazione > Campaign | partitions | partition[n] | Coremetrics**.

Valore predefinito

No

Valori validi

Yes | No

numRowsReadToParseDelimitedFile

Categoria configurazione

`Campaign|partitions|partition[n]|server|internal`

Descrizione

Questa proprietà viene utilizzata quando si associa un file delimitato come una tabella utente. Viene utilizzata anche dalla casella del processo Punteggio quando si importa un file di output del punteggio da IBM SPSS Modeler Advantage Enterprise Marketing Management Edition. Per importare o associare un file delimitato, Campaign deve analizzare il file per identificare le colonne, i tipi di dati (tipi di campo) e le larghezze delle colonne (lunghezze dei campi).

Il valore predefinito 100 indica che Campaign esamina le prime 50 e le ultime 50 voci di riga nel file delimitato. Campaign alloca quindi la lunghezza del campo in base al valore più grande rilevato in queste voci. Nella maggior parte dei casi, il valore predefinito è sufficiente per determinare le lunghezze dei campi. Tuttavia, nei file delimitati molto grandi, un campo successivo potrebbe superare la lunghezza stimata calcolata da Campaign e ciò potrebbe causare un errore durante il runtime del diagramma di flusso. Quindi, se si sta associando un file molto grande, è possibile aumentare questo valore per far sì che Campaign controlli più voci di riga. Ad esempio, un valore pari a 200 fa sì che Campaign esamini le prime 100 e le ultime 100 voci di riga del file.

Un valore pari a 0 consente di esaminare l'intero file. Di norma, questo valore viene utilizzato se si sta importando o associando file che hanno ampiezze di dati dei campi variabili che non possono essere identificate leggendo solo poche righe iniziali e finali. La lettura dell'intero file per file di dimensioni estremamente grandi può aumentare il tempo di elaborazione richiesto per l'esecuzione del mapping della tabella e della casella del processo Punteggio.

Valore predefinito

100

Valori validi

0 (tutte le righe) o qualsiasi valore interno positivo

Campaign | monitoring

Le proprietà di questa categoria specificano se è abilitata la funzione di monitoraggio operativo, l'URL del server di monitoraggio operativo e il comportamento di memorizzazione nella cache. Il monitoraggio operativo presenta e consente di controllare diagrammi di flusso attivi.

cacheCleanupInterval

Descrizione

La proprietà `cacheCleanupInterval` specifica l'intervallo, in secondi, tra le ripuliture automatiche della cache di stato del diagramma di flusso.

Questa proprietà non è disponibile nelle versioni di Campaign precedenti alla versione 7.0.

Valore predefinito

600 (10 minuti)

cacheRunCompleteTime

Descrizione

La proprietà cacheRunCompleteTime specifica la durata, in minuti, della memorizzazione delle esecuzioni completate nella cache e della visualizzazione nella pagina Monitoraggio.

Questa proprietà non è disponibile nelle versioni di Campaign precedenti alla versione 7.0.

Valore predefinito

4320

monitorEnabled

Descrizione

La proprietà monitorEnabled specifica se il monitoraggio è attivato.

Questa proprietà non è disponibile nelle versioni di Campaign precedenti alla versione 7.0.

Valore predefinito

FALSE

Valori validi

TRUE | FALSE

serverURL

Descrizione

La proprietà Campaign > monitoring > serverURL specifica l'URL del server del monitoraggio operativo. Questa è un'impostazione obbligatoria; modificare il valore se l'URL del server del monitoraggio operativo non è quello predefinito.

Se Campaign è configurato per utilizzare le comunicazioni SSL (Secure Sockets Layer), impostare il valore di questa proprietà per l'utilizzo di HTTPS. Ad esempio: serverURL=https://*host:SSL_port*/Campaign/OperationMonitor dove:

- *host* è il nome o l'indirizzo IP della macchina su cui l'applicazione web è installata
- *SSL_Port* è la porta SSL dell'applicazione web.

Si noti la presenza nell'URL di https.

Valore predefinito

http://localhost:7001/Campaign/OperationMonitor

monitorEnabledForInteract

Descrizione

Se questa proprietà è impostata su TRUE, abilita il server del connettore JMX di Campaign per Interact. Campaign non dispone della sicurezza JMX.

Se questa proprietà è impostata su FALSE, non sarà possibile connettersi al server del connettore JMX di Campaign.

Questo monitoraggio JMX è relativo solo al modulo della cronologia dei contatti e delle risposte di Interact.

Valore predefinito

FALSE

Valori validi

TRUE | FALSE

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

protocol

Descrizione

Protocollo di ascolto per il server del connettore JMX di Campaign, se la proprietà monitorEnabledForInteract è impostata su yes.

Questo monitoraggio JMX è relativo solo al modulo della cronologia dei contatti e delle risposte di Interact.

Valore predefinito

JMXMP

Valori validi

JMXMP | RMI

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

port

Descrizione

La porta di ascolto per il server del connettore JMX Campaign, se la proprietà monitorEnabledForInteract è impostata su yes.

Questo monitoraggio JMX è relativo solo al modulo della cronologia dei contatti e delle risposte di Interact.

Valore predefinito

2004

Valori validi

Un numero intero compreso tra 1025 e 65535.

Disponibilità

Questa proprietà è disponibile solo se è stato installato Interact.

Campaign | partitions | partition[n] | Interact | outboundChannels

Queste proprietà di configurazione consentono di regolare i canali in uscita per i messaggi attivati.

nome categoria

Descrizione

Questa proprietà definisce il nome di questo canale in uscita. Il nome deve essere univoco tra tutti i canali in uscita.

dominio

Descrizione

Il nome del canale in uscita.

Nota: È necessario riavviare il server delle applicazioni affinché i cambiamenti diventino effettivi.

Campaign | partitions | partition[n] | Interact | outboundChannels | Parameter Data

Queste proprietà di configurazione consentono di regolare i canali in uscita per i messaggi attivati.

nome categoria

Descrizione

Questa proprietà definisce il nome di questo parametro. Il nome deve essere univoco tra tutti i parametri per quel canale in uscita.

value

Descrizione

Questa proprietà definisce i parametri, nel formato di coppie nome-valore, necessari per questo gateway in uscita.

Campaign | partitions | partition[n] | Interact | Simulator

Queste proprietà di configurazione definiscono il gruppo di server che si desidera utilizzare per eseguire le simulazioni di API.

serverGroup

Descrizione

Specificare il gruppo dei server di runtime utilizzato per eseguire le simulazioni di API.

Valore predefinito

defaultServerGroup

Capitolo 15. Personalizzazione dell'offerta in tempo reale sul lato client

Potrebbero verificarsi situazioni in cui si desidera fornire una personalizzazione dell'offerta in tempo reale senza implementare codice Java di basso livello o chiamate SOAP al server Interact. Ad esempio, quando un visitatore inizialmente carica una pagina web dove il contenuto Javascript è l'unica programmazione estesa disponibile, o quando un visitatore apre un messaggio di posta elettronica in cui è possibile solo contenuto HTML. IBM Interact offre diversi connettori che forniscono una gestione delle offerte in tempo reale in situazioni in cui l'utente ha il controllo solo sul contenuto web caricato sul lato client, o situazioni in cui si desidera semplificare l'interfaccia a Interact.

L'installazione di Interact include due connettori per la personalizzazione delle offerte che viene avviata sul lato client:

- "Informazioni sul connettore del messaggio di Interact". Utilizzando il connettore del messaggio, il contenuto web nei messaggi di posta elettronica (ad esempio) o altri supporti elettronici possono contenere tag di immagine e di link per effettuare chiamate al server Interact per la presentazione di offerte con caricamento pagina e pagine di arrivo con click-through.
- "Informazioni su Interact Web Connector" a pagina 320. Utilizzando il Web Connector (detto anche connettore JS) le pagine web possono utilizzare JavaScript sul lato client per gestire l'arbitraggio delle offerte, la presentazione e la cronologia dei contatti/delle risposte tramite la presentazione di offerte con caricamento pagina e pagine di arrivo con click-through.

Informazioni sul connettore del messaggio di Interact

Il connettore del messaggio di Interact consente ai messaggi di posta elettronica ed altri supporti elettronici di effettuare chiamate a IBM Interact per consentire offerte personalizzate da presentare all'apertura, e quando il cliente esegue il click-through del messaggio verso il sito specificato. Questo si ottiene tramite l'utilizzo di due tag chiave: il tag di immagine (`IMG`), che carica le offerte personalizzate durante l'apertura, e il tag di link (`A`), che cattura le informazioni relative al click-through e reindirizza il cliente ad una specifica pagina di arrivo.

Esempio

Il seguente esempio mostra codice HTML che si potrebbe includere in uno spot di marketing (ad esempio, all'interno di un messaggio di posta elettronica) che contiene un URL di tag `IMG` (che passa informazioni quando il documento viene aperto per il server Interact e richiama l'immagine dell'offerta appropriata in risposta) e un URL di tag `A` (che determina quali informazioni vengono passate al server Interact al click-through):

```
<a href="http://www.example.com/MessageConnector/
offerClickthru.jsp?msgId=1234&linkId=1&userid=1&referral=test"><img
src="http:// www.example.com/MessageConnector/offerImage.jsp?msgId=1234
&linkId=1&userid=1&incomeLevel=5&incomeType=numeric"/></a>
```

Nel seguente esempio, un tag `IMG` è incluso in un tag `A`, che causa il seguente comportamento:

1. Quando il messaggio di posta elettronica viene aperto, il connettore del messaggio riceve una richiesta che contiene le informazioni codificate nel tag `IMG`: il `msgID` e il `linkID` del messaggio e i parametri del cliente che includono userid, livello di reddito e tipo di reddito.
2. Queste informazioni vengono trasmesse tramite una chiamata API al server di runtime di Interact.
3. Il server di runtime restituisce un'offerta al connettore del messaggio che recupera l'URL dell'immagine dell'offerta, fornisce quell'URL (con inclusi ulteriori parametri) e reindirizza la richiesta di immagine all'URL dell'offerta.
4. Il cliente vede l'offerta come immagine.

A quel punto il cliente può fare clic sull'immagine per rispondere all'offerta in qualche modo. Il click-through, con l'utilizzo del tag `A` e del relativo attributo specificato `HREF` (che specifica l'URL di destinazione) invia un'altra richiesta al connettore del messaggio di una pagina di arrivo collegata all'URL di quell'offerta. Il browser dei clienti viene quindi reindirizzato alla pagina di arrivo configurata nell'offerta.

Tenere presente che un tag `A` click-through non è strettamente necessario; l'offerta può essere costituita dalla sola immagine, come un coupon che il cliente può stampare.

Installazione del connettore del messaggio

I file necessari per installare, distribuire ed eseguire il connettore del messaggio sono stati inclusi automaticamente con l'installazione del server di runtime di IBM Interact. Questa sezione riepiloga gli step necessari per approntare il connettore del messaggio.

L'installazione e la distribuzione del connettore del messaggio implica le seguenti attività:

- Facoltativamente, la configurazione delle impostazioni predefinite per il connettore del messaggio come descritto in "Configurazione del connettore del messaggio".
- Creazione delle tabelle del database necessarie per memorizzare i dati di transazione del connettore del messaggio come descritto in "Creazione delle tabelle del connettore del messaggio" a pagina 315.
- Installazione dell'applicazione web connettore del messaggio come descritto in "Distribuzione ed esecuzione del connettore del messaggio" a pagina 316.
- Creazione di tag `IMG` e `A` negli spot di marketing (email o pagine web, ad esempio) necessari per richiamare le offerte del connettore del messaggio all'apertura e al click-through, come descritto in "Creazione dei link del connettore del messaggio" a pagina 317.

Configurazione del connettore del messaggio

Prima di distribuire il connettore del messaggio, è necessario personalizzare il file di configurazione incluso nell'installazione in modo che abbia corrispondenza con lo specifico ambiente. È possibile modificare il file XML denominato `MessageConnectorConfig.xml` che si trova nella directory del connettore del messaggio sul server di runtime di Interact, simile a `<Interact_home>/msgconnector/config/MessageConnectorConfig.xml`.

Informazioni su questa attività

Il file `MessageConnectorConfig.xml` contiene alcune impostazioni di configurazione obbligatorie ed altre facoltative. Le impostazioni utilizzate devono essere personalizzate per la specifica installazione. Seguire la procedura riportata di seguito per modificare la configurazione.

Procedura

1. Se il connettore del messaggio è distribuito e in esecuzione sul server delle applicazioni web, rimuoverne la distribuzione prima di continuare.
2. Sul server di runtime di Interact, aprire il file `MessageConnectorConfig.xml` in qualsiasi editor di testo o di XML.
3. Modificare le impostazioni della configurazione in base alle esigenze, assicurandosi che le seguenti impostazioni *obbligatorie* siano corrette per l'installazione.
   - `<interactUrl>`, l'URL del server di runtime di Interact al quale i tag del connettore del messaggio devono connettersi, e su cui il connettore del messaggio è in esecuzione.
   - `<imageErrorLink>`, l'URL al quale il connettore del messaggio verrà reindirizzato se si verifica un errore durante l'elaborazione di una richiesta di un'immagine di offerta.
   - `<landingPageErrorLink>`, l'URL a cui il connettore del messaggio verrà reindirizzato se si verifica un errore durante l'elaborazione di una richiesta di pagina di arrivo di un'offerta.
   - `<audienceLevels>`, una sezione del file di configurazione che contiene una o più serie di impostazioni di livello destinatario, e che specifica il livello destinatario predefinito se il link del connettore del messaggio non ne specifica uno. Deve essere configurato almeno un livello destinatario.

   Tutte le impostazioni della configurazione sono descritte con maggiori dettagli in “Impostazioni di configurazione del connettore del messaggi”.
4. Dopo aver completato le modifiche alla configurazione, salvare e chiudere il file `MessageConnectorConfig.xml`.
5. Continuare l'impostazione e la distribuzione del connettore del messaggio.

Impostazioni di configurazione del connettore del messaggi:

Per configurare il connettore del messaggio, è possibile modificare il file XML denominato `MessageConnectorConfig.xml` trovato nella directory del connettore del messaggio sul server di runtime Interact, in genere `<Interact_home>/msgconnector/config/MessageConnectorConfig.xml`. Ognuna delle configurazioni contenuta in questo file XML è descritta qui. Tenere presente che se si modifica questo file dopo che il connettore del messaggio è stato distribuito ed è in esecuzione, assicurarsi di annullare la distribuzione del connettore del messaggio ed eseguirla di nuovo oppure riavviare il server delle applicazioni per ricaricare queste impostazioni dopo aver terminato le modifiche al file.

Impostazioni generali

La seguente tabella contiene un elenco delle impostazioni facoltative e obbligatorie incluse nella sezione `generalSettings` del file `MessageConnectorConfig.xml`.

Tabella 24. Impostazioni generali del connettore del messaggio

Elemento	Descrizione	Valore predefinito
`<interactURL>`	L'URL del server di runtime Interact per gestire le chiamate da dai tag della pagina del connettore del messaggio, ad esempio il server di runtime su cui è in esecuzione il connettore del messaggio. Questo elemento è obbligatorio.	`http://localhost:7001/interact`
`<defaultDateTimeFormat>`	Il formato predefinito della data.	`MM/dd/yyyy`
`<log4jConfigFileLocation>`	L'ubicazione del file delle proprietà Log4j. È relativo alla variabile di ambiente *$MESSAGE_CONNECTOR_HOME*, se è impostata; altrimenti questo valore è relativo al percorso root dell'applicazione web del connettore del messaggio.	`config/ MessageConnectorLog4j.properties`

Valori parametro predefiniti

La seguente tabella contiene un elenco delle impostazioni facoltative e obbligatorie incluse nella sezione `defaultParameterValues` del file `MessageConnectorConfig.xml`.

Tabella 25. Impostazioni predefinite dei parametri del connettore del messaggio

Elemento	Descrizione	Valore predefinito
`<interactiveChannel>`	Il nome del canale interattivo predefinito.	
`<interactionPoint>`	Il nome del punto di interazione predefinito.	
`<debugFlag>`	Determina se il debug è abilitato. I valori consentiti sono `true` e `false`.	`false`
`<contactEventName>`	Il nome predefinito dell'evento contatto inviato.	
`<acceptEventName>`	Il nome predefinito dell'evento accettazione inviato.	
`<imageUrlAttribute>`	Il nome dell'attributo offerta predefinito che contiene l'URL dell'immagine dell'offerta, se non ne è specificato uno nel link del connettore del messaggio.	
`<landingPageUrlAttribute>`	L'URL predefinito della pagina di arrivo del click-through, se non ne è stato specificato uno nel link del connettore del messaggio.	

Impostazioni di comportamento

La seguente tabella contiene un elenco delle impostazioni facoltative e obbligatorie incluse nella sezione `behaviorSettings` del file `MessageConnectorConfig.xml`.

Tabella 26. Impostazioni di comportamento del connettore del messaggio

Elemento	Descrizione	Valore predefinito
<imageErrorLink>	L'URL a cui viene reindirizzato il connettore se si verifica un errore durante l'elaborazione di una richiesta di un'immagine di offerta. Questa impostazione è obbligatoria.	/images/default.jpg
<landingPageErrorLink>	L'URL a cui viene reindirizzato il connettore se si verifica un errore durante l'elaborazione di una richiesta di una pagina di arrivo di click-through. Questa impostazione è obbligatoria.	/jsp/default.jsp
<alwaysUseExistingOffer>	Determina se l'offerta memorizzata nella cache deve essere restituita, anche se è già scaduta. I valori consentiti sono true e false.	false
<offerExpireAction>	L'azione da intraprendere se viene individuata l'offerta originale ma è già scaduta. I valori consentiti sono: • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

Impostazioni di storage

La seguente tabella contiene un elenco delle impostazioni facoltative e obbligatorie incluse nella sezione storageSettings del file MessageConnectorConfig.xml.

Tabella 27. Impostazioni di storage del connettore del messaggio

Elemento	Descrizione	Valore predefinito
<persistenceMode>	Quando la cache rende persistenti le nuove voci nel database. I valori consentiti sono WRITE-BEHIND (dove i dati vengono inizialmente scritti nella cache e aggiornati nel database in un secondo momento) e WRITE-THROUGH (dove i dati vengono scritti nella cache e nel database contemporaneamente).	WRITE-THROUGH
<maxCacheSize>	Il numero massimo di voci nella cache della memoria.	5000
<maxPersistenceBatchSize>	La dimensione batch massima durante la persistenza delle voci nel database.	200
<macCachePersistInterval>	Il periodo di tempo massimo in secondi in cui una voce rimane nella cache prima che sia resa persistente nel database.	3
<maxElementOnDisk>	Il numero massimo di voci nella cache del disco.	5000

Tabella 27. Impostazioni di storage del connettore del messaggio (Continua)

Elemento	Descrizione	Valore predefinito
<cacheEntryTimeToExpireInSeconds>	La quantità massima di tempo a disposizione per le voci nella cache del disco per essere rese persistenti prima di scadere.	60000
<jdbcSettings>	Una sezione del file XML che contiene informazioni specifiche se viene utilizzata una connessione JDBC. È ad esclusione reciproca con la sezione <dataSourceSettings>.	È configurato per impostazione predefinita a connettersi ad un database SQLServer configurato sul server locale, ma se si abilita questa sezione è necessario fornire le impostazioni JDBC effettive e le credenziali per effettuare l'accesso.
<dataSourceSettings>	Una sezione del file XML che contiene informazioni specifiche se viene utilizzata una connessione all'origine dati. È ad esclusione reciproca con la sezione <jdbcSettings>.	È configurato per impostazione predefinita a connettersi all'origine dati InteractDS definita sul server delle applicazioni web locale.

Livelli destinatario

La seguente tabella contiene un elenco delle impostazioni facoltative e obbligatorie incluse nella sezione audienceLevels del file MessageConnectorConfig.xml.

Notare che l'elemento audienceLevels viene utilizzato facoltativamente per specificare il livello destinatario predefinito da utilizzare se non ne è specificato uno nel link del connettore del messaggio, come nel seguente esempio:

```
<audienceLevels default="Customer">
```

In questo esempio, il valore dell'attributo predefinito corrisponde al nome di un audienceLevel definito in questa sezione. Deve essere definito almeno un livello destinatario in questo file di configurazione.

Tabella 28. Impostazioni del livello destinatario del connettore del messaggio

Elemento	Elemento	Descrizione	Valore predefinito
<audienceLevel>		L'elemento che contiene la configurazione del livello destinatario. Fornire un attributo nome, come in <audienceLevel name="Customer">	
	<messageLogTable>	Il nome della tabella di log. Questo valore è obbligatorio.	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<field>	La definizione di uno o più campi ID destinatario per questo audienceLevel.	
	<name>	Il nome del campo ID destinatario, specificato nel runtime di Interact.	
	<httpParameterName>	Il nome parametro corrispondente per questo campo ID destinatario.	

Tabella 28. Impostazioni del livello destinatario del connettore del messaggio (Continua)

Elemento	Elemento	Descrizione	Valore predefinito
	<dbColumnName>	Il nome colonna corrispondente nel database per questo campo ID destinatario.	
	<type>	Il tipo di campo ID destinatario, specificato nel runtime di Interact. I valori possono essere string o numeric.	

Creazione delle tabelle del connettore del messaggio

Prima di poter distribuire il connettore del messaggio di IBM Interact, è necessario creare le tabelle nel database in cui sono memorizzati i dati di runtime di Interact. Creare una tabella per ogni livello destinatario definito. Per ogni livello destinatario, Interact utilizzerà le tabelle create per registrare le informazioni relative alle transazioni del connettore del messaggio.

Informazioni su questa attività

Utilizzare il client database per eseguire lo script SQL del connettore del messaggio sul database o schema appropriato per creare le tabelle necessarie. Gli script SQL per il database supportato vengono installati automaticamente durante l'installazione del server di runtime Interact. Per i dettagli sulla connessione al database che contiene le tabelle di runtime di Interact, consultare i fogli di lavoro completati in *IBM Interact - Guida all'installazione*.

Procedura

1. Avviare il client database e connettersi al database in cui attualmente sono memorizzate le tabella di runtime di Interact.
2. Eseguire lo script appropriato nella directory <*Interact_home*>/msgconnector/scripts/ddl. La tabella riportata di seguito elenca gli script SQL di esempio che possono essere utilizzati per creare manualmente le tabelle del connettore del messaggio:

Tabella 29. Script per la creazione delle tabelle del connettore del messaggio

Tipo di origine dati	Nome dello script
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

Tenere presente che questi script sono forniti come esempi. Si potrebbe utilizzare una convenzione di denominazione o una struttura diversi per i valori degli ID destinatario, quindi potrebbe essere necessario modificare lo script prima di eseguirlo. In generale, è consigliabile avere una tabella dedicata a ciascun livello destinatario.

Le tabelle vengono create per contenere le seguenti informazioni:

Tabella 30. Informazioni create dagli script SQL di esempio

Nome colonna	Descrizione
LogId	La chiave primaria di questa voce.
MessageId	L'identificativo univoco di ciascuna istanza di messaggistica.

Tabella 30. Informazioni create dagli script SQL di esempio (Continua)

Nome colonna	Descrizione
LinkId	L'identificativo univoco di ciascun link nel supporto elettronico (ad esempio un messaggio di posta elettronica).
OfferImageUrl	L'URL dell'immagine correlata dell'offerta restituita.
OfferLandingPageUrl	L'URL della pagina di arrivo correlata dell'offerta restituita.
TreatmentCode	Il codice trattamento dell'offerta restituita.
OfferExpirationDate	La data e l'ora di scadenza dell'offerta restituita.
OfferContactDate	La data e l'ora in cui l'offerta è stata restituita al client.
AudienceId	L'ID destinatario del supporto elettronico.

Considerare quanto segue riguardo alla tabella:

- A seconda del livello destinatario, sarà presente una colonna AudienceId per ogni componente della chiave destinatario.
- La combinazione di MessageId, LinkId e AudienceId(s) forma una chiave univoca per questa tabella.

Quando lo script termina l'esecuzione, sono state create le tabelle necessarie per il connettore del messaggio.

Risultati

Ora si è pronti a distribuire l'applicazione web del connettore del messaggio.

Distribuzione ed esecuzione del connettore del messaggio

Il connettore del messaggio di IBM Interact viene distribuito come applicazione web autonoma su un server di applicazioni web supportato.

Prima di iniziare

Prima di poter distribuire il connettore del messaggio, assicurarsi che le seguenti attività siano state completate:

- È necessario avere installato il server di runtime di IBM Interact. L'applicazione connettore del messaggio distribuibile viene installata automaticamente insieme al server di runtime ed è pronta per essere distribuita dalla directory principale di Interact.
- È necessario inoltre aver eseguito gli script SQL forniti con l'installazione per creare le tabelle necessarie nel database di runtime di Interact per l'utilizzo da parte del connettore del messaggio come descritto in “Creazione delle tabelle del connettore del messaggio” a pagina 315

Informazioni su questa attività

Così come si distribuiscono le altre applicazioni IBM su un server di applicazioni web prima di poterle eseguire, è necessario distribuire l'applicazione connettore del messaggio per renderla disponibile per la presentazione delle offerte.

Procedura

1. Connettersi all'interfaccia di gestione del server delle applicazioni web con i privilegi necessari per distribuire un'applicazione.

2. Seguire le istruzioni affinché il server delle applicazioni web distribuisca ed esegua il file denominato *<Interact_home>*/msgconnector/MessageConnector.war Sostituire *<Interact_home>* con la directory effettiva in cui è installato il server di runtime di Interact.

Risultati

Il connettore del messaggio ora è disponibile per l'utilizzo. Dopo aver configurato l'installazione di Interact per creare i dati sottostanti che il connettore del messaggio utilizzerà per fornire le offerte, come i canali interattivi e le strategie, i diagrammi di flusso, le offerte e così via, è possibile creare i link nei supporti elettronici che il connettore del messaggio accetterà.

Creazione dei link del connettore del messaggio

Per utilizzare il connettore del messaggio per fornire immagini di offerte personalizzate quando l'utente finale interagisce con i supporti elettronici (ad esempio aprendo un messaggio di posta elettronica), e pagine di arrivo personalizzare quando l'utente finale esegue il click-through dell'offerta, è necessario creare i link da includere nel messaggio. Questa sezione fornisce un riepilogo dei tag HTML di tali link.

Informazioni su questa attività

Indipendentemente dal sistema utilizzato per generate i messaggi in uscita per gli utenti finali, è necessario generare i tag HTML che contengono i campi appropriati (forniti nei tag HTML come attributi) con le informazioni che si desidera trasmettere al server di runtime di Interact. Seguire la procedura riportata di seguito per configurare le informazioni minime necessarie per un messaggio del connettore del messaggio.

Tenere presente che anche se le istruzioni fanno riferimento in modo specifico ai messaggi che contengono i link del connettore del messaggio, è possibile seguire la stessa procedura e la stessa configurazione per aggiungere link alle pagine web o qualsiasi altro supporto elettronico.

Procedura

1. Creare il link `IMG` che verrà visualizzato nel messaggio con almeno i seguenti parametri:
   - `msgID`, che indica l'identificativo univoco per questo messaggio.
   - `linkID`, che indica l'identificativo univoco per il link nel messaggio.
   - `audienceID`, l'identificativo del destinatario al quale appartiene il ricevente del messaggio.

     Tenere presente che se l'ID destinatario è un ID composto, tutti quei componenti devono essere inclusi nel link.

   È inoltre possibile includere parametri facoltativi che comprendono il livello destinatario, il nome del canale interattivo,il nome del punto di interazione, l'URL dell'ubicazione dell'immagine e i propri parametri personalizzati non specificatamente utilizzati dal connettore del messaggio.
2. Facoltativamente, creare un link `A` che racchiude il link `IMG` in modo che, quando l'utente fa clic sull'immagine, il browser carichi una pagina contenente l'offerta per l'utente. Il link `A` deve anche contenere i tre parametri elencati elencati prima (`msgID`, `linkID` e `audienceID`), più qualsiasi parametro facoltativo (livello destinatario, nome del canale interattivo e nome del punto interattivo) e

i parametri personalizzati non specificatamente utilizzati dal connettore del messaggio. Tenere presente che il link A molto probabilmente conterrà un link IMG del connettore del messaggio, ma può anche essere autonomo nella pagina, in base alle necessità. Se il link contiene un link IMG, il link IMG deve contenere la stessa serie di parametri del link A che lo racchiude (inclusi gli eventuali parametri facoltativi o personalizzati).

3. Quando i link sono definiti correttamente, generano e inviano i messaggi di posta elettronica.

Risultati

Per informazioni dettagliate sui parametri disponibili, e i link di esempio, consultare “Parametri richiesta HTTP tag "IMG" e "A"”

Parametri richiesta HTTP tag "IMG" e "A"

Quando il connettore del messaggio riceve una richiesta, perché un utente finale ha aperto un'email contenente un tag IMG codificato dal connettore del messaggio o perché l'utente finale ha eseguito un ClickThrough del tag A, analizza i parametri inclusi con la richiesta per restituire i dati offerta appropriati. Questa sezione fornisce un elenco dei parametri che possono essere inclusi nell'URL richiedente (tag IMG (caricato automaticamente quando viene visualizzata un'immagine con tag quando viene aperta l'email) o tag A (caricato quando la persona che sta visualizzando l'email fa clic attraverso il messaggio fino al sito specificato).

Parametri

Quando il connettore del messaggio riceve una richiesta, analizza i parametri inclusi con la richiesta. Questi parametri includono alcuni o tutti i seguenti elementi:

Nome parametro	Descrizione	Obbligatorio?	Valore predefinito
msgId	L'identificativo univoco dell'istanza email o della pagina web.	Sì	Nessuno. Viene fornito dal sistema durante la creazione dell'istanza univoca del messaggio email o della pagina web contenente il tag.
linkId	L'identificativo univoco del link in questa email o pagina web.	Sì	Nessuno. Viene fornito dal sistema durante la creazione dell'istanza univoca del messaggio email o della pagina web contenente il tag.
audienceLevel	Il livello destinatario a cui appartiene il destinatario di questa comunicazione.	No	L'audienceLevel specificato come predefinito nell'elemento audienceLevels trovato nel file MessageConnectorConfig.xml.
ic	Il nome del canale interattivo (IC) obiettivo	No	Il valore dell'elemento interactiveChannel trovato nella sezione defaultParameterValues del file MessageConnectorConfig.xml, che per impostazione predefinita è "interactiveChannel".
ip	il nome del punto di interazione (IP) di applicazione	No	Il valore dell'elemento interactionPoint trovato nella sezione defaultParameterValues del file MessageConnectorConfig.xml, che per impostazione predefinita è "headBanner".
offerImageUrl	L'URL dell'immagine dell'offerta obiettivo per l'URL IMG nel messaggio.	No	Nessuno.
offerImageUrlAttr	Il nome dell'attributo offerta che ha l'URL dell'immagine dell'offerta obiettivo	No	Il valore dell'elemento imageUrlAttribute trovato nella sezione defaultParameterValues del file MessageConnectorConfig.xml.
offerLandingPageUrl	L'URL della pagina di arrivo corrispondente all'offerta obiettivo.	No	Nessuno.

Nome parametro	Descrizione	Obbligatorio?	Valore predefinito
offerLandingPageUrlAttr	Il nome dell'attributo offerta che ha l'URL della pagina di arrivo corrispondente all'offerta obiettivo	No	Il valore dell'elemento landingPageUrlAttribute trovato nella sezione defaultParameterValues del file MessageConnectorConfig.xml.
contactEvent	Il nome dell'evento contatto.	No	Il valore dell'elemento contactEventName trovato nella sezione defaultParameterValues del file MessageConnectorConfig.xml, che per impostazione predefinita è "contact".
responseEvent	Il nome dell'evento accettazione.	No	Il valore dell'elemento acceptEventName trovato nella sezione defaultParameterValues del file MessageConnectorConfig.xml, che per impostazione predefinita è "accept".
debug	L'indicatore di debug. Impostare questo parametro su "true" solo per risolvere i problemi e su istruzione del supporto tecnico IBM .	No	Il valore dell'elemento debugFlag trovato nella sezione defaultParameterValues del file MessageConnectorConfig.xml, che per impostazione predefinita è "false".
<audience id>	L'ID destinatario di questo utente. Il nome di questo parametro è definito nel file di configurazione.	Sì	Nessuno.

Quando il connettore del messaggio riceve un parametro che non viene riconosciuto (ossia, non è presente nell'elenco precedente), viene gestito in uno dei seguenti due modi possibili:

- Se viene fornito un parametro non riconosciuto (ad esempio, "attribute", come in attribute="attrValue") ed esiste un parametro corrispondente con lo stesso nome più la parola "Type" (ad esempio, "attributeType", come in attributeType="string"), il connettore del messaggio crea un parametro Interact corrispondente e lo trasmette al runtime di Interact.

  I valori del parametro Type possono essere i seguenti:

  – string
  – numeric
  – datetime

    Per un parametro di tipo "datetime" il connettore del messaggio cerca anche un parametro con lo stesso nome più la parola "Pattern" (ad esempio, "attributePattern") il cui valore è un formato data/ora valido. Ad esempio, si potrebbe fornire il parametro attributePattern="MM/dd/yyyy".

    Tenere presente che se si specifica un tipo di parametro "datetime" ma non si fornisce un pattern di data corrispondente, viene utilizzato il valore specificato nel file di configurazione del connettore del messaggio (presente in <installation_directory>/msgconnector/config/MessageConnectorConfig.xml) sul server Interact.
- Se viene fornito un parametro non riconosciuto e non vi è un valore Type corrispondente, il connettore del messaggio lo passa all'URL di reindirizzamento di destinazione.

Per tutti i parametri non riconosciuti, il connettore messaggio li passa al server di runtime di Interact senza elaborarli o salvarli.

Esempio di codice del connettore del messaggio

Il seguente tag A contiene un esempio di una serie di link del connettore del messaggio che potrebbero essere visualizzati in un messaggio email:

```
<a href="http://www.example.com/MessageConnector/offerClickthru.jsp?msgId=234
   &linkId=1&userid=1&referral=xyz">
 <img src="http://www.example.com/MessageConnector/offerImage.jsp?msgId=234&linkId=1
      &userid=1&incomeCode=3&incomeType=numeric"/>
</a>
```

In questo esempio, il tag `IMG` si carica automaticamente quando viene aperto il messaggio di posta elettronica. Richiamando l'immagine dalla pagina specificata, il messaggio trasmette i parametri per l'identificatore messaggio univoco (msgID), l'identificatore link univoco (linkID) e l'identificatore utente univoco (userid), insieme a due ulteriori parametri (incomeCode e incomeType) da trasmettere al runtime di Interact.

Il tag `A` fornisce l'attributo HREF (Hypertext Reference) che trasforma l'immagine dell'offerta in un link selezionabile nel messaggio di posta elettronica. Se chi visualizza il messaggio, vedendo l'immagine, esegue il click-through alla pagina di arrivo, l'identificativo messaggio univoco (`msgId`), l'identificativo link (`linkId`) e l'identificativo utente (`userid`) vengono trasmessi al server, insieme ad un parametro aggiuntivo (`referral`) che viene trasmesso all'URL di reindirizzamento di destinazione.

Informazioni su Interact Web Connector

Interact WebConnector (anche detto JavaScript Connector, o JSConnector) fornisce un servizio sul server di runtime Interact che consente al codice JavaScript di richiamare l'API Interact Java. Questo abilita le pagine web ad effettuare chiamate a Interact per la personalizzazione dell'offerta in tempo reale utilizzando solo il codice JavaScript integrato, senza dover utilizzare i linguaggi di sviluppo web (come Java, PHP, JSP e così via). Ad esempio, è possibile integrare un piccolo frammento di codice JavaScript su ogni pagina del sito Web che fornisca le offerte consigliate da Interact, in modo che ogni volta che la pagina viene caricata, vengono effettuate chiamate all'API di Interact per garantire che vengano visualizzate le migliori offerte nella pagina di caricamento per il visitatore del sito.

Utilizzare Interact Web Connector in situazioni in cui si desidera visualizzare le offerte ai visitatori su una pagina quando non si dispone del controllo programmatico lato server sulla visualizzazione della pagina (come succederebbe con, ad esempio, PHP o altro linguaggio di script basato sul server), ma è comunque possibile integrare codice JavaScript nel contenuto della pagina che verrà eseguito dal browser Web del visitatore.

Suggerimento: i file di Interact Web Connector sono installati automaticamente nel server di runtime Interact, nella directory `<Interact_home>/jsconnector`. Nella directory `<Interact_home>/jsconnector`, è possibile trovare un file ReadMe.txt contenente informazioni importanti e i dettagli sulle funzioni di Web Connector, così come file di esempio e il codice di origine di Web Connector da utilizzare come base per lo sviluppo delle proprie soluzioni. Se le informazioni presenti qui non sono soddisfacenti, andare alla directory `jsconnector` per ulteriori informazioni.

Installazione di Web Connector sul server di runtime

Un'istanza di Web Connector è installata automaticamente con il server di runtime IBM Interact ed è abilitata per impostazione predefinita. Tuttavia, ci sono alcune impostazioni che è necessario modificare prima di poter configurare e utilizzare Web Connector.

Informazioni su questa attività

Le impostazioni che è necessario modificare prima di poter utilizzare il Web Connector che è installato sul server di runtime vengono aggiunte alla configurazione del server delle applicazioni Web. Per questo motivo, sarà necessario riavviare il server delle applicazioni Web dopo il completamento di questi step.

Procedura

1. Per il server delle applicazioni Web su cui è installato il server di runtime Interact, impostare le seguenti proprietà Java:

   `-DUI_JSCONNECTOR_ENABLE_INPROCESS=true`

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Sostituire `<jsconnectorHome>` con il percorso per la directory `jsconnector` sul server di runtime, che è `<Interact_Home>/jsconnector`.

   Il modo in cui impostare le proprietà Java dipende dal server delle applicazioni Web in uso. Ad esempio, in WebLogic, occorre modificare il file `startWebLogic.sh` o `startWebLogic.cmd` per aggiornare l'impostazione `JAVA_OPTIONS`, come in questo esempio:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
   jsconnector"
   ```

   In WebSphere Application Server, occorre impostare questa proprietà nel pannello JVM (Java virtual machine) della console di gestione.

   Consultare la documentazione del server delle applicazioni Web in uso per istruzioni specifiche sull'impostazione delle proprietà Java.
2. Riavviare il server delle applicazioni Web, se era già in esecuzione, o avviarlo ora per assicurarsi che vengano utilizzate le nuove proprietà Java.

Risultati

Quando il server delle applicazioni Web ha completato il suo processo di avvio, l'installazione di Web Connector sul server di runtime è terminata. Lo step successivo consiste nel collegarsi alla relativa pagina Web di configurazione all'indirizzo `http://<host>:<porta>/interact/jsp/WebConnector.jsp`, dove `<host>` è il nome del server di runtime Interact e `<porta>` è la porta su cui Web Connector è in ascolto, come specificato dal server delle applicazioni Web.

Installazione di Web Connector come un'applicazione web separata

Un'istanza di Web Connector è installata automaticamente con il server di runtime IBM Interact ed è abilitata per impostazione predefinita. Tuttavia, è anche possibile distribuire il Web Connector come applicazione Web a sé stante (ad esempio, in un server delle applicazioni Web su un sistema separato) e configurarlo per comunicare con il server di runtime Interact remoto.

Informazioni su questa attività

Queste istruzioni descrivono il processo di distribuzione del Web Connector come applicazione Web separata con accesso a un server di runtime Interact remoto.

Prima di poter distribuire il Web Connector, è necessario avere installato il server di runtime IBM Interact e avere un server delle applicazioni Web su un altro

sistema con accesso di rete (non bloccata da un firewall) al server di runtime Interact.

Procedura

1. Copiare la directory `jsconnector` che contiene i file di Web Connector dal server di runtime Interact al sistema in cui il server delle applicazioni Web (ad esempio WebSphere Application Server) è già configurato e in esecuzione. È possibile trovare la directory `jsconnector` nella directory di installazione di Interact.
2. Sul sistema in cui verrà distribuita l'istanza del Web Connector, configurare il file `jsconnector/jsconnector.xml` utilizzando un qualsiasi editor di testo o XML per modificare l'attributo `interactURL`.

   L'impostazione predefinita è `http://localhost:7001/interact`, ma è necessario modificarla in modo che corrisponda all'URL del server di runtime Interact remoto, ad esempio `http://runtime.example.com:7011/interact`.

   Dopo aver distribuito il Web Connector, è possibile utilizzare un'interfaccia Web per personalizzare le rimanenti impostazioni nel file `jsconnector.xml`. Consultare “Configurazione di Web Connector” a pagina 323 per ulteriori informazioni.
3. Per il server delle applicazioni Web su cui verrà distribuito il Web Connector, impostare la seguente proprietà Java:

   `-DUI_JSCONNECTOR_HOME=<jsconnectorHome>`

   Sostituire *<jsconnectorHome>* con il percorso reale all'ubicazione in cui è stata copiata la directory `jsconnector` sul server delle applicazioni Web.

   Il modo in cui impostare le proprietà Java dipende dal server delle applicazioni Web in uso. Ad esempio, in WebLogic, occorre modificare il file `startWebLogic.sh` o `startWebLogic.cmd` per aggiornare l'impostazione `JAVA_OPTIONS`, come in questo esempio:

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/InteractFiles/
   jsconnector"
   ```

   In WebSphere Application Server, occorre impostare questa proprietà nel pannello JVM (Java virtual machine) della console di gestione.

   Consultare la documentazione del server delle applicazioni Web in uso per istruzioni specifiche sull'impostazione delle proprietà Java.
4. Riavviare il server delle applicazioni Web, se era già in esecuzione, o avviarlo in questo step, per assicurarsi che venga utilizzata le nuove proprietà Java.

   Attendere che il server delle applicazioni web completi il processo di avvio prima di continuare.
5. Connettersi all'interfaccia di gestione del server delle applicazioni web con i privilegi necessari per distribuire un'applicazione.
6. Seguire le istruzioni per fare in modo che il server delle applicazioni Web distribuisca ed esegua il seguente file: `jsConnector/jsConnector.war`

Risultati

Il Web Connector ora è distribuito nell'applicazione Web. Quando il server Interact sarà completamente configurato e in esecuzione, collegarsi alla pagina Web di configurazione del Web Connector all'indirizzo `http://` *<host>*`:` *<porta>*`/interact/jsp/WebConnector.jsp`, dove *<host>* è il sistema su cui è in esecuzione il server delle applicazioni Web in cui è stato appena distribuito il Web Connector e *<porta>* è la porta su cui Web Connector è in ascolto, come specificato dal server delle applicazioni Web.

Configurazione di Web Connector

Le impostazioni di configurazione per Interact Web Connector sono memorizzate in un file denominato `jsconnector.xml` che è memorizzato sul sistema in cui è distribuito il Web Connector (come il server di runtime Interact stesso o un sistema separato in esecuzione su un server di applicazioni Web). È possibile modificare il file `jsconnector.xml` direttamente, mediante un editor di testo o editor XML; tuttavia, un modo più semplice per configurare quasi tutte le impostazioni di configurazione disponibili consiste nell'utilizzare la pagina di configurazione di Web Connector dal browser Web.

Prima di iniziare

Prima di poter utilizzare l'interfaccia Web per configurare il Web Connector, è necessario installare e distribuire l'applicazione Web che fornisce il Web Connector. Sul server di runtime Interact, un'istanza di Web Connector è installata automaticamente quando si installa e si distribuisce Interact. Su ogni altro server delle applicazioni Web, è necessario installare e distribuire l'applicazione Web Web Connector come descritto in "Installazione di Web Connector come un'applicazione web separata" a pagina 321.

Procedura

1. Aprire il browser Web supportato e aprire un URL simile al seguente:

   `http://<host>:<porta>/interact/jsp/WebConnector.jsp`

   - Sostituire <*host*> con il server su cui Web Connector è in esecuzione, come ad esempio il nome host del server di runtime o il nome del server su cui è distribuita un'istanza separata di Web Connector.
   - Sostituire <*porta*> con il numero di porta su cui l'applicazione Web Web Connector è in ascolto delle connessioni, che di solito corrisponde alla porta predefinita del server delle applicazioni Web.
2. Nella pagina Configurazioni che viene visualizzata, completare le seguenti sezioni:

Tabella 31. Riepilogo delle impostazioni di configurazione di Web Connector.

Sezione	Impostazioni
Impostazioni di base	Utilizzare la pagina Impostazioni di base per configurare il comportamento generale di Web Connector per il sito su cui verranno visualizzate le pagine con tag. Queste impostazioni includono l'URL di base per il sito, le informazioni che Interact deve utilizzare relativamente ai visitatori del sito e impostazioni simili che si applicano a tutte le pagine per cui si prevede di applicare tag con codice Web Connector. Per ulteriori dettagli consultare "Configurazione WebConnector - Opzioni di base" a pagina 325.

Tabella 31. Riepilogo delle impostazioni di configurazione di Web Connector (Continua).

Sezione	Impostazioni
Tipi di visualizzazione HTML	Utilizzare la pagina Tipi di visualizzazione HTML per determinare il codice HTML che verrà fornito per ciascun punto di interazione sulla pagina. È possibile scegliere dall'elenco di modelli predefiniti (file .flt) che contengono una combinazione di codice CSS (Cascading Style Sheet), codice HTML e codice Javascript da utilizzare per ciascun punto di interazione. È possibile utilizzare i modelli così come vengono forniti, personalizzarli come necessario oppure crearne di propri. le impostazioni di configurazione su questa pagina corrispondono alla sezione `interactionPoints` del file di configurazione `jsconnector.xml`. Per ulteriori dettagli consultare "Configurazione WebConnector - Tipi di visualizzazione HTML" a pagina 327.
Pagine avanzate	Utilizzare la pagina Pagine avanzate per associazione le impostazioni specifiche della pagina a un pattern di URL. Ad esempio, è possibile impostare un mapping di pagina tale che ogni URL contenente il testo "index.htm" visualizzi la pagina di benvenuto generica, con punti di interazione ed eventi di caricamento pagina specifici definiti per tale mapping. le impostazioni di configurazione su questa pagina corrispondono alla sezione `pageMapping` del file di configurazione `jsconnector.xml`. Per ulteriori dettagli consultare "Configurazione WebConnector - Pagine avanzate" a pagina 330.

3. Nella pagina Impostazioni di base, verificare che le impostazioni a livello di sito siano valide per la propria installazione e, facoltativamente, specificare la modalità di debug (non consigliato a meno che non si stia tentando di risolvere un problema), l'integrazione tag di pagina Digital Analytics for On Premises e le impostazioni predefinite per la maggior parte dei punti di interazione, quindi fare clic sul link Tipi di visualizzazione HTML sotto Configurazioni.
4. Nella pagina Tipi di visualizzazione HTML, attenersi alla seguente procedura per aggiungere o modificare i modelli di visualizzazione che definiscono i punti di interazione sulla pagina web del cliente.

   Per impostazione predefinita, i modelli di visualizzazione (file .flt) vengono memorizzati in `<jsconnector_home>/conf/html`.

   a. Selezionare il file .flt dall'elenco che si desidera esaminare o utilizzare come punto di partenza o fare clic su Aggiungi un tipo per creare un nuovo modello di punto di interazione vuoto da utilizzare.

      Le informazioni sul contenuto del modello, se presenti, vengono visualizzate accanto all'elenco di modelli.

   b. Facoltativamente, modificare il nome del modello nel campo **Nome file per questo tipo di visualizzazione**. Per un nuovo modello, modificare `CHANGE_ME.flt` con una indicazione più significativa.

      Se si ridenomina il modello qui, il Web Connector crea un nuovo file con tale nome la volta successiva che il modello viene salvato. I modelli vengono salvati quando si modifica il corpo del testo e successivamente si passa a un qualsiasi altro campo.

   c. Modificare o completare le informazioni Frammento HTML come necessario, compreso qualsiasi codice foglio di stile (CSS), JavaScript e HTML che si desidera includere. Tenere presente che è anche possibile includere variabili che verranno sostituite dai parametri Interact durante il

runtime. Ad esempio, `${offer.HighlightTitle}` viene sostituito automaticamente dal titolo dell'offerta nell'ubicazione specificata del punto di interazione.

Utilizzare gli esempi che appaiono sotto il campo Frammento HTML per indicazioni su come formattare i blocchi di codice CSS, JavaScript o HTML.

5. Utilizzare la pagina Pagine avanzate come necessario per impostare i mapping di pagina che determinano il modo in cui vengono gestiti i pattern URL specifici sulle pagine.
6. Dopo aver terminato di impostare le proprietà di configurazione, fare clic su **Diffondi le modifiche**. Facendo clic su **Diffondi le modifiche** si verificano le seguenti azioni:
   - Si visualizza la tag di pagina IBM Interact Web Connector che contiene il codice JavaScript che è possibile copiare dal Web Connector e inserire nelle pagine Web.
   - Si esegue il backup dei file di configurazione del Web Connector esistenti sul server Interact (il file `jsconnector.xml` sul server in cui è installato Web Connector) e si crea un nuovo file di configurazione con le impostazioni definite.

     I file di configurazione di backup vengono memorizzati in `<jsconnector_home>/conf/archive/jsconnector.xml.<date>.<time>`, come in `jsconnector.xml.20111113.214933.750-0500` (dove la stringa date è 20111113 e la stringa time, incluso l'indicatore del fuso orario, è 214933.750-0500)

Risultati

La configurazione del Web Connector ora è completa.

Per modificare la configurazione, è possibile ritornare all'inizio di questa procedura ed eseguirla nuovamente con nuovi valori oppure è possibile aprire il file di configurazione (`<Interact_home>/jsconnector/conf/jsconnector.xml`) in qualsiasi editor di testo o XML e modificarlo come necessario.

Configurazione WebConnector - Opzioni di base

Utilizzare la pagina Impostazioni di base della pagina Configurazioni di Web Connector per configurare il comportamento generale di Web Connector per il sito su cui verranno visualizzate le pagine con tag. Queste impostazioni includono l'URL di base per il sito, le informazioni che Interact deve utilizzare relativamente ai visitatori del sito e impostazioni simili che si applicano a tutte le pagine per cui si prevede di applicare tag con codice Web Connector.

Impostazioni per l'intero sito

Le opzioni di configurazione Impostazioni per l'intero sito, sono impostazioni globali che influenzano il comportamento generale dell'installazione del Web Connector che si sta configurando. È possibile specificare i seguenti valori:

Tabella 32. Impostazioni per l'intero sito per l'installazione di Web Connector

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
URL API Interact	L'URL di base del server di runtime Interact. **Nota:** questa impostazione viene utilizzata solo se Web Connector non è in esecuzione all'interno del server di runtime Interact (cioè, è stato distribuito separatamente).	*<interactURL>*
URL Web Connector	L'URL di base utilizzato per generare l'URL click-through.	*<jsConnectorURL>*
Nome canale interattivo per il sito web di destinazione	Il nome del canale interattivo definito sul server Interact che rappresenta questo mapping di pagina.	*<interactiveChannel>*
Livello destinatario dei visitatori	Il livello destinatario di Campaign per il visitatore in entrata; utilizzato nella chiamata API al runtime Interact.	*<audienceLevel>*
Nome campo ID del destinatario nella tabella profili	Il nome del campo audienceId che verrà utilizzato nella chiamata API a Interact. Notare che attualmente non vi è alcun supporto per gli identificativi destinatario multi-campo.	*<audienceIdField>*
Tipo di dati del campo ID del destinatario	Il tipo di dati del campo ID del destinatario ("numeric" o "string") da utilizzare nella chiamata API a Interact.	*<audienceIdFieldType>*
Nome del cookie che rappresenta l'ID sessione	Il nome del cookie che conterrà l'ID sessione.	*<sessionIdCookie>*
Nome del cookie che rappresenta l'ID visitatore	Il nome del cookie che conterrà l'ID visitatore.	*<visitorIdCookie>*

Funzioni facoltative

Le opzioni di configurazione Funzioni facoltative, sono impostazioni globali per l'installazione del Web Connector che si sta configurando. È possibile specificare i seguenti valori:

Tabella 33. Impostazioni facoltative per l'intero sito per l'installazione di Web Connector

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
Abilita modalità di debug	Specifica (con una risposta `sì` o `non`) se utilizzare o meno una modalità di debug speciale. Se si abilita questa funzione, il contenuto restituito da Web Connector include una chiamata Javascript a 'alert' che informa il client che il mapping di pagina specifico è appena avvenuto. Il client deve avere una voce nel file specificato dall'impostazione <*authorizedDebugClients*> per poter richiamare l'avviso.	<*enableDebugMode*>
File host dei client di debug autorizzati	Il percorso al file che contiene l'elenco di host o indirizzi IP (Internet Protocol) che qualificano per la modalità di debug. Il nome host o l'indirizzo IP del client deve essere presente nel file specificato perché le informazioni di debug vengano raccolte.	<*authorizedDebugClients*>
Abilita integrazione tag di pagina Digital Analytics for On Premises	Specifica (con una risposta `sì` o `non`) se il Web Connector deve collegare la tag IBM Digital Analytics for On Premises specificata alla fine del contenuto della pagina.	`<enableNetInsightTagging>`
File modello HTML tag Digital Analytics for On Premises	Il modello HTML/Javascript utilizzato per integrare una chiamata alla tag Digital Analytics for On Premises. In generale, si consiglia di accettare l'impostazione predefinita a meno che non venga richiesto di fornire un modello diverso.	`<netInsightTag>`

Configurazione WebConnector - Tipi di visualizzazione HTML

Utilizzare la pagina Tipi di visualizzazione HTML per determinare il codice HTML che verrà fornito per ciascun punto di interazione sulla pagina. È possibile scegliere dall'elenco di modelli predefiniti (file .flt) che contengono una combinazione di codice CSS (Cascading Style Sheet), codice HTML e codice JavaScript da utilizzare per ciascun punto di interazione. È possibile utilizzare i modelli così come vengono forniti, personalizzarli come necessario oppure crearne di propri.

Nota: le impostazioni di configurazione su questa pagina corrispondono alla sezione `interactionPoints` del file di configurazione `jsconnector.xml`.

Il punto di interazione può anche contenere segnaposti (zone) in cui possono essere rilasciati automaticamente gli attributi dell'offerta. Ad esempio, è possibile includere `${offer.TREATMENT_CODE}` che verrebbe sostituito con il codice trattamento assegnato a tale offerta durante l'interazione.

I modelli che vengono visualizzati in questa pagina sono caricati automaticamente dai file memorizzati nella directory `<Interact_home>/jsconnector/conf/html` del server Web Connector. Anche tutti i nuovi modelli creati qui vengono memorizzati in tale directory.

Per utilizzare la pagina Tipi di visualizzazione HTML per visualizzare o modificare qualsiasi modello esistente, selezionare il file `.flt` dall'elenco.

Per creare un nuovo modello sulla pagina Tipi di visualizzazione HTML, fare clic su **Aggiungi un tipo**.

Indipendentemente dal metodo che si sceglie per creare o modificare un modello, vengono visualizzate le seguenti informazioni accanto all'elenco di modelli:

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
Nome per questo tipo di visualizzazione	Il nome assegnato al modello che si sta modificando. Questo nome deve essere valido per il sistema operativo su cui è in esecuzione il Web Connector; ad esempio, non è possibile utilizzare una barra (/) nel nome se il sistema operativo è Microsoft Windows. Se si sta creando un nuovo modello, questo campo è preimpostato su `CHANGE_ME.flt`. È necessario modificare questa impostazione su un valore significativo prima di continuare.	*<htmlSnippet>*

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
Frammento HTML	Il contenuto specifico che il Web Connector deve inserire nel punto di interazione sulla pagina web. Questo frammento può contenere codice HTML, informazioni di formattazione CSS o JavaScript da eseguire sulla pagina. Ognuno di questi tre tipi di contenuto deve essere racchiuso tra i codici BEGIN e END, come nei seguenti esempi: • ${BEGIN_HTML} <*contenuto HTML*> ${END_HTML} • ${BEGIN_CSS} <*informazioni foglio di stile specifico del punto di interazione*> ${END_CSS} • ${BEGIN_JAVASCRIPT} <*codice JavaScript specifico del punto di interazione*> ${END_JAVASCRIPT} È anche possibile immettere un numero di codici speciali predefiniti che vengono sostituiti automaticamente quando la pagina viene caricata, incluso i seguenti: • `${logAsAccept}`: una macro che prende due parametri (un URL di destinazione e il TreatmentCode utilizzato per identificare l'accettazione dell'offerta) e li sostituisce con l'URL click-through. • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}` • `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` Ciascuno dei codici offerta qui elencati rappresenta gli attributi dell'offerta definiti nel modello dell'offerta in IBM Campaign utilizzati dal marketer per creare le offerte che Interact sta restituendo. Si noti che il Web Connector utilizza un motore di modelli denominato FreeMarker che fornisce numerose opzioni aggiuntive utili nell'impostazione di codici sui propri modelli di pagina. Consultare http://freemarker.org/docs/index.html per ulteriori informazioni.	Nessun equivalente perché il frammento HTML si trova in un proprio file separato da jsconnector.xml.

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
Codici speciali di esempio	Contiene esempi del tipo di codici speciali, inclusi i codici che identificano i blocchi come HTML, CSS o javascript, e zone rilasciabili che è possibile inserire per fare riferimento a specifici metadati dell'offerta.	Nessun equivalente.

Le modifiche apportate a questa pagina vengono salvate automaticamente quando si passa ad un'altra pagina di configurazione di Web Connector.

Configurazione WebConnector - Pagine avanzate

Utilizzare la pagina Pagine avanzate per associazione le impostazioni specifiche della pagina a un pattern di URL. Ad esempio, è possibile impostare un mapping di pagina tale che ogni URL in entrata contenente il testo "index.htm" visualizzi la pagina di benvenuto generica, con punti di interazione ed eventi di caricamento pagina specifici definiti per tale mapping.

Nota: le impostazioni di configurazione su questa pagina corrispondono alla sezione `pageMapping` del file di configurazione `jsconnector.xml`.

Per utilizzare la pagina Pagine avanzate per creare un nuovo mapping di pagina, fare clic sul link **Aggiungi una pagina** e completare le informazioni necessarie per il mapping.

Info pagina

Le opzioni di configurazione delle informazioni di pagina per il mapping di pagina, definiscono il pattern dell'URL che agisce come trigger per questo mapping e alcune altre impostazioni per la modalità di gestione di questo mapping di pagina da parte di Interact.

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
URL contiene	Si tratta del pattern dell'URL che il Web Connector deve cercare nella richiesta di pagina in entrata. Ad esempio, se l'URL di richiesta contiene "ipoteca.htm" è possibile metterlo in corrispondenza con la pagina di informazioni sull'ipoteca.	*<urlPattern>*
Nome descrittivo per questa pagina o serie di pagine	Un nome significativo per proprio riferimento che descrive l'obiettivo di questo mapping di pagina, ad esempio "Pagina informazioni ipoteca".	*<friendlyName>*
Restituisci offerte anche come dati JSON per utilizzo JavaScript	Un elenco a discesa per indicare se si desidera che il Web Connector includa o meno i dati offerta non elaborati in formato JSON (JavaScript Object Notation) (http://www.json.org/) alla fine del contenuto della pagina.	*<enableRawDataReturn>*

Eventi da attivare (caricare) quando una visita viene effettuata su questa pagina o serie di pagine

Questa serie di opzioni di configurazione per il mapping di pagina definisce il pattern dell'URL che agisce come trigger per questo mapping e alcune altre impostazioni per la modalità di gestione di questo mapping di pagina da parte di Interact.

Nota: le impostazioni di configurazione in questa sezione corrispondono alla sezione `<pageLoadEvents>` del file `jsconnector.xml`.

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
Eventi singoli	Un elenco di eventi disponibili per questa pagina o serie di pagine. Gli eventi in questo elenco sono quelli che sono stati definiti in Interact, selezionare uno o più eventi che si desidera si verifichino quando la pagina viene caricata. La sequenza di chiamate API Interact è la seguente: 1. `startSession` 2. `postEvent` per ogni singolo evento di caricamento pagina (premesso che siano stati definiti singoli eventi in Interact) 3. Per ciascun punto di interazione: • `getOffers` • `postEvent(ContactEvent)`	*<event>*

Punti di interazione (ubicazioni di visualizzazione offerta) su questa pagina o serie di pagine

Queste serie di opzioni di configurazione per il mapping di pagina consente di selezionare quali punti di interazione vengono visualizzati nella pagina Interact.

Nota: le impostazioni di configurazione in questa sezione corrispondono alla sezione `<pageMapping>` | `<page>` | `<interactionPoints>` del file `jsconnector.xml`.

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
Casella di spunta nome punto di interazione	Ogni punto di interazione che è stato definito nel file di configurazione viene visualizzato in questa sezione della pagina. Selezionando la casella di spunta accanto al nome del punto di interazione viene visualizzato un numero di opzioni disponibili per quel punto di interazione.	*<interactionPoint>*

Impostazione	Descrizione	Impostazione equivalente in jsconnector.xml
ID elemento HTML (Interact imposterà l'HTML interno)	Il nome dell'elemento HTML che deve ricevere il contenuto per questo punto di interazione. Ad esempio, se si specifica `<div id="welcomebanner">` nella pagina, immettere `welcomebanner` (il valore ID) in questo campo.	*<htmlElementId>*
Tipo di visualizzazione HTML	Un elenco a discesa che consente di selezionare il tipo di visualizzazione HTML (i frammenti HTML o i file .flt definiti su una precedente pagina di configurazione di Web Connector) da utilizzare per questo punto di interazione.	*<htmlSnippet>*
Numero massimo di offerte da presentare (in caso di sequenza o flipbook)	Il numero massimo di offerte che il Web Connector deve richiamare dal server Interact per questo punto di interazione. Questo campo è facoltativo e si applica solo per un punto di interazione che aggiorna regolarmente le offerte presentate senza ricaricare la pagina, come nello scenario in sequenza dove vengono richiamate più offerte in modo che possano essere rese disponibili una alla volta.	*<maxNumberOfOffers>*
Evento da attivare quando viene presentata l'offerta	Il nome dell'evento di contatto da inviare per questo punto di interazione.	*<contactEvent>*
Evento da attivare quando viene accettata l'offerta	Il nome dell'evento di accettazione da inviare per questo punto di interazione nel momento in cui si fa clic sul link offerta.	*<acceptEvent>*
Evento da attivare quando viene rifiutata l'offerta	Il nome dell'evento di rifiuto da inviare per questo punto di interazione. **Nota:** in questo momento, questa funzione non è utilizzata	*<rejectEvent>*

Opzioni di configurazione di Web Connector

In generale, è possibile utilizzare un'interfaccia Web Connector grafica per configurare le impostazioni Web Connector. Tutte le impostazioni specificate sono anche memorizzate in un file denominato jsconnector.xml, che si trova nella directory `jsconnector/conf`. Ciascuno dei parametri salvati nel file di configurazione jsconnector.xml è descritto in questa sezione.

Parametri e relative descrizioni

I seguenti parametri vengono memorizzati nel file `jsconnector.xml` e vengono utilizzati per le interazioni Web Connector. Esistono due modi per modificare queste impostazioni:

- Utilizzando la pagina Web di configurazione di Web Connector che viene resa automaticamente disponibile dopo aver distribuito e avviato l'applicazione Web

Connector. Per utilizzare la pagina Web di configurazione, utilizzare il browser Web per aprire un URL simile al seguente: `http://<host>:<porta>/interact/jsp/WebConnector.jsp`.

Le modifiche apportate alla pagina web di gestione vengono memorizzate nel file `jsconnector.xml` sul server in cui è distribuito il Web Connector.

- Modificare il file `jsconnector.xml` direttamente utilizzando un qualsiasi editor di testo o editor XML. È opportuno disporre delle giuste conoscenze per modificare i valori e le tag XML prima di utilizzare questo metodo.

  Nota: ogni volta che si modifica il file `jsconnector.xml` manualmente, è possibile ricaricare tali impostazioni aprendo la pagina di gestione di Web Connector (all'indirizzo `http://<host>:<porta>/interact/jsp/jsconnector.jsp`) e facendo clic su **Ricarica configurazione**.

La seguente tabella descrive le opzioni di configurazione che è possibile impostare, come appaiono nel file `jsconnector.xml`.

Tabella 34. Opzioni di configurazione di Web Connector

Gruppo di parametri	Parametro	Descrizione
`defaultPageBehavior`		
	`friendlyName`	Un identificativo leggibile per il pattern URL per la visualizzazione sulla pagina di configurazione di Web Connector.
	`interactURL`	L'URL di base del server di runtime Interact. Nota: è necessario impostare questo parametro solo se il servizio Web Connector (jsconnector) è in esecuzione come applicazione Web distribuita. Non è necessario impostare questo parametro se WebConnector viene eseguito automaticamente come parte del server di runtime Interact.
	`jsConnectorURL`	L'URL di base utilizzato per generare l'URL click-through, ad esempio `http://host:porta/jsconnector/clickThru`
	`interactiveChannel`	Il nome del canale interattivo che rappresenta questo mapping di pagina.
	`sessionIdCookie`	Il nome del cookie che contiene l'ID sessione utilizzato nelle chiamate API a Interact.
	`visitorIdCookie`	Il nome del cookie che contiene l'ID del destinatario.
	`audienceLevel`	Il livello destinatario della campagna per il visitatore in entrata, utilizzato nella chiamata API al runtime Interact.
	`audienceIdField`	Il nome del campo `audienceId` utilizzato nella chiamata API al runtime Interact. **Nota:** Nota: attualmente non vi è alcun supporto per gli identificativi destinatario multi-campo.
	`audienceIdFieldType`	Il tipo di dati del campo ID del destinatario [`numeric | string`] utilizzato nella chiamata API al runtime Interact

Tabella 34. Opzioni di configurazione di Web Connector (Continua)

Gruppo di parametri	Parametro	Descrizione
	audienceLevelCookie	Il nome del cookie che contiene il livello destinatario. Facoltativo. Se non si imposta questo parametro, il sistema utilizza quanto definito per audienceLevel.
	relyOnExistingSession	Utilizzato nella chiamata API al runtime Interact. In generale, questo parametro è impostato su "true".
	enableInteractAPIDebug	Utilizzato nella chiamata API al runtime Interact per abilitare l'output di debug nei file di log.
	pageLoadEvents	L'evento che verrà inviato quando questa particolare pagina viene caricata. Specificare uno o più eventi all'interno di questa tag, nel formato simile a <event>event1</event>.
	interactionPointValues	Tutti gli elementi in questa categoria agiscono come valori predefiniti per i valori mancanti nelle categorie specifiche dell'IP.
	interactionPointValuescontactEvent	Il nome predefinito dell'evento di contatto da inviare per questo particolare punto di interazione.
	interactionPointValuesacceptEvent	Il nome predefinito dell'evento di accettazione da inviare per questo particolare punto di interazione.
	interactionPointValuesrejectEvent	Il nome predefinito dell'evento di rifiuto da inviare per questo particolare punto di interazione. (Nota: in questo momento, questa funzione non è utilizzata.)
	interactionPointValueshtmlSnippet	Il nome predefinito del modello HTML da fornire per questo punto di interazione.
	interactionPointValuesmaxNumberOfOffers	Il numero massimo predefinito di offerte da richiamare da Interact per questo punto di interazione.
	interactionPointValueshtmlElementId	Il nome predefinito dell'elemento HTML che riceve il contenuto per questo punto di interazione.
	interactionPoints	Questa categoria contiene la configurazione per ciascun punto di interazione. Per qualsiasi proprietà mancante il sistema si basa su quanto è configurato nella categoria interactionPointValues.
	interactionPointname	Il nome del punto di interazione (o IP-Interaction Point).
	interactionPointcontactEvent	Il nome dell'evento di contatto da inviare per questo particolare punto di interazione.
	interactionPointacceptEvent	Il nome dell'evento di accettazione da inviare per questo particolare punto di interazione.
	interactionPointrejectEvent	Il nome dell'evento di rifiuto da inviare per questo particolare punto di interazione. (Tenere presente che questa funzione non è ancora in uso.)

Tabella 34. Opzioni di configurazione di Web Connector (Continua)

Gruppo di parametri	Parametro	Descrizione
	interactionPointhtmlSnippet	Il nome del modello HTML da fornire per questo punto di interazione.
	interactionPointmaxNumberOfOffers	Il numero massimo di offerte da richiamare da Interact per questo punto di interazione.
	interactionPointhtmlElementId	Il nome dell'elemento HTML che riceve il contenuto per questo punto di interazione.
	enableDebugMode	L'indicatore booleano (valori accettabili: true o false) per attivare la modalità debug speciale. Se si imposta questo parametro su true, il contenuto restituito da Web Connector include una chiamata JavaScript a 'alert' che informa il client che il mapping di pagina specifico è appena avvenuto. Il client deve avere una voce nel file authorizedDebugClients per generare l'avviso.
	authorizedDebugClients	Un file utilizzato dalla modalità di debug speciale che contiene l'elenco dei nomi host o degli indirizzi IP (Internet Protocol) che qualificano per la modalità di debug.
	enableRawDataReturn	Un indicatore booleano (valori accettabili: true o false) per determinare se il Web Connector collega i dati dell'offerta non elaborati in formato JSON all'estremità finale del contenuto.
	enableNetInsightTagging	Un indicatore booleano (valori accettabili: true o false) per determinare se il Web Connector collega una tag Digital Analytics for On Premises alla fine del contenuto.
	apiSequence	Rappresenta una implementazione dell'interfaccia APISequence, che indica la sequenza delle chiamate API effettuate dal Web Connector quando viene richiamato un pageTag. Per impostazione predefinita, l'implementazione utilizza una sequenza di StartSession, pageLoadEvents, getOffers e logContact, dove gli ultimi due sono specifici per ciascun punto di interazione.
	clickThruApiSequence	Rappresenta una implementazione dell'interfaccia APISequence, che indica la sequenza delle chiamate API effettuate dal Web Connector quando viene richiamato un clickThru. Per impostazione predefinita, l'implementazione utilizza una sequenza di StartSession e logAccept.
	netInsightTag	Rappresenta il modello JavaScript e HTML utilizzati per integrare una chiamata alla tag Digital Analytics for On Premises. In generale, non dovrebbe essere necessario modificare questa opzione.

Utilizzo della Pagina di gestione di Web Connector

Il Web Connector include una pagina di gestione che fornisce alcuni strumenti utili per gestire e verificare la configurazione in base a come potrebbe essere utilizzata con specifici pattern di URL. È anche possibile utilizzare la pagina di gestione per ricaricare una configurazione che è stata modificata.

Informazioni sulla pagina di gestione

Utilizzando un browser Web supportato, è possibile aprire `http://host:porta/interact/jsp/jsconnector.jsp`, dove *host:porta* corrisponde al nome host su cui è in esecuzione il Web Connector e alla porta su cui è in ascolto delle connessioni, ad esempio `runtime.example.com:7001`

È possibile utilizzare la pagina di gestione nei seguenti modi:

Tabella 35. Opzioni di Pagina di gestione di Web Connector

Opzione	Scopo
Ricarica configurazione	Fare clic sul link **Ricarica configurazione** per ricaricare tutte le modifiche di configurazione che sono state salvate su disco nella memoria. Ciò è necessario quando sono state apportate modifiche direttamente al file di configurazione jsconnector.xml del Web Connector invece di utilizzare le pagine Web di configurazione.
Visualizza configurazione	Visualizzare la configurazione WebConnector in base al pattern URL immesso nel campo **Visualizzare configurazione**. Quando si immette l'URL di una pagina e si fa clic su **Visualizza configurazione**, il Web Connector restituisce la configurazione che il sistema utilizzerà in base a tale mapping di pattern. Se non viene trovata alcuna corrispondenza, viene restituita la configurazione predefinita. Questo è utile per verificare se per una particolare pagina è stata utilizzata la configurazione corretta.
Esegui tag di pagina	Completando i campi su questa pagina e facendo clic su **Esegui tag di pagina**, il Web Connector restituisce il risultato `pageTag` in base al pattern dell'URL. Ciò simula la chiamata di una tag di pagina. La differenza tra la chiamata di pageTag da questo strumento e l'utilizzo di un sito web reale sta nel fatto che l'utilizzo della Pagina di gestione consente la visualizzazione di eventuali errori o eccezioni. Per un sito Internet reale, le eccezioni non vengono restituite e sono visibili solo nel file di log del Web Connector.

Pagina Web Connector di esempio

Come esempio, è stato incluso un file denominato `WebConnectorTestPageSA.html` con Interact Web Connector (nella directory `<Interact_Home/jsconnector/webapp/html`) che mostra come a molte delle funzioni del Web Connector vengono appplicate tag in una pagina. Per comodità, la pagina di esempio viene mostrata anche qui.

Pagina HTML Web Connector di esempio

```
<?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[
/* ##################################################################
Questa è una pagina di test che contiene il pageTag WebConnector.  Poiché
il nome di questo file ha TestPage incorporato, il WebConnector rileverà
una corrispondenza pattern URL con quello "testpage" nella versione predefinita
di jsconnector.xml - la definizione di configurazione associata a tale
pattern URL "testpage" verrà applicata qui. Ciò significa che in questa pagina
deve essere presente
gli id elemento html corrispondenti agli IP per questo
pattern URL (ad esempio, 'welcomebanner', 'crosssellcarousel' e 'textservicemessage')
################################################################################ */

/* ##################################################################
Questa sezione imposta i cookie per sessionId e visitorId.
Notare che in un sito di produzione reale, ciò è effettuato molto probabilmente
dal componente di accesso. Al fine del test, viene fatto qui; il nome del cookie
deve corrispondere a quanto configurato in jsconnector xml.
################################################################################ */
     function setCookie(c_name,value,expiredays)
     {
       var exdate=new Date();
       exdate.setDate(exdate.getDate()+expiredays);
       document.cookie=c_name+ "=" +escape(value)+
       ((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
     }
     setCookie("SessionID","123");
     setCookie("CustomerID","1");

   /* ######################################################
   Ora impostare gli ID elementi html che corrispondono ai punti di interazione
   ###################################################### */
     document.writeln("<div id='welcomebanner'> This should change, "
  + "otherwise something is wrong <\/div>");
     document.writeln("<div id='crosssellcarousel'> This should change, "
  + "otherwise something is wrong <\/div>");
     document.writeln("<div id='textservicemessage'> This should change, "
  + "otherwise something is wrong <\/div>");
   //]]&gt;
</script><!--
################################################################################
questo è quanto incollato dal file pageTag.txt nella directory conf
dell'installazione WebConnector... var unicaWebConnectorBaseURL deve essere
adattato per conformarsi all'ambiente WebConnector locale
################################################################################
-->
<!-- BEGIN: IBM Interact Web Connector Page Tag -->
<!--
# ****************************************************************************
#  Licensed Materials - Property of IBM
#  IBM Interact
#  (c) Copyright IBM Corporation 2001, 2012.
#  US Government Users Restricted Rights - Use, duplication or disclosure
#  restricted by GSA ADP Schedule Contract with IBM Corp.
# ****************************************************************************
-->
<script language="javascript" type="text/javascript">
//<![CDATA[
   var unicaWebConnectorBaseURL=
        "[CHANGE ME - http://host:port/<jsconnector>/pageTag]";
   var unicaURLData = "ok=Y";
   try {
    unicaURLData += "&url=" + escape(location.href)
   } catch (err) {}
   try {
    unicaURLData += "&title=" + escape(document.title)
   } catch (err) {}
   try {
```

```
 unicaURLData += "&referrer=" + escape(document.referrer)
} catch (err) {}
try {
 unicaURLData += "&cookie=" + escape(document.cookie)
} catch (err) {}
try {
 unicaURLData += "&browser=" + escape(navigator.userAgent)
} catch (err) {}
try {
 unicaURLData += "&screensize=" +
 escape(screen.width + "x" + screen.height)
} catch (err) {}
try {
 if (affiliateSitesForUnicaTag) {
     var unica_asv = "";
     document.write("<style id=\"unica_asht1\" type=\"text/css\"> "
  + "p#unica_ashtp a {border:1px #000000 solid; height:100px "
  + "!important;width:100px "
  + "!important; display:block !important; overflow:hidden "
  + "!important;} p#unica_ashtp a:visited {height:999px !important;"
  + "width:999px !important;} <\/style>");
     var unica_ase = document.getElementById("unica_asht1");
     for (var unica_as in affiliateSitesForUnicaTag) {
      var unica_asArr = affiliateSitesForUnicaTag[unica_as];
      var unica_ashbv = false;
      for (var unica_asIndex = 0; unica_asIndex <
  unica_asArr.length && unica_ashbv == false;
  unica_asIndex++)
{
       var unica_asURL = unica_asArr[unica_asIndex];
       document.write("<p id=\"unica_ashtp\" style=\"position:absolute; "
  + "top:0;left:-10000px;height:20px;width:20px;overflow:hidden; \
margin:0;padding:0;visibility:visible;\"> \
<a href=\"" + unica_asURL + "\">" + unica_as + " <\/a><\/p>");
       var unica_ae = document.getElementById("unica_ashtp").childNodes[0];
       if (unica_ae.currentStyle) {
        if (parseFloat(unica_ae.currentStyle["width"]) > 900)
    unica_ashbv = true
       } else if (window.getComputedStyle) {
        if (parseFloat(document.defaultView.getComputedStyle
    (unica_ae, null).getPropertyValue("width")) > 900)
 unica_ashbv = true
       }
       unica_ae.parentNode.parentNode.removeChild(unica_ae.parentNode)
      }
      if (unica_ashbv == true) {
       unica_asv += (unica_asv == "" ? "" : ";") + unica_as
      }
     }
     unica_ase.parentNode.removeChild(unica_ase);
     unicaURLData += "&affiliates=" + escape(unica_asv)
 }
} catch (err) {}
document.write("<script language='javascript' "
  + " type='text/javascript' src='" + unicaWebConnectorBaseURL + "?"
+ unicaURLData + "'><\/script>");
//]]&gt;
</script>
<style type="text/css">
/*<![CDATA[*/
   .unicainteractoffer {display:none !important;}
   /*]]&gt;*/
</style>
   <title>Sample Interact Web Connector Page</title>
   </head>
   <body>
   <!-- END: IBM Interact Web Connector Page Tag -->
```

```
   <!--
################################################################################
fine dell'aggiunta pageTag
################################################################################
-->
   </body>
</html>
```

Capitolo 16. Interact e l'integrazione Digital Recommendations

IBM Interact può effettuare l'integrazione con IBM Digital Recommendations per fornire raccomandazioni di prodotti basati su Interact. Entrambi i prodotti possono fornire raccomandazioni di prodotti per le offerte ma utilizzando metodi differenti. Digital Recommendations utilizza un comportamento web del visitatore (filtro collaborativo) per creare correlazioni tra i visitatori e le offerte consigliate. Interact si basa sul comportamento passato del cliente, sugli attributi, la cronologia e meno sulle offerte a livello visivo, apprendendo quali offerte corrispondono meglio dal profilo di comportamento del cliente (basandosi su informazioni demografiche ed altre informazioni relative al cliente). I tassi di accettazione delle offerte sono utili per creare un modello predittivo tramite l'apprendimento automatico. Utilizzando il meglio di entrambi i prodotti, Interact può utilizzare un profilo personale per definire le offerte che passeranno un ID categoria a Digital Recommendations e può richiamare i prodotti consigliati in base alla popolarità (la "saggezza delle folle") per la visualizzazione al visitatore come parte delle offerte selezionate. Questo può fornire raccomandazioni migliori per i clienti che comporteranno un numero maggiore di click-through e migliori risultati che non con uno solo dei prodotti.

Le seguenti sezioni descrivono in che modo funziona questa integrazione e come utilizzare l'applicazione di esempio fornita per creare la propria integrazione di offerta personalizzata.

Panoramica sull'integrazione di Interact con Digital Recommendations

Questa sezione descrive in che modo IBM Interact può integrarsi con IBM Digital Recommendations per fornire le raccomandazioni di prodotti basate su Interact, inclusa una descrizione del processo e del meccanismo con il quale avviene l'integrazione.

IBM Interact si integra con IBM Digital Recommendations tramite un'API (application programming interface) REST (Representational state transfer), resa disponibile dall'installazione di Digital Recommendations. Effettuando le chiamate API REST con l'ID categoria appropriato, Interact può richiamare i prodotti consigliati e può includerli nelle informazioni delle offerte visualizzate nella pagina personalizzata che il visitatore sta visualizzando.

Quando un visitatore visualizza l'URL della pagina web (come la pagina JSP di esempio inclusa con l'installazione di Interact), la pagina richiama Interact per estrarre un'offerta. Presupponendo che l'offerta sia stata configurata all'interno di Interact con i parametri corretti, si verificano i seguenti step, nel caso più semplice:

1. La logica della pagina identifica l'ID cliente del visitatore.
2. Viene effettuata una chiamata API a Interact, che trasmette le informazioni richieste per generare un'offerta per quel cliente.
3. L'offerta restituita fornisce la pagina web con almeno tre attributi: l'URL per l'immagine dell'offerta, l'URL della pagina di arrivo quando il cliente esegue il click-through, e l'ID categoria da utilizzare per determinare quali prodotti consigliare.

4. L'ID categoria viene quindi utilizzato per chiamare Digital Recommendations per recuperare i prodotti consigliati. Questa serie di prodotti è in formato JSON (JavaScript Object Notation) ordinata in base ai prodotti più venduti in quella categoria.
5. L'offerta e i prodotti vengono quindi visualizzati nel browser del visitatore.

Questa integrazione è utile per combinare la raccomandazione dell'offerta e le raccomandazioni dei prodotti. Ad esempio, in una pagina web si potrebbero avere due punti di interazione: uno per un'offerta e uno per le raccomandazioni corrispondenti all'offerta. A tale scopo, la pagina web effettua una chiamata a Interact per effettuare una segmentazione in tempo reale per determinare la migliore offerta (ad esempio, il 10% di sconto sui piccoli elettrodomestici). Quando la pagina riceve l'offerta da Interact, quell'offerta contiene l'ID categoria (in questo esempio, per i piccoli elettrodomestici). La pagina quindi trasmette l'ID categoria dei piccoli elettrodomestici a Digital Recommendations utilizzando una chiamata API, e riceve in risposta le raccomandazioni prodotto migliori per quella categoria in base alla popolarità.

Un esempio più semplice potrebbe essere una pagina web che effettua una chiamata a Interact solo per individuare una categoria (ad esempio, posate di pregio) che corrisponde al profilo del cliente. L'ID categoria ricevuto verrebbe poi trasmesso a Digital Recommendations per ottenere le raccomandazioni prodotto sulle posate.

Prerequisiti di integrazione

Prima di poter utilizzare l'integrazione Digital Recommendations - Interact, è necessario assicurarsi che siano soddisfatti i prerequisiti descritti in questa sezione.

Assicurarsi che i seguenti prerequisiti siano soddisfatti:

- L'utente deve avere familiarità con l'utilizzo dell'API di Interact come documentato altrove nella *Guida dell'amministratore* e nella guida in linea.
- L'utente deve avere familiarità con l'API REST Digital Recommendations come descritto nella documentazione per lo sviluppatore Digital Recommendations.
- L'utente deve avere conoscenze di base di HTML, JavaScript, CSS e JSON (JavaScript Object Notation).

  JSON è importante perché l'API REST Digital Recommendations restituisce le informazioni sul prodotto richieste dall'utente come dati in formato JSON.
- L'utente deve avere familiarità con la codifica delle pagine web lato server, poiché l'applicazione di dimostrazione fornita con Interact utilizza JSP (anche se JSP non è richiesto).
- L'utente deve disporre di un account Digital Recommendations valido e dell'elenco di ID categoria che si prevede che Interact richiami le raccomandazioni di prodotti (i prodotti più venduti o più popolari nella categoria specificata).
- L'utente deve disporre del link all'API REST Digital Recommendations (un URL per l'ambiente Digital Recommendations).

  Per un esempio, vedere l'applicazione di esempio inclusa con l'installazione di Interact oppure per ulteriori informazioni vedere il codice di esempio in "Utilizzo del progetto campione di integrazione" a pagina 344.

Configurazione di un'offerta per l'integrazione Digital Recommendations

Prima che una pagina web possa richiamare Digital Analytics Digital Recommendations per recuperare un prodotto consigliato, è necessario configurare l'offerta IBM Interact con le informazioni necessarie da trasmettere a Digital Recommendations.

Informazioni su questa attività

Per configurare un'offerta a collegarsi a Digital Recommendations, assicurarsi prima che siano presenti le seguenti condizioni:

- Assicurarsi che il server di runtime Interact sia configurato e che funzioni correttamente.
- Assicurarsi che il server di runtime possa stabilire una connessione con il server Digital Recommendations, accertandosi anche che il firewall non impedisca lo stabilirsi della connessione web standard in uscita (porta 80).

Per configurare un'offerta per l'integrazione con Digital Recommendations, effettuare gli step riportati di seguito.

Procedura

1. Creare o modificare un'offerta per Interact.

   Per informazioni sulla creazione e la modifica di offerte, consultare *IBM Interact - Guida dell'utente* e la documentazione relativa a IBM Campaign.
2. Oltre alle altre impostazioni nell'offerta, assicurarsi che l'offerta includa i seguenti attributi:
   - L'URL (uniform resource locator) che collega all'immagine dell'offerta.
   - L'URL che collega alla pagina di arrivo dell'offerta.
   - Un ID categoria Digital Recommendations associato a questa offerta.

     È possibile recuperare l'ID categoria manualmente dalla configurazione di Digital Recommendations. Interact non può recuperare i valori degli ID categoria direttamente.

   Nell'applicazione web dimostrativa inclusa con l'installazione di Interact, questi attributi dell'offerta sono denominati `ImageURL`, `ClickThruURL` e `CategoryID`. È possibile utilizzare qualsiasi nome abbia un significato per l'utente, purché l'applicazione web abbia corrispondenza con i valori previsti dall'offerta.

   Ad esempio, si potrebbe definire un'offerta denominata "10PercentOff" che contiene questi attributi, dove l'ID categoria (quello recuperato dalla configurazione di Digital Recommendations) è `PROD1161127`, l'URL del clickthrough dell'offerta è `http://www.example.com/success` e l'URL dell'immagine da visualizzare per l'offerta è `http://localhost:7001/sampleIO/img/10PercentOffer.jpg` (un URL che sia, in questo caso, locale per il server di runtime Interact).
3. Definire le regole di trattamento per un canale interattivo per includere questa offerta, e distribuire il canale interattivo nel solito modo.

Risultati

L'offerta ora è definita con le informazioni richieste per l'integrazione Digital Recommendations. Il restante lavoro per consentire a Digital Recommendations di fornire a Interact i consigli di prodotti viene eseguito configurando le pagine ad effettuare le chiamate API appropriate.

Quando si configura l'applicazione web a servire la pagina integrata ai visitatori, accertarsi che i seguenti file siano inclusi nella directory `WEB-INF/lib`:

- *Interact_Home*`/lib/interact_client.jar`, richiesto per gestire le chiamate dalla pagina web all'API Interact.
- *Interact_Home*`/lib/JSON4J_Apache.jar`, richiesto per gestire i dati restituiti dalla chiamata all'API REST Digital Recommendations, che restituisce dati in formato JSON.

Per ulteriori informazioni su come servire le offerte ai clienti, consultare “Utilizzo del progetto campione di integrazione”.

Utilizzo del progetto campione di integrazione

Ogni installazione di runtime di Interact include un progetto di esempio che dimostra Digital Recommendations - Processo di integrazione di Interact. Il progetto di esempio fornisce una dimostrazione completa, dall'inizio alla fine, di creazione di una pagina web che richiama un'offerta che contiene un ID categoria, che viene trasmesso a Digital Recommendations per richiamare un elenco di prodotti consigliati per la presentazione nei punti di interazione della pagina.

Panoramica

È possibile utilizzare il progetto di esempio incluso così come viene fornito, se si desidera testare il processo di integrazione, oppure utilizzarlo come punto di partenza per sviluppare pagine personalizzate. Il progetto di esempio so trova nel seguente file:

Interact_home`/samples/IntelligentOfferIntegration/MySampleStore.jsp`

Questo file, oltre a contenere un esempio completo funzionante di processo di integrazione, contiene anche commenti esaustivi che spiegano cosa impostare in Interact, cosa personalizzare nel file `.jsp` e come distribuire correttamente la pagina per poterla eseguire con la propria installazione.

MySampleStore.jsp

Per comodità, di seguito è riportato il file MySampleStore.jsp. Questo esempio potrebbe essere stato aggiornato nelle release successive di Interact, pertanto utilizzare il file incluso con l'istallazione come punto di partenza per gli esempi necessari.

```
<!--
# ********************************************************************************
#  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());
     }
%>
```

Capitolo 17. Interact e l'integrazione Digital Data Exchange

Con Digital Data Exchange, il sito web può collegarsi a Interact per fornire un potente motore di esecuzione su tutti i canali che distribuisce le migliori offerte ai migliori canali e si evolve (impara) dal feedback dell'offerta per aumentare costantemente la produttività del marketing.

È possibile utilizzare questo strumento se il team del marketing utilizza Interact per la gestione delle offerte su tutti i canali e si desidera estendere tali offerte intelligenti personalizzat e ai propri siti web.

IBM Digital Data Exchange integra IBM e le soluzioni di marketing di terze parti con analisi dei clienti digitali tramite un'API per la diffusione dei dati in tempo reale e una soluzione di gestione delle tag a livello aziendale.

Senza IBM Digital Data Exchange, i marketer dipendono dall'IT per collegare Interact al sito web e chiamare l'API Interact dalle varie pagine web. Con IBM Digital Data Exchange, i marketer possono ignorare l'IT e andare direttamente in IBM Digital Data Exchange per includere le tag IBM Digital Data Exchange sulle varie pagine web.

Prerequisiti

Prima di poter utilizzare l'integrazione Interact e Digital Data Exchange, è necessario assicurarsi che siano soddisfatti i prerequisiti descritti in questa sezione.

Assicurarsi che i seguenti prerequisiti siano soddisfatti.

- L'utente deve avere familiarità con l'API JavaScript Interact come documentato altrove nella Guida dell'amministratore e nella guida in linea.
- L'utente deve avere familiarità con i gruppi di pagine e tag Digital Data Exchange.
- Si dispone di un account Digital Data Exchange valido.
- Il file `interactapi.js` è pubblicamente ospitato in modo che ci si possa accedere nelle impostazioni **Fornitore**.

Integrazione di IBM Interact con il sito web tramite IBM Digital Data Exchange

Utilizzare questa procedura per integrare Interact con il sito web tramite Digital Data Exchange.

Procedura

1. Specificare l'ubicazione del file `Interactapi.js`.
   a. Spostarsi in **Fornitori** > **Impostazioni fornitore** in Digital Data Exchange.
   b. Selezionare IBM Interact dal menu a discesa **Fornitore**.
   c. In **Percorso libreria**, immettere l'URL alla posizione in cui si trova `Interactapi.js`. Non includere il protocollo (http o https) in questo URL.
   d. In **Percorso al servlet Rest pubblico**, aggiungere il percorso al servlet Rest.
2. Spostarsi in **Gestisci** > **Impostazioni globali** in Digital Data Exchange per specificare il nome oggetto da utilizzare come identificativo pagina in

Identificativo pagina univoco. Ad esempio, è possibile impostare il nome oggetto su `digitalData.pageInstanceID`.

3. Includere il file `eluminate.js` e un identificativo nella pagina web su cui si desidera che Digital Data Exchange inserisca le tag. Fornire a ciascuna pagina web un identificativo univoco in modo che Digital Data Exchange sia in grado di fare una distinzione tra le varie pagine.

   Ad esempio, è possibile aggiungere il seguente script alla home page.

```
<!-- Setting Page Identifier -->
          <script>
                  digitalData={pageInstanceID:"INTERACT_HomePage"};
          </script>

<!-- Including eluminate script -->
         <script type="text/javascript" src="http://libs.
                  coremetrics.com/eluminate.js">
         </script>
         <script type="text/javascript">
                  cmSetClientID("51310000|INTERACTTEST",false,"data.
                  coremetrics.com",document.domain);
         </script>
```

4. In Digital Data Exchange creare le tag, i segmenti di codice, le funzioni e altri elementi che si desidera aggiungere alla pagina web.
5. Creare gruppi di pagine per definire ciò che si desidera inserire in ciascuna pagina.

   Consultare IBM Digital Data Exchange User Guide per ulteriori informazioni.

Tag Interact in Digital Data Exchange

Utilizzare le tag Digital Data Exchange predefinite per definire le variazioni delle tag appropriate alle pagine web dove i dati sono presentati da ubicazioni differenti. Una volta definite, tali tag sono aggiunte all'elenco di tag Interact. Le tag possono non avere campi da definire o non avere campi tag obbligatori e possono essere utilizzate direttamente.

Le seguenti tag Interact sono disponibili in Digital Data Exchange sotto **Tag**.

- Sessione finale
- Ottieni offerte
- Libreria di caricamento
- Invia evento
- Imposta destinatario
- Avvia sessione

Per utilizzare le tag Interact, modificare le tag per definire il campo tag, il metodo, il nome oggetto, il tipo di dati e i l modificatore per ciascuna tag Interact.

Le tag Invia evento, Imposta destinatario e Avvia sessione accettano campi tag personalizzati. Utilizzare l'icona per l'aggiunta di campi tag, quindi fare clic sull'icona Modifica, per definire il parametro personalizzato. Il processo è uguale a quello per la definizione di qualsiasi parametro con l'eccezione che il nome del parametro può essere modificato e deve includere il nome parametro, i due punti e il tipo di dati del parametro. L'ordine dei parametri personalizzati nella tag può essere modificato con le frecce verso l'alto e verso il basso.

Le tag possono anche essere collegate a funzioni JavaScript o oggetti HTML in modo che vengano attivate all'attivazione della funzione o al verificarsi dell'evento oggetto HTML.

Per ulteriori informazioni su come definire, collegare e utilizzare le tag, consultare IBM Digital Data Exchange User Guide.

Per casi di utilizzo dettagliati dell'integrazione Interact e Digital Data Exchange, consultare https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Sessione finale

La tag Sessione finale contrassegna la fine di una sessione web.

I seguenti campi tag sono disponibili per la tag Sessione finale.

Tabella 36. Tag Sessione finale

Campo Tag	Descrizione
*ID sessione	Identifica l'ID sessione.
Nome funzione di richiamata alla riuscita	Definisce il nome della funzione da chiamare quando il metodo sessione finale riesce.
Nome funzione di richiamata all'errore	Definisce il nome della funzione da chiamare quando il metodo sessione finale non riesce.

Qualsiasi **Campo tag** contrassegnato con un * è obbligatorio.

Ottieni offerte

Utilizzare la tag Ottieni offerte per richiedere offerte dal server di runtime.

I seguenti campi tag sono disponibili per la tag Ottieni offerte.

Tabella 37. Tag Ottieni offerte

Campo Tag	Descrizione
*ID sessione	Identifica l'ID sessione.
*Nome punto di interazione	Identifica il nome del punto di interazione a cui questo metodo fa riferimento. Questo nome deve corrispondere esattamente al nome del punto di interazione definito nel canale interattivo.
*Numero richiesto	Identifica il numero di offerte richieste.
Nome funzione di richiamata alla riuscita	Definisce il nome della funzione da chiamare quando il metodo ottieni offerte riesce.
Nome funzione di richiamata all'errore	Definisce il nome della funzione da chiamare quando il metodo ottieni offerte non riesce.

Qualsiasi **Campo tag** contrassegnato con un * è obbligatorio.

La tag Ottieni offerte dovrebbe essere assegnata a un gruppo di pagine il cui contenitore è impostato su `Predefinito`.

Libreria di caricamento

La tag Libreria di caricamento carica la libreria JavaScript Interact nella sezione iniziale della pagina.

La tag Libreria di caricamento non ha parametri. Acquisisce l'ubicazione della libreria da `Percorso libreria` in **Impostazioni fornitore**. Deve essere inclusa in un gruppo di pagine con un contenitore impostato su `Intestazione` ed eseguita su tutte le pagine con tag Interact.

Importante: nessuna delle altre tag funziona se non è inclusa la tag libreria di caricamento. Il file interact.js non viene caricato se non è inclusa questa tag.

Invia evento

Utilizzare la tag Invia evento per eseguire qualsiasi evento definito nel canale interattivo.

I seguenti campi tag sono disponibili per la tag Invia evento.

Tabella 38. Tag Invia evento

Campo Tag	Descrizione
*ID sessione	Identifica l'ID sessione.
*Nome evento	Identifica il nome dell'evento. Il nome dell'evento deve corrispondere al nome dell'evento definito nel canale interattivo. Questo nome non è sensibile al maiuscolo/minuscolo.
Nome funzione di richiamata alla riuscita	Definisce il nome della funzione da chiamare quando il metodo invia evento riesce.
Nome funzione di richiamata all'errore	Definisce il nome della funzione da chiamare quando il metodo invia evento non riesce.

Qualsiasi **Campo tag** contrassegnato con un * è obbligatorio.

È possibile aggiungere parametri facoltativi con la funzione campo tag personalizzato. I nomi tag personalizzati devono consistere del nome parametro, i due punti e il tipo di dati.

Imposta destinatario

Utilizzare la tag Imposta destinatario per impostare ID e livello destinatario di un visitatore.

I seguenti campi tag sono disponibili per la tag Imposta destinatario.

Tabella 39. Tag Imposta destinatario

Campo Tag	Descrizione
*ID sessione	Identifica l'ID sessione.
*ID destinatario	Identifica l'ID del destinatario. I nomi devono corrispondere ai nomi colonna fisica di ogni tabella che contiene l'ID destinatario. L'ID destinatario non può contenere più di 17 cifre. Se un ID del destinatario è più lungo di 17 cifr significative deve essere suddiviso o occorre modificarlo in un tipo stringa.

Tabella 39. Tag Imposta destinatario (Continua)

Campo Tag	Descrizione
*Livello destinatario	Definisce il livello del destinatario.
Nome funzione di richiamata alla riuscita	Definisce il nome della funzione da chiamare quando il metodo imposta destinatario riesce.
Nome funzione di richiamata all'errore	Definisce il nome della funzione da chiamare quando il metodo imposta destinatario non riesce.

Qualsiasi **Campo tag** contrassegnato con un * è obbligatorio.

È possibile aggiungere parametri facoltativi con la funzione campo tag personalizzato. I nomi tag personalizzati devono consistere del nome parametro, i due punti e il tipo di dati.

Avvia sessione

La tag Avvia sessione crea e definisce una sessione web.

I seguenti campi tag sono disponibili per la tag Avvia sessione.

Tabella 40. Tag Avvia sessione

Campo Tag	Descrizione
*ID sessione	Identifica l'ID sessione.
*Canale interattivo	Definisce il nome del canale interattivo a cui questa sessione fa riferimento. Questo nome deve corrispondere esattamente al nome del canale interattivo definito in Campaign.
*ID destinatario	Identifica l'ID del destinatario. I nomi devono corrispondere ai nomi colonna fisica di ogni tabella che contiene l'ID destinatario.
*Livello destinatario	Definisce il livello del destinatario.
*Considera sessione esistente	Definisce se questa sessione utilizza una nuova sessione o una esistente
*Debug	Abilita o disabilita le informazioni di debug.
Nome funzione di richiamata alla riuscita	Definisce il nome della funzione da chiamare quando il metodo avvia sessione riesce.
Nome funzione di richiamata all'errore	Definisce il nome della funzione da chiamare quando il metodo avvia sessione non riesce.

Qualsiasi **Campo tag** contrassegnato con un * è obbligatorio.

È possibile aggiungere parametri facoltativi con la funzione campo tag personalizzato. I nomi tag personalizzati devono consistere del nome parametro, i due punti e il tipo di dati.

La tag Avvia sessione dovrebbe essere assegnata a un gruppo di pagine il cui contenitore è impostato su `Predefinito`.

Esempio di impostazioni di tag

Questo esempio mostra una configurazione semplice delle impostazioni delle tag Avvia sessione, Invia evento, Ottieni offerte e Sessione finale.

Per qualsiasi tag è possibile ottenere i valori campo tag dal cookie con il metodo cookie o dall'oggetto JavaScript con il metodo javascriptobject.

Queste tag supportano ulteriori parametri che questo breve esempio non indica. È possibile trovare ulteriori informazioni sui parametri aggiuntivi in IBM Digital Data Exchange User Guide.

Per casi di utilizzo dettagliati dell'integrazione Interact e Digital Data Exchange, consultare https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20and%20IBM%20Digital%20Data%20Exchange%20Integration.

Esempio di impostazioni della tag Avvia sessione

Fare clic su **Tag** > **Tag IBM** > **IBM Interact** > **Tipo: Avvia sessione** per creare una tag Avvia sessione. Modificare la tag con le seguenti impostazioni.

Impostazioni ID sessione

- **Metodo**: `Constant`
- **Costante**: `5555`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni Canale interattivo

- **Metodo**: `Constant`
- **Costante**: `WSCDemo`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni ID destinatario

- **Metodo**: `Constant`
- **Costante**: `USERS_ID,2002,numeric`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni Livello destinatario

- **Metodo**: `Constant`
- **Costante**: `WSCUserId`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni Considera sessione esistente

- **Metodo**: `Constant`
- **Costante**: `False`
- **Tipo dati**: `Boolean`
- **Modificatore**: `<null>`

Debug

- **Metodo**: `Constant`
- **Costante**: `True`

- **Tipo dati**: Boolean
- **Modificatore**: <null>

Impostazioni Nome funzione di richiamata alla riuscita
- **Metodo**: Unassigned
- **Valore**: <null>

Impostazioni Nome funzione di richiamata all'errore
- **Metodo**: Unassigned
- **Valore**:<null>

Esempio di impostazioni della tag Ottieni offerte

Fare clic su **Tag** > **Tag IBM** > **IBM Interact** > **Tipo: Ottieni offerte** per creare una tag Ottieni offerte. Modificare la tag con le seguenti impostazioni.

Impostazioni ID sessione
- **Metodo**: Constant
- **Costante**: 5555
- **Tipo dati**: String
- **Modificatore**: <null>

Impostazioni Nome punto interazione
- **Metodo**: Constant
- **Costante**: AuroraHomepageHeaderBannerLeft
- **Tipo dati**: String
- **Modificatore**: <null>

Impostazioni Numero richiesto
- **Metodo**: Constant
- **Costante**: 1
- **Tipo dati**: integer
- **Modificatore**: <null>

Impostazioni Nome funzione di richiamata alla riuscita
- **Metodo**: Constant
- **Costante**: onOfferReturnSuccess
- **Tipo dati**: string
- **Modificatore**: <null>

Impostazioni Nome funzione di richiamata all'errore
- **Metodo**: Constant
- **Costante**: onOfferReturnError
- **Tipo dati**: string
- **Modificatore**: <null>

Esempio di impostazioni della tag Invia evento

Fare clic su **Tag** > **Tag IBM** > **IBM Interact** > **Tipo: Invia evento** per creare una tag Invia evento. Modificare la tag con le seguenti impostazioni.

Impostazioni ID sessione

- **Metodo**: `Constant`
- **Costante**: `5555`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni Nome evento

- **Metodo**: `Constant`
- **Costante**: `ACCEPTOFFER`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni Nome funzione di richiamata alla riuscita

- **Metodo**: `Constant`
- **Costante**: `onSuccessTestFunction`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni Nome funzione di richiamata all'errore

- **Metodo**: `Constant`
- **Costante**: `onErrorTestFunction`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni di ulteriori campi parametro

- **Campo Tag**: `UACIOfferTrackingCode:string`
- **Metodo**: `JavaScriptObject`
- **Nome oggetto**: `oa.treatmentCode`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Esempio di impostazioni della tag Sessione finale

Fare clic su **Tag** > **Tag IBM** > **IBM Interact** > **Tipo: Sessione finale** per creare una tag Sessione finale. Modificare la tag con le seguenti impostazioni.

Impostazioni ID sessione

- **Metodo**: `Constant`
- **Costante**: `5555`
- **Tipo dati**: `String`
- **Modificatore**: `<null>`

Impostazioni Nome funzione di richiamata alla riuscita

- **Metodo**: `Unassigned`
- **Valore**: `<null>`

Impostazioni Nome funzione di richiamata all'errore

- **Metodo**: `Unassigned`
- **Valore**:`<null>`

Funzioni di esempio

Per le funzioni utilizzate per le impostazioni di Nome funzione di richiamata alla riuscita e Nome funzione di richiamata all'errore, è sufficiente specificare il nome funzione quando si crea una nuova tag se la funzione è già presente sulla pagina web.

È anche possibile utilizzare i programmi di utilità di Digital Data Exchange per creare funzioni e aggiungerle alle pagine web.

Il seguente esempio mostra in che modo visualizzare un'offerta restituita da Interact sulla pagina web. È necessario includere questo script nella pagina o utilizzare il frammento di codice Digital Data Exchange per inserirlo.

```
<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>
```

Verifica della configurazione dell'integrazione

Utilizzare lo strumento di test Digital Data Exchange e il file `Interact.log` per risolvere eventuali problemi di configurazione.

È possibile utilizzare lo strumento di test Digital Data Exchange per controllare l'encicolopedia e vedere se la configurazione funziona come previsto. Per aprire lo strumento di test, fare clic su **Distribuzione** > **Strumento di test** in Digital Data Exchange.

Consultare IBM Digital Data Exchange User Guide per ulteriori informazioni sullo strumento di test.

È possibile visualizzare il file `Interact.log` per vedere i dettagli sulle varie chiamate all'API Interact effettuate. Aggiungere la funzione di richiamata alla riuscita e la funzione di richiamata all'errore a ogni tag, per eseguire il debug delle chiamate.

Capitolo 18. Configurare i gateway per i messaggi attivati

Utilizzare i gateway dei messaggi attivati per inviare e ricevere informazioni sull'offerta da canali in entrata e in uscita.

È possibile utilizzare i seguenti gateway in entrata e in uscita con i messaggi attivati.

- Gateway in entrata IBM Interact per IBM Universal Behavior Exchange
- Gateway in uscita IBM Interact per IBM Universal Behavior Exchange
- Gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud
- Gateway in uscita IBM Interact per notifica push mobile IBM

Per ulteriori informazioni, consultare https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W214f7731a379_4712_a1ce_5d7a833d4cca/page/IBM%20Interact%20Triggered%20Messages.

Utilizzo del gateway in entrata IBM Interact per IBM Universal Behavior Exchange

Per utilizzare il gateway in entrata IBM Interact per IBM Universal Behavior Exchange, è necessario configurare Interact, configurare un endpoint sottoscrittore UBX e creare un endpoint e un evento in UBX.

Utilizzare le seguenti configurazioni come esempio per la propria.

È possibile scaricare il gateway sottoscrittore da http://www.ibm.com/support/fixcentral/swg/quickorder?parent=Enterprise%2BMarketing%2BManagement&product=ibm/Other+software/Unica+Interact&release=All&platform=All&function=fixId&fixids=IBM_Interact_OMO_Gateway_for_UBX_Subscriber_2.0&includeRequisites=1&includeSupersedes=0&downloadMethod=http&source=fc.

Configurazione di Interact per il gateway in entrata IBM Interact per IBM Universal Behavior Exchange

Utilizzare la seguente procedura per configurare Interact.

1. Nella proprietà di configurazione **Interact | activityOrceshtrator | receivers**, aggiungere un nuovo ricevitore. Impostare **Tipo** su `IBMMQ` o `Personalizzato`. Se si sceglie `Personalizzato`, immettere **ClassName** e **ClassPath**. Se si sceglie `IBMMQ`, lasciare vuoti **ClassPath** e **ClassName**.
2. Aggiungere i parametri `providerURL`, `queueManager`, `messageQueueName`, `authDS` e `asmUserFor...."` per il ricevitore.
3. Nella proprietà di configurazione **Interact | activityOrceshtrator | gateways**, aggiungere un nuovo gateway. Impostare **ClassPath** sull'URI dell'ubicazione del file `OMO_InteractGateway_UBX.jar` e **ClassName** su

   ```
   com.ibm.interact.offerorchestration.inboundgateway.ubx.
   UBXInboundGateway
   ```
4. Creare una cartella `Interactubx11` nella cartella `UBX` del gateway in entrata e copiare i file delle proprietà in questa nuova cartella. Il nome cartella deve corrispondere a quello dell'endpoint sottoscrittore creato in UBX.

5. Nel file interactEventNamemapping.properties, aggiungere una voce per associare il valore del campo evento payload al nome dell'evento Interact. Ad esempio, recommededOffers=recommendedOffers.
6. Nel file interactEventPayloadMapping.properties, aggiungere le definizioni di campo con i nomi di questi parametri impostati su OMO-conf_inbound_UBX_interactEventNameMapping e OMO-conf_inbound_UBX_interactEventNameMapping, rispettivamente

   Ad esempio:

   ```
   [SessionID]=(String)interactprofileid
   [EventName]=(String)code
   [AudienceIDFieldNames]=(String)"CustomerID"
   [AudienceIDFieldValues]=(Numeric)interactprofileid
   [AudienceLevel]=(String)"Customer"
   [InteractChannel]=(String"UBX_MM"
   ```
7. Aggiungere le ubicazioni dei file Interactubx11/interactEven tNameMapping.properties e Interactubx11/interactEven tPayloadMapping.properties come parametri per il gateway in **Interact | activityOrceshtrator | gateways | [gatewayname] | Parameter Data**.
8. Creare un canale interattivo e aggiungere un evento al canale interattivo.
9. Aggiungere una regola messaggi attivati con l'evento recommendedOffers e assegnare un'offerta alla regola.
10. Distribuire il canale interattivo.
11. Riavviare il server Interact.
12. Pubblicare un evento in UBX con un client API REST.

    Esempio di corpo dell'evento:

    ```
    {
      "channel" : "mobile",
      "identifiers" : [
        {
          "name" : "interactprofileid",
          "value" : "55"
        }
      ],
    "events" : [
      {
        "code" : "recommendedOffers",
        "timestamp" : "2015-12-28T20:16:12Z"
       }
      ]
    }
    ```
13. Controllare il log di Interact per vedere se è stato attivato l'evento messaggi attivati.

Configurazione del gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange

Questo è un campione di endpoint che è possibile utilizzare come esempio.

È anche opportuno utilizzare le istruzioni per completare le seguenti configurazioni.

- Endpoint UBX con IBM MQ
- File ubxInboundEndpoint.properties dell'endpoint
- File inboundProducerNameConfig.properties dell'endpoint
- File inboundQueueNameConfig.properties dell'endpoint
- File log4j.properties dell'endpoint

Distribuzione del gateway in entrata IBM Interact per IBM Universal Behavior Exchange e per l'endpoint

1. Scaricare e decomprimere `IBM_Interact_OMO_Gateway_for_UBX_Subscriber_2.0.zip` nella directory in cui è stato installato Interact sul server di runtime Interact.
2. Scaricare e decomprimere `IBM_Interact_OMO_Endpoint_for_UBX_Subscriber_2.0.zip` in una directory qualsiasi (ad esempio, c:\ubxInboundEndpoint) su un server delle applicazioni abilitato a JavaEE e accessibile pubblicamente o su un server web. Questo server pubblicherà nella coda JMS in entrata di Interact i dati che dovranno essere utilizzati in seguito dal gateway in entrata IBM Interact per IBM Universal Behavior Exchange.

Configurazione del gateway in entrata IBM Interact per l'endpoint del gateway in entrata Interact di IBM

Il gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange è configurato in modo da accettare le richieste da Universal Behavior Exchange e inviarle al gateway in entrata IBM Interact per IBM Universal Behavior Exchange.

Per configurare l'endpoint del gateway sottoscrittore di Universal Behavior Exchange, è necessario completare le seguenti attività

1. È necessario configurare una nuova proprietà di sistema Java (-DubxInboundEndpointConfigPath) modificando il file di configurazione nel server web o nella console di gestione del server delle applicazioni. La proprietà -D deve puntare alla directory di installazione dell'endpoint sul server. Questa directory contiene i file di configurazione per la coda JMS di destinazione e vari livelli di registrazione per l'endpoint. Ad esempio -DubxInboundEndpointConfigPath=c:\ubxInboundEndpoint.
2. Distribuire il gateway in entrata IBM Interact per il file dell'archivio web dell'endpoint di IBM Universal Behavior Exchange (ubxInboundEndpoint.war) dalla directory di installazione come descritto nella documentazione del server web o del server delle applicazioni.

Per verificare che l'endpoint sia stato installato correttamente, immettere il seguente indirizzo in un browser e ricercare il messaggio `UBX End Point is UP`.

`http://[Server]:[Port]/[ContextRoot]/UBXEndPoint`

Nota: È necessario proteggere il gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange accessibile pubblicamente aggiungendo le regole firewall necessarie per accettare le richieste http solo dal server di IBM Universal Behavior Exchange.

Ad esempio, è possibile utilizzare le seguenti istruzioni per configurare e distribuire il gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange su WebSphere Application Server.

1. Aprire la console di gestione.
2. Selezionare **Server** > **(Espandere i tipi di server)** > **nome_server** > **(Espandere Java™ e Gestione processo)** > **Definizione di processo** > **JVM (Java Virtual Machine)**.
3. Negli argomenti JVM generici, aggiungere la proprietà -DubxInboundEndpointConfigPath=<Directory di installazione dell'endpoint del gateway sottoscrittore di Universal Behavior Exchange sul server delle

applicazioni>. Ad esempio, aggiungere la proprietà -DubxInboundEndpointConfigPath=C:\ubxInboundEndpoint.

4. Fare clic su **OK** per salvare le modifiche alla configurazione principale.
5. Riavviare il server delle applicazioni.

Distribuire l'endpoint in WebSphere Application Server.

1. Accedere alla console di gestione.
2. Andare a **Applicazioni** > **Tipi di applicazione** > **Applicazioni enterprise Websphere**. Fare clic su **Installa**.
3. Utilizzare l'opzione **Preparazione per l'installazione dell'applicazione** per individuare il file war dell'endpoint (`ubxInboundEndpoint.war`) da installare, quindi fare clic su **Avanti**.
4. Fare clic su **Avanti** nelle pagine successive per raggiungere **Mapping di root di contesto per i moduli Web**.
5. Utilizzare **Mapping di root di contesto per i moduli Web** per individuare la root di contesto e modificare il valore su `/UBXEndPoint`; questa diventa la root di contesto. Fare clic su **Avanti**.
6. Fare clic su **Fine**.
7. Una volta completata l'installazione dell'applicazione, fare clic su **Salva** per conservare le modifiche alla configurazione principale.
8. Tornare alle applicazioni elencate e installate, selezionare la casella di spunta per ubxInboundEndpoint_war e fare clic su **Inizio** per eseguire il caricamento.

Configurazione del gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange con IBM MQ (facoltativo)

Per impostazione predefinita, il gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange si integra a ActiveMQ. Utilizzare le seguenti istruzioni per configurare l'endpoint con IBM MQ.

Preparazione dei file JAR di IBM MQ:

Il client che esegue l'endpoint deve disporre di determinati file JAR di IBM MQ per consentire il funzionamento delle factory di connessione.

Se IBM MQ è già installato sulla macchina dell'endpoint, i file JAR necessari sono già inclusi nell'installazione di IBM MQ. Aggiungere i seguenti due file JAR alla variabile di ambiente a livello di sistema CLASSPATH. In Windows, i file JAR vengono aggiunti automaticamente al percorso classi quando viene installato IBM MQ.

```
[MQ_HOME]\java\bin\com.ibm.mq.jar
[MQ_HOME]\java\bin\com.ibm.mqjms.jar
```

Se, invece, IBM MQ non è installato sulla macchina, è necessario copiare com.ibm.mq.allclient.jar e jms.jar dal server MQ al proprio server endpoint e aggiungerli manualmente a CLASSPATH.

Per ulteriori informazioni sull'installazione o sullo spostamento dei file JAR di IBM MQ, consultare http://www.ibm.com/support/docview.wss?uid=swg21376217.

Il server delle applicazioni deve avere in esecuzione Java 1.7 o versioni successive, in quanto i file JAR di IBM MQ v8 non supportano Java 1.6.

WebSphere Application Server include il supporto a IBM MQ e non richiede file JAR aggiuntivi.

Configurazione dell'endpoint

1. Andare alla directory <directory di installazione dell'endpoint sul server delle applicazioni>.
2. Eseguire il backup o ridenominare `ubxInboundEndpoint-spring.xml` e `ubxInboundEndpoint.properties`.
3. Andare alla directory secondaria `IBMMQ`. Conterrà versioni alternative dei file sopra indicati.
4. Aggiungere le informazioni per la connessione al server MQ a questa versione di `ubxInboundEndpoint.properties`.
5. Copiare `ubxInboundEndpoint-spring.xml` e `ubxInboundEndpoint.properties` da `/ubxInbdoundEndpoint/IBMMQ` nella directory `main/ubxInboundEndpoint`.

Configurazione del gateway in entrata IBM Interact per il file ubxInboundEndpoint.properties dell'endpoint di IBM Universal Behavior Exchange

Utilizzare il file `ubxInboundEndpoint.properties` per scegliere dove inviare il payload eventi di Universal Behavior Exchange al gateway in entrata IBM Interact per IBM Universal Behavior Exchange. Il file `ubxInboundEndpoint.properties` si trova nella directory <directory di installazione dell'endpoint del gateway sul server delle applicazioni>.

jmsBrokerUrl
: Obbligatorio - Le informazioni sulla coda JMS sulla quale il produttore scrive i dati.

jmsMaximumRetries
: Obbligatorio - Il numero massimo di tentativi di invio di un messaggio alla coda JMS.

jmsRetryDelay
: Obbligatorio - Il ritardo tra le consegne, in millisecondi.

maximumEndPointThreadPoolSize
: Obbligatorio - Il numero massimo di thread per il pool di thread per gestire i dati degli eventi di IBM Universal Behavior Exchange e scrivere sulla coda JMS. Questo numero intero definisce la dimensione del pool di thread.

clientIDFieldName
: Facoltativo - Il nome del campo utilizzato nel payload per l'ID del client (categoria secondaria). Una categoria secondaria viene utilizzata quando questo programma è in esecuzione su più istanze dello stesso prodotto. Ad esempio: clientIDFieldName=clientID

Un riavvio dell'applicazione web dell'endpoint del gateway (ubxInboundEndpoint.war) è obbligatorio per il server web o per il server delle applicazioni affinché le modifiche apportate a questo file vengano rese effettive.

Configurazione del gateway in entrata IBM Interact per il file inboundProducerNameConfig.properties dell'endpoint di IBM Universal Behavior Exchange (facoltativo)

Il gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange invia l'evento a Interact scrivendo su una coda JMS. Il messaggio evento predefinito utilizza il valore del nome del produttore `UBX`. Utilizzare il file `inboundProducerNameConfig.properties` per sovrascrivere il nome del produttore in base al valore del campo di origine UBX dal payload. Di solito, questo è il nome dell'endpoint di UBX. Il file `inboundProducerNameConfig.properties` si trova nella directory <directory di installazione dell'endpoint del gateway sul server delle applicazioni>.

SOURCE.{nome origine UBX}={nome produttore}

Esempio: SOURCE.CustomerAEndpoint=UBX-CustomerAEndpoint.

Un riavvio dell'applicazione web dell'endpoint del gateway (ubxInboundEndpoint.war) è obbligatorio per il server web o per il server delle applicazioni affinché le modifiche apportate a questo file vengano rese effettive.

Configurazione del file inboundQueueNameConfig.properties dell'endpoint del gateway (facoltativo)

Il gateway in entrata IBM Interact per l'endpoint di IBM Universal Behavior Exchange invia l'evento a Interact scrivendo su una coda JMS. Il nome predefinito della coda è identico al nome del produttore. Utilizzare il file `inboundQueueNameConfig.properties` per sovrascrivere il nome della coda JMS predefinito sul nome del produttore. Il nome del produttore predefinito è `UBX`, a meno che non sia sovrascritto nel file `inboundQueueNameConfig.properties`. Il file `inboundProducerNameConfig.properties` si trova nella directory <directory di installazione dell'endpoint del gateway sul server delle applicazioni>.

{nome produttore}={nome coda JMS}

```
Esempio:
UBX=UBXInboundQueue.
UBX-CustomerAEndpoint=UBX-CustomerAEndpointQueue
```

Un riavvio dell'applicazione web dell'endpoint del gateway (ubxInboundEndpoint.war) è obbligatorio per il server web o per il server delle applicazioni affinché le modifiche apportate a questo file vengano rese effettive.

Configurazione del file log4j.properties dell'endpoint del gateway

Utilizzare il file log4j.properties per configurare un livello di registrazione diverso per l'endpoint. Il file log4j.properties si trova nella directory <directory di installazione dell'endpoint del gateway sul server delle applicazioni>.

Descrizione

Impostare il livello di registrazione per log4j.logger.com.ibm.x1solution.jms.producer, log4j.logger.com.ibm.web.offerorchestration.inbound.common e log4j.logger.com.ibm.web.offerorchestration.inbound.ubx di conseguenza.

Configurazione del file interactEventNameMapping.properties

Utilizzare questo file per associare il valore di un campo evento del payload definito nel file `interactEventPayloadMapping.properties` come [EventName] sul

nome dell'evento di Interact. L'alternativa è utilizzare il nome dell'evento come definito dal payload eventi di Universal Behavior Exchange. Il file `interactEventNameMapping.properties` si trova nella directory <Install dir>\conf\inbound\UBX.

{nome evento UBX}={nome evento Interact}

Esempio: matchedIdentity=recommendedOfferEven

Se è necessario il supporto per i dati del payload da origini specifiche, è possibile collocare questo file nella directory <Install dir>\conf\inbound\UBX\{source}. Il valore di source deve corrispondere al valore del campo di origine nel payload eventi di Universal Behavior Exchange, solitamente il nome dell'endpoint di Universal Behavior Exchange. Se il supporto per i dati che utilizzano versioni specifiche è necessario, è possibile collocare questo file nella directory <Install dir>\conf\inbound\UBX\{source}\version-{version}. Il valore di version deve corrispondere al valore del campo versione nel payload eventi di Universal Behavior Exchange. Per supportare più dati istanza di Universal Behavior Exchange, è inoltre possibile collocare questo file nella directory <Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID}. Il valore di clientID deve corrispondere al valore di clientID nel payload eventi di Universal Behavior Exchange.

Configurazione del file interactEventPayloadMapping.properties

Utilizzare il file `interactEventPayloadMapping.properties` per associare il campo in entrata ai parametri dell'API di Interact. Il file `interactEventPayloadMapping.properties` si trova nella directory <Install dir>\conf\inbound\UBX.

Parametri dell'API di Interact: il valore deve iniziare con una definizione del tipo di campo, seguita da un valore statico quando il valore è compreso tra doppi apici o dal nome di un campo dai dati del payload. (FIELD_TYPE)"STATIC_VALUE" o (FIELD_TYPE)PAYLOAD_FIELD_NAME. FIELD_TYPE può essere `String`, `Numeric` o `DateTime`.

```
Esempio:
[SessionID]=(String)interactprofileid
[EventName]=(String)code
[AudienceIDFieldNames]=(String)"change_me"
[AudienceIDFieldValues]=(String)interactprofileid
[AudienceLevel]=(String)"change_me"
[InteractChannel]=(String)"change_me"
```

Dati evento: queste proprietà vengono utilizzate per associare gli attributi dell'evento che possono essere utilizzati nelle comunicazioni del canale in uscita. La parte sinistra contiene i nomi delle variabili utilizzate nelle comunicazioni del canale in uscita.

Il valore deve iniziare con una definizione del tipo di campo, seguita da un valore statico quando il valore è compreso tra doppi apici o dal nome di un campo dai dati del payload. (FIELD_TYPE)"STATIC_VALUE" o (FIELD_TYPE)PAYLOAD_FIELD_NAME. FIELD_TYPE può essere `String`, `Numeric` o `DateTime`.

Se è necessario il supporto per i dati del payload da origini specifiche, è possibile collocare questo file nella directory <Install dir>\conf\inbound\UBX\{source}. Il valore di source deve corrispondere al valore del campo di origine nel payload eventi di Universal Behavior Exchange, solitamente il nome dell'endpoint di

Universal Behavior Exchange. Se il supporto per i dati che utilizzano versioni specifiche è necessario, è possibile collocare questo file nella directory <Install dir>\conf\inbound\UBX\{source}\version-{version}. Il valore di version deve corrispondere al valore del campo versione nel payload eventi di Universal Behavior Exchange. Per supportare più dati istanza di Universal Behavior Exchange, è inoltre possibile collocare questo file nella directory <Install dir>\conf\inbound\UBX\{source}\version-{version}\account-{clientID}. Il valore di clientID deve corrispondere al valore di clientID nel payload eventi di Universal Behavior Exchange.

Creazione di un endpoint e di un evento in UBX

È possibile utilizzare questo endpoint ed evento come esempio.

Completare la seguente procedura per creare un endpoint e un evento in UBX.

1. Utilizzare il client API REST per pubblicare le richieste su UBX.
2. Registrare un endpoint in UBX con JSON. Fare riferimento al seguente esempio.

```
Method Call: PUT
URL: https://ubx-qa1-api.adm01.com/v1/endpoint
Headers:
Content-Type : application/json
Accept-Charset: UTF-8
Authorization: Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
(Nota: questa è una chiave di autorizzazione generata dalla UI UBX.)
Corpo
 {
   "name":"Interactubxdk1",
   "description":"Interactubxdk1",
   "providerName":"IBM", "
   "url":"http://169.38.71.122:9081/ubxEndPoint/UBXEndPoint",
   "endpointTypes":{
     "event":{
       "source":{
         "enabled":true
       },
       "destination":{
         "enabled":true,
         "url":"http://169.38.71.122:9081/UBXEndPoint/UBXEndPoint",
         "destinationType":"push"
       }
      }
},
"marketingDatabasesDefinition":{
  "marketingDatabases":[
    {
      "name":"IDSync",
      "identifiers":[
        {
      "name":"interactprofileid",
      "type":"INTERACTID"
          }
       ]
     }
  ]
}
}
```

3. Registrare un tipo di evento in UBX con JSON. Fare riferimento al seguente esempio.

```
Event Registration for Interact Event in UBX
Method Call: POST
URL: https://ubx-qa1-api.adm01.com/v1/eventtype
```

```
Headers:
Content-Type : application/json
Accept-Charset: UTF-8
Authorization: Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
(Nota: questa è una chiave di autorizzazione generata dalla UI UBX.)
Bearer 912586bf-190d-48f9-8488-26f1bf532ef3
Corpo
{
   "name": "recommendedOffers",
   "description": "recommended offers by OMO",
   "code": "recommendedOffers"
}
```

4. Pubblicare un evento in UBX con JSON. Fare riferimento al seguente esempio.

```
{
  "channel" : "mobile",
  "identifiers" : [
    {
      "name" : "interactprofileid",
      "value" : "55"
    }
  ],
"events" : [
  {
    "code" : "recommendedOffers",
    "timestamp" : "2015-12-28T20:16:12Z"
   }
  ]
}
```

Utilizzo del gateway in uscita IBM Interact per IBM Universal Behavior Exchange

Per utilizzare il gateway in uscita IBM Interact per IBM Universal Behavior Exchange, è necessario configurare Interact, UBX e il gateway.

Utilizzare le seguenti configurazioni come esempio per la propria.

Se si utilizza UBX come un canale in uscita, Interact agisce come endpoint publisher, che pubblica eventi su UBX. Da UBX questi eventi possono essere inviati al sottoscrittore.

Prima di iniziare la configurazione, richiedere l'accesso in uscita alla macchina host. È necessario abilitare l'accesso di rete per la macchina host.

È possibile scaricare il gateway da http://www.ibm.com/support/fixcentral/swg/quickorder?parent=Enterprise%2BMarketing%2BManagement&product=ibm/Other+software/Unica+Interact&release=All&platform=All&function=fixId&fixids=IBM_Interact_OMO_Gateway_for_UBX_Publisher_2.0&includeRequisites=1&includeSupersedes=0&downloadMethod=http&source=fc.

Registrazione di endpoint ed eventi in UBX

1. Da UBX, andare alla scheda **Endpoint**. Fare clic su **Registra nuovo endpoint** per ottenere una chiave di autorizzazione. La chiave di autorizzazione generata da UBX deve essere utilizzata per l'endpoint publisher e per aggiungere eventi. Per l'endpoint sottoscrittore, la nuova chiave di autorizzazione deve essere generata da UBX. Annotare la chiave.
2. Registrare l'endpoint publisher.
   a. Aprire lo strumento client API REST.

b. Selezionare il metodo PUT.

c. Passare le intestazioni come

```
Content-Type : application/json
Accept-Charset : UTF-8
Authorization : Bearer 520301d7-7855-4ea7-b19d-0b395c1e6ae4
(authKey generated in UBX)
```

d. Passare l'URL come

```
URL: https://ubx-qa1-api.adm01.com/v1/endpoint
```

e. Per il corpo, passare il nome appropriato per l'endpoint publisher.

Ad esempio:

```
{
    "name":"Interact_Publisher",
    "description":"Endpoint for server created on 30thJan",
    "providerName":"IBM",    "url":"",
    "endpointTypes":{
        "event":{
        "source":{
           "enabled":true
        }
    }
},
"marketingDatabasesDefinition":{
    "marketingDatabases":[
      {
          "name":"IDSync",
          "identifiers":[
            {
          "name":"interactprofileid",
     "type":"INTERACTID"
        }
      ]
    }
   ]
}
}
```

3. Registrare l'evento. Annotare il codice [Event] passato nel corpo. Dovrà essere riassociato nel file ubxContentMapping.properties. È sensibile al maiuscolo/minuscolo.

a. Aprire lo strumento client API REST.

b. Selezionare il metodo POST.

c. Passare le stesse intestazioni utilizzate per l'endpoint nello step precedente.

d. Passare l'URL come

```
URL: https://ubx-qa1-api.adm01.com/v1/eventtype
```

e. Per il corpo, passare il nome appropriato per l'evento.

Ad esempio:

```
{
   "name": "recommendedOffer",
   "description": "recommended
      contact frm UBX",    "code":
      "recommendedOffer"}
```

Nota: il codice evento passato deve essere riassociato nel file ubxContentMapping.properties. Il codice evento è sensibile al maiuscolo/minuscolo.

4. Aggiungere l'endpoint sottoscrittore.

a. Aprire lo strumento client API REST.

b. Selezionare il metodo PUT.

c. Passare le stesse intestazioni utilizzate per l'endpoint nello step precedente.

d. Per registrare l'endpoint sottoscrittore, creare una nuova chiave di autorizzazione in UBX.

e. Passare l'URL come

```
URL: https://ubx-qa1-api.adm01.com/v1/endpoint
```

f. Per il corpo, passare il nome appropriato per l'endpoint publisher.

Ad esempio:

```
{
   "name":"UBX_Subscriber",
   "description":"UBX_Subscriber for Subscribing Events ",
   "providerName":"IBM",
   "url":"http://ubxeventconsumer.mybluemix.net/ubxeventconsumer",
   "endpointTypes":{
      "event":{
        "source":{
          "enabled":true
        },
      "destination":{
        "enabled":true,
         "url":"http://ubxeventconsumer.mybluemix.net
          /ubxeventconsumer",
      }
   }
},
"marketingDatabasesDefinition":   {
  "marketingDatabases":[
    {
       "name":"IDSync",
       "identifiers":[
          {
        "name":"interactprofileid",
     "type":"INTERACTID"
          }
       ]
   }
 ]
}
```

5. Dopo aver aggiunto gli endpoint publisher e sottoscrittore e l'evento, è necessario sottoscrivere l'evento dal publisher al sottoscrittore in UBX.

a. In UBX, fare clic su **Sottoscrivi a eventi** nella scheda **Eventi**.

b. Selezionare l'evento e la destinazione.

c. Fare clic su **Sottoscrivi**.

Configurazione di Interact e del gateway

1. Aggiungere il gateway UBX nella proprietà di configurazione **Interact | triggeredMessage | gateways**. Impostare **ClassPath** su `file:///root/opt/OMO/lib/OMO_OutboundGateway_UBX.jar` e **ClassName** su

   `com.ibm.interact.offerorchestration.outboundgateway.ubx.UBXOutboundGateway`

2. Decomprimere il file `OMO_OutboundGateway_UBX.zip` sulla macchina host e puntarlo al jar UBX dal percorso estratto.

3. Aggiungere `OMO-conf_outbound_common_httpConnectionConfig` come parametro in **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Impostare il **valore** su `file:///opt/Interact<version>/Interact/OMO/conf/outbound/common/httpConnectionConfig.properties`. Questa è la directory di installazione di Interact. Il programma di installazione del gateway scarica la directory del gateway nella directory di installazione di Interact.

Nel file `httpConnectionConfig.properties`, nella cartella Interact, specificare il valore di timeout.

Ad esempio:

```
connectTimeoutMs=180000
```

4. Aggiungere `OMO-conf_outbound_ubx_ubxConfig` come parametro in **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Impostare il **valore** sul percorso del file `ubxConfig.properties` nella cartella Interact.

   Nel file `ubxConfig.properties`, specificare `ubxURL`, `authKey` e `interactProfileIdFieldName`.

   Ad esempio:

   ```
   authKey=912586bf-190d-48f9-8488-26f1bf532ef3
   [Auth Key used to register publisher endpoint and event in UBX]
   interactProfileIdFieldName=interactprofileid
   [Field name from the ubxContentMapping.properties file]
   ```

5. Aggiungere `OMO-conf_outbound_ubx_ubxContentAdditionalAttributes` come parametro in **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Impostare il **valore** sul percorso del file `ubxContentAdditionalAttributes.properties` nella cartella Interact.

6. Aggiungere `OMO-conf_outbound_ubx_ubxContentMapping` come parametro in **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Impostare il **valore** sul percorso del file `ubxContentMapping.properties` nella cartella Interact.

   Aggiornare i valori per `interactprofileid` e `eventName` nel file `ubxContentMapping.properties`.

   È possibile passare il nome evento in 3 formati: quando il valore è tra virgolette, è un valore statico; quando è nel formato `offer.offerAttributeName`, si associa all'attributo offerta offerAttributeName; quando il valore è nel formato `profile.profileAttributeName`, si associa all'attributo profilo profileAttributeName. Il valore del nome evento deve corrispondere al codice utilizzato per registrare l'evento in UBX. È sensibile al maiuscolo/minuscolo.

   Ad esempio:

   ```
   eventName="abandoned_shopping_carts"
   eventName=offer.Card
   eventName=profile.EMAIL
   ```

7. Aggiungere un canale nella proprietà di configurazione **Interact | triggeredMessage | channel**.
8. Definire lo stesso canale nella fase di progettazione in **Campaign | partitions | partition [n] |Interact | outboundChannels**
9. Riavviare il server delle applicazioni.
10. Creare una regola messaggi attivati con un nome evento e che utilizzi il canale aggiunto nello step precedente.
11. Distribuire il canale interattivo.
12. Dal client di test API, avviare la sessione per il canale interattivo in cui è configurata la regola messaggi attivati e l'evento di invio che attiva l'offerta in UBX.

Utilizzo del gateway in uscita IBM Interact per la notifica push mobile IBM

Per utilizzare questo gateway publisher o in uscita push mobile è necessario configurare Interact, IBM Marketing Cloud e il gateway.

Utilizzare le seguenti configurazioni come esempio per la propria.

È possibile scaricare questo gateway da https://www-945.ibm.com/support/fixcentral/swg/downloadFixes

Configurazione di IBM Marketing Cloud

1. Assicurarsi di disporre di un account IBM Marketing Cloud con accesso push. Prendere nota dell'ID client, del segreto client e del token di aggiornamento.
2. Sulla scheda **Dati**, creare un nuovo database. Aggiungere un nuovo `ID utente mobile` al database e riempire i campi predefiniti.
3. Sulla scheda **Cerca**, cercare in base al campo `ID utente mobile`. Passare il puntatore del mouse sul primo campo `Nessuna email`. Viene visualizzato l'`ID destinatario` alla fine della finestra del browser. Aggiungere l'`ID destinatario` alla tabella profili di Interact.

Configurazione del gateway in uscita IBM Interact per la notifica push mobile IBM

1. Scaricare e installare il gateway in uscita push mobile da https://www-945.ibm.com/support/fixcentral/swg/downloadFixes
2. Configurare il file `silverpopEngagePushConfig.properties`.

   Ad esempio:

   ```
   OauthServiceURL=https://apipilot.silverpop.com/oauth/token
   pushServiceURL=https://apipilot.silverpop.com/rest/channels/push/sends
   ```
3. Configurare il file `silverpopEngagePushContentMapping.properties`.

   Ad esempio:

   ```
   Attributi tabella profili Interact:
   appKey=appKey
   engageRecipientId=recipientId
   mobileUserId=mobileUserId
   deviceType=deviceType

   Attributi offerta Interact:
   simpleSubject=simpleSubjectAttr
   simpleMessage=simpleMessageAttr
   simpleActionData=simpleActionDataAttr
   simpleActionType=simpleActionTypeAttr
   simpleActionLabel=simpleActionLabelAttr
   personalizeAttributeList=personalizeAttributeList
   contentId=ContentID
   campaignId=campaignId
   ```

Configurazione di Interact

1. Creare i seguenti attributi offerta.

   ```
   simpleActionDataAttr: string
   simpleActionLabelAttr: String
   simpleActionTypeAttr: string
   simpleMessageAttr: string
   ```

```
simpleSubjectAttr: string
contentID: string
campaignId=string
personalizeAttributeList=string
```

2. Creare un modello dell'offerta con gli attributi dell'offerta e i seguenti valori offerta.

   ```
   simpleActionDataAttr: www.ibm.com
   simpleActionLabelAttr: Open URL
   simpleActionTypeAttr: url
   simpleMessageAttr: <Immettere il messaggio di testo>
   simpleSubjectAttr: <Immettere l'oggetto>
   contentID: ID del modello messaggio push creato in Engage.
   PersonalizeAttributeList: Un elenco di nomi attributi separati da virgole
   coppie di valori che si desidera inserire nella sezione personalizationDefaults
   del payload da inviare a Engage.
   ```

   Quando si utilizza l'attributo `contentID`, gli altri attributi `simple..` sono ignorati perché i dettagli completi sono presi dal modello Engage.

   Esempio personalizedAttributeList

   ```
   personalizeAttributeList=discount=10,Offercost=20
   campaignId=nomecampagna da utilizzare per questa campagna.
   ```

3. La tabella profili ha le seguenti colonne e valori.

   ```
   appKey: gcsTQo6v79
   recipientId: 13472242
   deviceType: android o ios
   ```

4. Aggiungere il gateway nella proprietà di configurazione **Interact | triggeredMessage | gateways**. Impostare **ClassName** su

   ```
   com.ibm.interact.offerorchestration.outboundgateway.silverpop.engage.push.
   SilverpopEngagePushOutboundGateway
   ```

   Impostare **ClassPath** su `file://<EngagePushGateway_home_dir>/lib/OMO_OutboundGateway_Silverpop_Engage_Push.jar`.

5. Aggiungere `OMO-silverpopEngagePushConfig` come parametro in **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Impostare il **valore** sul percorso del file `silverpopEngagePushConfig.properties`.
6. Aggiungere `OMO-silverpopEngagePushContentMapping` come parametro in **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Impostare il **valore** sul percorso del file `silverpopEngagePushContentMapping.properties`.
7. Aggiungere `OMO-conf_outbound_common_httpConnectionConfig` come parametro in **Interact | triggeredMessage | gateways | [gatewayName] | Parameter Data**. Impostare il **valore** sul percorso del file `httpConnectionConfig.properties`.

   Nel file `httpConnectionConfig.properties`, nella cartella Interact, specificare il valore di timeout.

   Ad esempio:

   ```
   connectTimeoutMs=6000
   ```

8. Creare un canale e un gestore in **Interact | triggeredMessage** e utilizzare il gateway [Push_mobile] creato precedentemente in tale canale. Questo canale viene utilizzato nel messaggio attivato per inviare i messaggi push.
9. Creare un canale interattivo e aggiungere un messaggio attivato che utilizzi l'offerta creata in precedenza per la regola di trigger.
10. Distribuire il canale interattivo.

11. Dal client di test API, eseguire uno `startSession` per il canale interattivo in cui è configurata la regola messaggi attivati e il `postEvent` che attiva l'offerta in push mobile.
12. Controllare i log di Interact per verificare che il push sia stato inviato correttamente. Il codice di stato `202` indica una consegna riuscita.

Utilizzo del gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud

È possibile utilizzare questa integrazione con Silverpop, Interact e con il gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud per inviare offerte email attivate ai clienti.

Assicurarsi che i seguenti prerequisiti siano soddisfatti.

- Creare una tabella profili destinatario cliente con una colonna email. Utilizzare questa tabella profili per il canale interattivo.
- Richiedere che l'accesso di rete alla macchina host sia abilitato per il canale in uscita.
- Copiare ed estrarre il file `OMO_OutboundGateway_Silverpop.zip` sulla macchina host

È possibile scaricare il gateway da http://www.ibm.com/support/fixcentral/swg/quickorder?parent=Enterprise%2BMarketing%2BManagement&product=ibm/Other+software/Unica+Interact&release=All&platform=All&function=fixId&fixids=IBM_Interact_OMO_OutboundGateway_Silverpop_2.0&includeRequisites=1&includeSupersedes=0&downloadMethod=http&source=fc.

Aggiunta di un dispatcher per l'integrazione del gateway

Il dispatcher aggiunge l'offerta in una coda per il gateway Email in uscita (Transact) IBM Interact per IBM Marketing Cloud in modo che la propria email di offerta possa essere inviata.

Informazioni su questa attività

È necessario aggiungere un dispatcher per utilizzare il gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud.

Procedura

1. Andare a **Interact | triggeredMessage | dispatchers | <dispatcherName>** nelle proprietà di configurazione.
2. Aggiungere un **Nuovo nome categoria** per il dispatcher.
3. Selezionare un **tipo**. È possibile scegliere tra InMemoryQueue, JMSQueue e Custom.
4. Immettere il **className**.
5. Immettere il **classPath**.

Aggiunta di un gateway per il gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud

Nell'integrazione, il gateway invia offerte idonee ai clienti via email.

Informazioni su questa attività

È necessario aggiungere un gateway per l'integrazione.

Nota: Interact non supporta più istanze dello stesso gateway.

Procedura

1. Andare a **Interact | triggeredMessage | gateways | <gatewayName>** nelle proprietà di configurazione.
2. Aggiungere un **Nuovo nome categoria** per il gateway.
3. Impostare il **className** sul seguente percorso.

   ```
   com.ibm.interact.offerorchestration.outboundgateway.
   silverpop.SilverpopEmailOutboundGateway
   ```

4. Impostare il **classPath** sull'ubicazione del percorso del jar del gateway in uscita dalla cartella estratta.

   Ad esempio:

   ```
   file:///opt/OMO_SilverPop/OMO_OutboundGateway_Silverpop/lib/
   OMO_OutboundGateway_Silverpop.jar
   ```

5. Aggiungere i seguenti parametri al gateway.

   ```
   OMO-conf_outbound_common_httpConnectionConfig
   OMO-conf_outbound_silverpop_silverpopConfig
   OMO-conf_outbound_silverpop_silverpopContentMapping
   deliveryTimeoutMillis
   ```

Configurazione del parametro OMO-conf_outbound_common_httpConnectionConfig

È necessario configurare il parametro OMO-conf_outbound_common_httpConnectionConfig per il gateway.

Procedura

1. Andare a **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | OMO-conf_outbound_common_httpConnectionConfig** nelle proprietà di configurazione.
2. Impostare il **valore** su `file:///opt/Interact<version>/Interact/OMO/conf/outbound/common/httpConnectionConfig.properties`. Questa è la directory di installazione di Interact. Il programma di installazione di Interact scarica il file `httpConnectionConfig.properties` nella directory di installazione di Interact.
3. Nel file `httpConnectionConfig.properties`, nella cartella Interact, specificare il valore di timeout.

   Ad esempio:

   ```
   connectTimeoutMs=60000
   ```

Configurazione del parametro OMO-conf_outbound_silverpop_silverpopConfig

È necessario configurare il parametro OMO-conf_outbound_silverpop_silverpopConfig per il gateway.

Procedura

1. Andare a **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | OMO-conf_outbound_silverpop_silverpopConfig** nelle proprietà di configurazione.

2. Impostare il **valore** sul percorso del file `silverpopConfig.properties` nella cartella `OMO_OutboundGateway_silverpop`.

   Ad esempio:

   `file:///opt/OMO_SilverPop/OMO_OutboundGateway_Silverpop/conf/outbound/silverpop/silverpopConfig.properties`

3. Nel file `silverpopConfig.properties` nella cartella estratta `OMO_OutboundGateway_Silverpop.zip`, impostare i valori per `OauthServiceURL`, `xmlAPIServiceURL`, `clientID`, `clientSecret` e `refreshToken`. Consultare l'amministratore di Marketing Cloud per ottenere i valori specifici del cliente dal file `transact.xml`.

Configurazione del parametro OMO-conf_outbound_silverpop_silverpop ContentMapping

È necessario configurare il parametro OMO-conf_outbound_silverpop_silverpopContentMapping per il gateway.

Procedura

1. Andare a **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | OMO-conf_outbound_silverpop_silverpopContentMapping** nelle proprietà di configurazione.
2. Impostare il **valore** sul percorso del file `silverpopContentMapping.properties` nella cartella `OMO_OutboundGateway_silverpop`.
3. Nel file `silverpopContentMapping.properties` nella cartella `OMO_OutboundGateway_Silverpop.zip`, impostare il valore per il mapping del contenuto.
   a. Impostare la proprietà `campaignId`. Il valore per questa proprietà è un nome attributo dell'offerta specificato nei modelli dell'offerta.
   b. Impostare la proprietà `email`. Il valore per questa proprietà è il nome colonna nella tabella profili. Aggiungere una colonna `email` nella tabella profili e specificare gli ID email. Questi sono gli ID email dei destinatari.
   c. Definire gli attributi dell'offerta in `additionalOfferPfAttributesUsedInEmail`. Questa proprietà imposta gli attributi dal modello dell'offerta necessari per il modello del servizio di mailing. È possibile utilizzare `additionalProfilePfAttributesUsedInEmail` per definire i campi dalla tabella profili. È possibile utilizzare `*` per considerare tutti gli attributi dell'offerta e i valori colonna.

Configurazione del parametro deliveryTimeoutMillis

Per aumentare il valore di timeout del server Interact per connettersi con il server Marketing Cloud, impostare il parametro `deliveryTimeoutMills`.

Informazioni su questa attività

Procedura

1. Andare a **Interact | triggeredMessage | gateways | <SilverpopGatewayName> | deliveryTimeoutMillis** nelle proprietà di configurazione.
2. Impostare il **valore**. Ad esempio, è possibile impostare il **valore** su 60000. Ciò aumenterà il timeout del server a 60000 millisecondi.

Aggiungere un gestore canali per il gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud

Aggiungere un gestore canali nell'ambiente di runtime Interact.

Procedura

1. Andare a **Interact | triggeredMessage | channels | <SilverpopChannelName> | <handlerName>** nelle proprietà di configurazione.
2. Aggiungere un **Nuovo nome categoria** per il gestore canali.
3. Impostare il nome del dispatcher aggiunto in precedenza.
4. Impostare il nome del gateway aggiunto in precedenza.
5. Impostare la **modalità**. Se è selezionato Failover, questo gestore viene utilizzato solo quando tutti i gestori con priorità più elevate definiti in questo canale non sono riusciti a inviare le offerte. Se è selezionato **Componente aggiuntivo**, questo gestore viene utilizzato indipendentemente dal fatto che altri gestori hanno correttamente inviato offerte.
6. Impostare la **priorità** per questo gestore.

Aggiunta di un canale in uscita per il gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud

Aggiungere un canale in uscita nell'ambiente di progettazione Interact.

Procedura

1. Andare a **Campaign | partitions | partition[n] | Interact | outboundChannels** nelle proprietà di configurazione.
2. Aggiungere un **Nuovo nome categoria** per il canale in uscita.
3. Aggiungere un **nome** per il canale in uscita. Assicurarsi che il nome del canale sia uguale a quello aggiunto nella proprietà di configurazione **Interact | triggeredMessage | channels | <SilverpopChannelName>**.

Configurazione del servizio di mailing transazionale con il gateway email in uscita (Transact) IBM Interact per IBM Marketing Cloud

È necessario configurare il servizio di mailing transazionale per inviare l'offerta email.

Procedura

1. In Marketing Cloud (Transact), fare clic su **Dati** > **Crea database**. Quindi fare clic su **Create** per creare una tabella profili. È anche possibile importare la tabella profili in cui è stata aggiunta al colonna email.
2. Fare clic su **Automation** > **Transactional messages** > **Create Group**. Selezionare **Transact** per **Event Trigger**. È anche necessario selezionare l'origine dati creata in precedenza. Fare clic su **Save & Activate**.

   L'offerta inviata tramite Marketing Cloud deve avere lo stesso attributo impostato per `campaignId` nel file `silverpopContentMapping.properties`. Il valore per questo attributo dell'offerta è il `campaignId` generato per il gruppo di messaggi automatizzati.
3. Fare clic su **Content** > **Create Mailings** e selezionare l'origine del contenuto dallo step precedente. Immettere il corpo del servizio di mailing. Fare clic su

Automate. Selezionare **Assign Mailing to Existing Group of Automated Messages**. Fare clic su **Submit & Activate**.

La riga oggetto del servizio di mailing può essere personalizzata utilizzando gli attributi dell'offerta e quelli del profilo. Utilizzare la sintassi `%%Attribute_Name%%` per definire gli attributi.

4. Il server Marketing Cloud accetta solo inoltri di gateway in uscita da indirizzi IP impostati in precedenza. Per aggiungere un indirizzo IP, andare a **Settings** > **Org Admin** > **Security Settings** > **Access Restrictions**.
5. Se si utilizza WebSphere Application Server, è necessario importare il certificato SSL di Marketing Cloud. Non è necessario per gli utenti WebLogic.
   a. Nella console di WebSphere Application Server, andare a **Gestione chiavi e certificati SSL** > **Keystore e certificati** > **NodeDefaultTrustStore** > **Certificati firmatario** > **Richiama da porta**.
   b. Impostare l'host e la porta.
   c. Riavviare WebSphere Application Server.

Before you contact IBM technical support

If you encounter a problem that you cannot resolve by consulting the documentation, your company's designated support contact can log a call with IBM technical support. Use these guidelines to ensure that your problem is resolved efficiently and successfully.

If you are not a designated support contact at your company, contact your IBM administrator for information.

Nota: Technical Support does not write or create API scripts. For assistance in implementing our API offerings, contact IBM Professional Services.

Information to gather

Before you contact IBM technical support, gather the following information:

- A brief description of the nature of your issue.
- Detailed error messages that you see when the issue occurs.
- Detailed steps to reproduce the issue.
- Related log files, session files, configuration files, and data files.
- Information about your product and system environment, which you can obtain as described in "System information."

System information

When you call IBM technical support, you might be asked to provide information about your environment.

If your problem does not prevent you from logging in, much of this information is available on the About page, which provides information about your installed IBM applications.

You can access the About page by selecting **Help > About**. If the About page is not accessible, check for a `version.txt` file that is located under the installation directory for your application.

Contact information for IBM technical support

For ways to contact IBM technical support, see the IBM Product Technical Support website: (http://www.ibm.com/support/entry/portal/open_service_request).

Nota: To enter a support request, you must log in with an IBM account. This account must be linked to your IBM customer number. To learn more about associating your account with your IBM customer number, see **Support Resources > Entitled Software Support** on the Support Portal.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
B1WA LKG1
550 King Street
Littleton, MA 01460-1250
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Privacy Policy and Terms of Use Considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. A cookie is a piece of data that a web site can send to your browser, which may then be stored on your computer as a tag that identifies your computer. In many cases, no personal information is collected by these cookies. If a Software Offering you are using enables you to collect personal information through cookies and similar technologies, we inform you about the specifics below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name, and other personal information for purposes of session management, enhanced user usability, or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

Various jurisdictions regulate the collection of personal information through cookies and similar technologies. If the configurations deployed for this Software Offering provide you as customer the ability to collect personal information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for providing notice and consent where appropriate.

IBM requires that Clients (1) provide a clear and conspicuous link to Customer's website terms of use (e.g. privacy policy) which includes a link to IBM's and Client's data collection and use practices, (2) notify that cookies and clear gifs/web beacons are being placed on the visitor's computer by IBM on the Client's behalf along with an explanation of the purpose of such technology, and (3) to the extent required by law, obtain consent from website visitors prior to the placement of cookies and clear gifs/web beacons placed by Client or IBM on Client's behalf on website visitor's devices

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Online Privacy Statement at: http://www.ibm.com/privacy/details/us/en section entitled "Cookies, Web Beacons and Other Technologies."

Stampato in Italia