IBM Unica Interact

버전 8 릴리스 6

2012년 5월 25일

관리자 가이드

참고

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 반드시 277 페이지의 『주의사항』의 일반 정보를 읽으십시오.

이 개정판은 새 개정판에 표시되지 않는 한 IBM Unica Interact(제품 번호 5725-D22) 버전 8, 릴리스 5, 수정 0과 모든 후속 릴리스 및 수정판에 적용됩니다.

© Copyright IBM Corporation 2001, 2012.

목차

© Copyright IBM Corp. 2001, 2012

제 1 장 IBM Unica Interact 관리

Interact 관리는 여러 작업으로 이루어집니다. 이 작업은 다음을 포함하지만 단지 이 작업으로만 제한되지는 않습니다.

- 사용자 및 역할 유지보수
- 데이터 소스 유지보수
- Interact 선택적 오퍼 제공 기능 구성
- 런타임 환경 성능 모니터링 및 유지보수

Interact 관리를 시작하기 전에 작업을 보다 쉽게 수행하도록 이해해야 하는 Interact의 작동 방식에 관한 몇 가지 핵심 개념이 있습니다. 다음 섹션에서 Interact와 연관된 관리 작업을 설명합니다.

이 가이드의 두 번째 파트는 Interact의 사용 가능한 API(Application Programming Interface)인 Interact API, ExternalCallout API, 학습 API에 대해 설명합니다.

Interact 핵심 개념

이 섹션은 Interact에 대해 작업하기 전에 이해해야 하는 몇 가지 핵심 개념을 설명합니다.

대상 수준

대상 수준은 캠페인의 대상이 될 수 있는 ID 콜렉션입니다. 예를 들어, 캠페인 세트에 "가정," "잠재 고객," "고객," "계정" 대상 수준을 사용할 수 있습니다. 각 수준은 캠페인에 사용 가능한 마케팅 데이터의 특정 보기를 나타냅니다.

대상 수준은 일반적으로 계층 구조순으로 구성됩니다. 위의 예를 사용하면,

- 가정은 계층 구조의 맨 위에 있으며 하나 이상의 잠재 고객과 많은 고객을 포함할 수 있습니다.
- 고객은 계층 구조의 그 다음이며 각 고객에 여러 계정이 있을 수 있습니다.
- 계정은 계층 구조의 맨 아래에 있습니다.

비즈니스 대 비즈니스 환경에는 보다 복잡한 다른 대상 계층 구조 예가 존재하며, 이 예에는 비즈니스, 회사, 부서, 그룹, 개인, 계정 등의 대상 수준이 존재해야 합니다.

대상 수준은 서로 일대일, 다대일 또는 다대다 등 각기 다른 관계에 있을 수 있습니다. 대상 수준을 정의하면 대상화 용도로 사용자가 여러 다른 대상 사이의 관계를 관리할

© Copyright IBM Corp. 2001, 2012

수 있도록 Campaign 내에 대상 수준 개념을 표시할 수 있습니다. 예를 들어, 가정별로 여러 잠재 고객이 있을 수 있지만 가정당 한 잠재 고객만으로 메일링을 제한할 수 있습니다.

디자인 환경

디자인 환경은 Interact 구성의 대부분을 수행하는 위치입니다. 디자인 환경에서 이벤트, 상호작용 지점, 스마트 세그먼트, 처리 규칙을 정의합니다. 이러한 구성 요소를 구성한 후 런타임 환경에 배포합니다.

디자인 환경은 Campaign 웹 응용 프로그램과 함께 설치됩니다.

이벤트

이벤트는 런타임 환경에서 작업(예: 세그먼트에 방문자 배치, 오퍼 제공 또는 데이터 로깅)을 트리거하는 작업으로, 방문자가 수행합니다.

이벤트는 먼저 대화식 채널에 작성된 후 `postEvent` 메소드를 사용하여 Interact API 호출에 의해 트리거됩니다. 이벤트는 Interact 디자인 환경에 정의된 다음 중 하나 이상의 작업을 초래할 수 있습니다.

- 재세그먼트 트리거
- 오퍼 컨택 로그
- 오퍼 수락 로그
- 오퍼 거부 로그

이벤트를 사용하여 `postEvent` 메소드가 정의한 작업(테이블에 데이터 로깅, 학습 데이터 포함, 개별 플로차트 트리거 등)도 트리거할 수 있습니다.

디자인 환경에서는 편의상 이벤트를 범주로 구성할 수 있습니다. 런타임 환경에서 범주는 기능적 목적이 없습니다.

대화식 채널

대화식 채널은 Campaign에서 인터페이스 메소드가 대화식 대화 상자인 접점을 나타냅니다. 이 소프트웨어 표시는 대화식 마케팅에 관련된 모든 개체, 데이터, 서버 자원을 조정하는 데 사용됩니다.

대화식 채널은 상호작용 지점 및 이벤트를 정의하는 데 사용하는 도구입니다. 대화식 채널의 분석 탭에서 대화식 채널에 대한 보고서에도 액세스할 수 있습니다.

대화식 채널은 운용 런타임 및 스테이징 서버 지정도 포함합니다. 몇 개의 대화식 채널을 작성하여 운용 런타임 및 스테이징 서버 집합이 하나만 있는 경우 이벤트 및 상호작용 지점을 구성하거나 이벤트 및 상호작용 지점을 고객 대면 시스템별로 분할할 수 있습니다.

대화식 플로차트

대화식 플로차트는 Campaign 일괄처리 플로차트와 관련이 있지만 약간 다릅니다. 대화식 플로차트는 일괄처리 플로차트와 동일한 주요 기능(고객을 세그먼트로 알려진 그룹으로 분할)을 수행합니다. 그러나 대화식 플로차트의 경우 그룹은 스마트 세그먼트입니다. Interact는 이 대화식 플로차트를 사용하여 동작 이벤트 또는 시스템 이벤트가 방문자 재세그먼트가 필요함을 표시할 때 세그먼트에 프로파일을 지정합니다.

대화식 플로차트는 몇 개의 대화식 플로차트 특정 프로세스는 물론 일괄처리 플로차트 프로세스 하위 집합을 포함합니다.

참고: Campaign 세션에서만 대화식 플로차트를 작성할 수 있습니다.

상호작용 지점

상호작용 지점은 오퍼를 제공할 접점의 위치입니다. 상호작용 지점은 제공할 기타 적합한 컨텐츠가 런타임 환경에 없는 경우 기본 필러 컨텐츠를 포함합니다.

상호작용 지점을 영역으로 구성할 수 있습니다.

오퍼

오퍼는 다양한 방식으로 전달할 수 있는 단일 마케팅 메시지를 나타냅니다.

Campaign에서 하나 이상의 캠페인에 사용할 수 있는 오퍼를 생성합니다.

오퍼는 다음의 경우 재사용 가능합니다.

- 다른 캠페인에
- 제 시간에 다른 위치에서
- 다른 개인 그룹에(셀)
- 오퍼의 매개변수화된 필드를 다양화하여 다른 "버전"으로

컨택 프로세스 중 하나를 사용하여 플로차트의 대상 셀에 오퍼를 지정하고 오퍼를 받은 고객과 응답한 고객에 대한 데이터를 캡처하여 캠페인 결과를 추적합니다.

프로파일

프로파일은 런타임 환경에서 사용하는 고객 데이터 집합입니다. 이 데이터는 고객 데이터베이스에서 사용 가능한 고객 데이터 하위 집합, 실시간으로 수집된 데이터 또는 둘의 조합일 수 있습니다. 이 데이터는 다음 목적으로 사용됩니다.

- 실시간 상호작용 시나리오에서 하나 이상의 스마트 세그먼트에 고객 지정

  세그먼트로 나눌 각 대상 수준에 대한 프로파일 데이터 집합이 필요합니다. 예를 들어, 위치별로 세그먼트로 나누는 중이면 보유하고 있는 모든 주소 정보에서 고객의 우편 번호만 포함시킬 수 있습니다.

- 오퍼 사용자 개인 설정
- 학습에 대해 추적할 속성으로

  예를 들어, 방문자의 결혼 여부 및 특정 오퍼를 수락하는 각 상태의 방문자 수를 모니터하도록 Interact를 구성할 수 있습니다. 그러면 런타임 환경이 해당 정보를 사용하여 오퍼 선택을 세분화할 수 있습니다.

이 데이터는 런타임 환경의 경우 읽기 전용입니다.

런타임 환경

런타임 환경은 접점에 연결하고 상호작용을 수행합니다. 런타임 환경은 접점에 연결된 하나 또는 다수의 런타임 서버로 구성될 수 있습니다.

런타임 환경은 디자인 환경에서 배포된 정보를 Interact API와 결합하여 사용하여 접점에 오퍼를 제공합니다.

런타임 세션

런타임 세션은 각 접촉점 방문자마다 런타임 서버에 있습니다. 이 세션은 런타임 환경이 세그먼트 및 권장되는 오퍼에 방문자를 지정하는 데 사용하는 방문자에 대한 모든 데이터를 보유합니다.

`startSession` 호출을 사용할 때 런타임 세션을 작성하십시오.

접점

접점은 고객과 상호작용할 수 있는 응용 프로그램 또는 위치입니다. 접점은 고객이 컨택을 시작("인바운드" 상호작용)하거나 사용자가 고객과 컨택("아웃바운드" 상호작용)하는 채널일 수 있습니다. 일반적인 예는 웹 사이트 및 콜센터 응용 프로그램입니다. Interact API를 사용하면 Interact를 접점과 통합하여 접점에서의 작업을 기준으로 고객에게 오퍼를 제공할 수 있습니다. 접점을 클라이언트 대면 시스템(CFS)이라고도 합니다.

처리 규칙

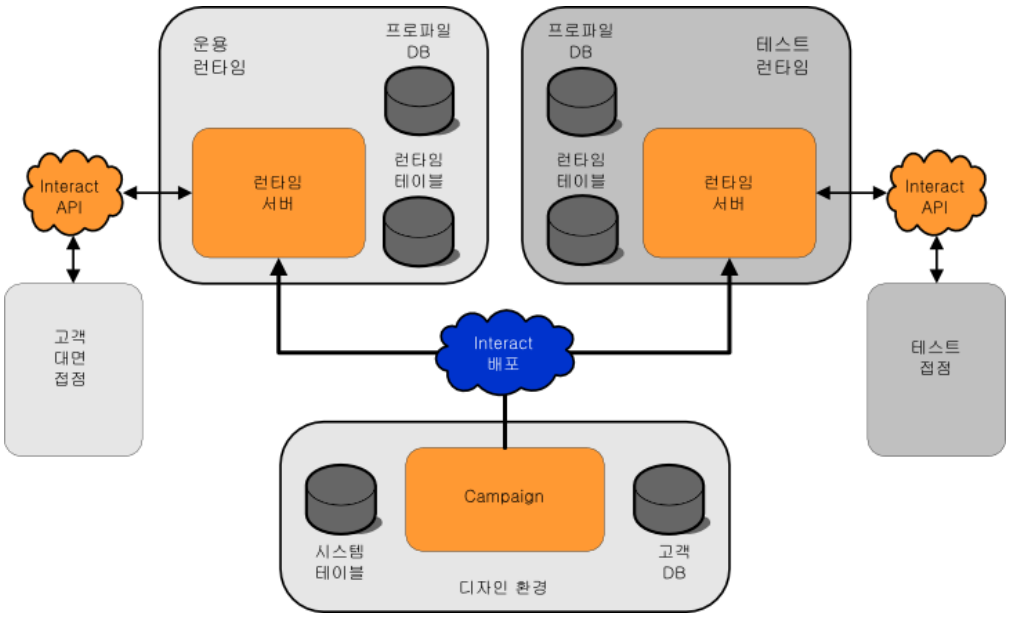
처리 규칙은 스마트 세그먼트에 오퍼를 지정합니다. 이러한 지정은 처리 규칙에서 오퍼와 연관시키는 사용자 정의 영역에 의해 추가로 제한됩니다. 예를 들어, "로그인" 영역에 스마트 세그먼트를 지정하는 오퍼 집합이 하나 있지만 "구매 후" 영역에 동일한 세그먼트에 대해 다른 오퍼 집합이 있습니다. 처리 규칙은 캠페인의 상호작용 전략 탭에 정의됩니다.

각 처리 규칙에는 마케팅 점수도 있습니다. 두 개 이상의 세그먼트에 고객이 할당되고 따라서 두 개 이상의 오퍼를 적용할 수 있는 경우, 마케팅 점수는 Interact가 권장하는

오퍼를 정의하도록 돕습니다. 런타임 환경이 권장하는 오퍼는 학습 모듈, 오퍼 제외 목록, 전역 및 개별 오퍼 지정의 영향을 받을 수 있습니다.

Interact 아키텍처

Interact 환경은 최소 두 가지 주요 구성 요소인 디자인 환경과 런타임 환경으로 이루어집니다. 선택적으로 테스트 런타임 서버를 구성할 수도 있습니다. 다음 그림은 상위 레벨 아키텍처 개요를 보여줍니다.



디자인 환경은 Interact 구성의 대부분을 수행하는 위치입니다. 디자인 환경은 Campaign 웹 응용 프로그램과 함께 설치되며 Campaign 시스템 테이블과 고객 데이터베이스를 참조합니다. 디자인 환경을 사용하여 API에 사용할 상호작용 지점 및 이벤트를 정의합니다.

런타임 환경에서 고객 상호작용을 처리할 방식을 디자인하고 구성한 후 스테이징 서버 그룹에 테스트 용도로 또는 실시간 고객 상호작용을 위해 운용 런타임 서버 그룹에 이 데이터를 배포합니다.

Interact API는 접점과 런타임 서버 간의 연결을 제공합니다. 사용자는 Interact API로 디자인 환경에 생성된 개체(접점과 이벤트)를 참조하고 이를 사용하여 런타임 서버의 정보를 요청합니다.

Interact 네트워크 고려사항

Interact의 운용 설치는 최소 두 시스템을 사용합니다. 여러 Interact 런타임 서버와 분산 데이터베이스가 있는 높은 볼륨의 운용 환경에서는 수십 개의 시스템이 사용될 수 있습니다. 최상의 성능을 위해 고려할 여러 네트워크 토폴로지 요구 사항이 있습니다.

- Interact API 구현이 동일한 호출로 세션을 시작하고 종료하는 경우(예:

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

  로드 밸런서와 Interact 런타임 서버 간의 세션 지속(스티키 세션)을 설정할 필요가 없습니다. 지역 캐시 유형에 맞는 Interact 런타임 서버 세션 관리를 구성할 수 있습니다.

- Interact API 구현이 여러 호출을 사용하여 세션을 시작하고 종료하며(예:

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

)Interact 런타임 서버의 로드 밸런서를 사용 중인 경우에는 로드 밸런서에 대한 일부 지속 유형(스티키 세션으로도 알려짐)을 설정해야 합니다. 가능하지 않거나 로드 밸런서를 사용하고 있지 않으면 배포된 cacheType에 맞는 Interact 서버 세션 관리를 구성하십시오. 배포된 캐시를 사용 중인 경우 모든 Interact 런타임 서버가 멀티캐스트를 통해 통신할 수 있어야 합니다. 동일한 멀티캐스트 IP 주소 및 포트를 사용한 Interact 서버 간의 통신이 시스템 성능을 저해하지 않도록 네트워크를 튜닝해야 할 수 있습니다. 스티키 세션이 설정된 로드 밸런서 사용이 배포된 캐시 사용보다 성능면에서 우수합니다.

- 배포된 cacheType을 사용하는 여러 서버 그룹이 있는 경우 각 그룹이 고유 멀티캐스트 포트를 사용해야 합니다. 각 서버마다 고유 멀티캐스트 포트와 주소를 사용하면 더 좋습니다.
- 최상의 성능을 위해서는 런타임 환경 Interact 서버, Marketing Platform, 로드 밸런서, 접점을 동일한 지리적 위치에 두십시오. 디자인 시간과 런타임은 다른 지리적 위치에 있을 수 있으며 이 경우 배포 속도가 느립니다.
- Interact 운용 서버 그룹 및 연관된 접점 간의 빠른 네트워크 연결(최소 1GB)을 유지하십시오.
- 디자인 시간에 배포 작업을 완료하려면 런타임에 대한 http 또는 https 액세스가 필요합니다. 배포를 허용하려면 방화벽이나 다른 네트워크 응용 프로그램을 구성해야 합니다. 대규모 배포인 경우 디자인 환경과 런타임 환경 간의 HTTP 제한시간 기간을 연장해야 할 수 있습니다.
- 컨택 및 응답 기록 모듈에 디자인 시간 데이터베이스(Campaign 시스템 테이블) 및 런타임 데이터베이스(Interact 런타임 테이블)에 대한 액세스가 필요합니다. 이 데이터 전송에 적합하게 데이터베이스와 네트워크를 구성해야 합니다.

테스트 또는 스테이징 설치에서는 Interact 디자인 시간 및 런타임을 동일한 시스템에 설치할 수 있습니다. 운용 환경에는 이 시나리오를 권장하지 않습니다.

제 2 장 IBM Unica Interact 사용자 구성

Interact에서는 두 가지 사용자 세트인 런타임 환경 사용자와 디자인 환경 사용자를 구성해야 합니다.

- **런타임 사용자**는 런타임 서버에 대해 작업하도록 구성된 Marketing Platform에 생성됩니다.
- **디자인 시간 사용자**는 Campaign 사용자입니다. Campaign에 대해 디자인 팀의 다양한 구성원에 대한 보안을 구성하십시오.

런타임 환경 사용자 구성

Interact를 설치한 후 최소 하나의 Interact 사용자, 런타임 환경 사용자를 구성해야 합니다.

런타임 환경 사용자는 런타임 테이블에 대한 액세스 권한을 제공합니다. 이는 대화식 채널을 배포할 때 사용하는 사용자 이름과 암호입니다. 런타임 서버는 웹 응용 프로그램 서버 JDBC 인증을 데이터베이스 신임 정보에 사용하므로 런타임 환경 데이터 소스를 런타임 환경 사용자에 추가할 필요가 없습니다.

중요사항: 같은 서버 그룹에 속하는 모든 런타임 서버는 동일한 사용자 신임 정보를 공유해야 합니다. 각 런타임 서버의 개별 Marketing Platform 인스턴스가 있는 경우에는 각각에 동일한 사용자와 암호를 생성해야 합니다.

데이터베이스 로드 유틸리티를 사용 중이면 런타임 환경에 대한 로그인 신임 정보의 데이터 소스로 런타임 테이블을 정의해야 합니다. 이 데이터 소스의 이름은 `systemTablesDataSource`여야 합니다.

JMXMP 프로토콜에 대한 JMX 모니터링에 보안을 설정한 경우 JMX 모니터링 보안을 위한 별도의 사용자가 필요합니다.

디자인 환경 사용자 구성

디자인 환경 사용자는 Campaign 사용자입니다. Campaign 역할 권한을 구성할 때와 동일한 방식으로 디자인 환경 사용자를 구성합니다.

대화식 플로차트를 편집할 권한이 있는 Campaign 사용자에게 테스트 실행 테이블 데이터 소스에 대한 액세스 권한을 부여해야 합니다.

© Copyright IBM Corp. 2001, 2012

Interact를 설치하고 구성한 경우에는 기본 전역 정책 및 새 정책에 대한 다음 추가 옵션이 사용 가능합니다. 일부 디자인 환경 사용자는 사용자 정의 매크로와 같은 몇 가지 Campaign 권한도 필요함을 기억하십시오.

범주	권한
캠페인	• 캠페인 상호작용 전략 보기 - 캠페인의 상호작용 전략 탭을 볼 수 있지만 편집할 수는 없습니다. • 캠페인 상호작용 전략 편집 - 처리 규칙을 포함한 상호작용 전략을 변경할 수 있습니다. • 캠페인 상호작용 전략 삭제 - 캠페인에서 상호작용 전략을 제거할 수 있습니다. 대화식 채널 배포에 상호작용 전략이 포함된 경우 상호작용 전략 탭 삭제가 제한됩니다. • 캠페인 상호작용 전략 추가 - 캠페인에 새 상호작용 전략 탭을 생성할 수 있습니다. • 캠페인 상호작용 전략 배포 시작 - 배포 또는 배포 제거에 대한 상호작용 전략 탭을 표시할 수 있습니다.
대화식 채널	• 대화식 채널 배포 - Interact 런타임 환경에 대화식 채널을 배포할 수 있습니다. • 대화식 채널 편집 - 대화식 채널의 요약 탭을 변경할 수 있습니다. • 대화식 채널 삭제 - 대화식 채널을 제거할 수 있습니다. 대화식 채널이 배포된 경우에는 대화식 채널 삭제가 제한됩니다. • 대화식 채널 보기 - 대화식 채널을 볼 수 있지만 편집할 수는 없습니다. • 대화식 채널 추가 - 새 대화식 채널을 생성할 수 있습니다. • 대화식 채널 보고서 보기 - 대화식 채널의 분석 탭을 볼 수 있습니다. • 대화식 채널 하위 개체 추가 - 상호작용 지점, 영역, 이벤트, 범주를 추가할 수 있습니다.
세션	• 대화식 플로차트 보기 - 세션의 대화식 플로차트를 볼 수 있습니다. • 대화식 플로차트 추가 - 세션에 새 대화식 플로차트를 생성할 수 있습니다. • 대화식 플로차트 편집 - 대화식 플로차트를 변경할 수 있습니다. • 대화식 플로차트 삭제 - 대화식 플로차트를 제거할 수 있습니다. 이 대화식 플로차트가 할당된 대화식 채널이 배포된 경우에는 대화식 플로차트 삭제가 제한됩니다. • 대화식 플로차트 복사 - 대화식 플로차트를 복사할 수 있습니다. • 대화식 플로차트 실행 테스트 - 대화식 플로차트의 테스트 실행을 시작할 수 있습니다. • 대화식 플로차트 검토 - 대화식 플로차트를 보고 프로세스를 열어 설정을 볼 수 있지만 변경할 수는 없습니다. • 대화식 플로차트 배포 - 배포 또는 배포 제거에 대해 대화식 플로차트를 표시할 수 있습니다.

예제 디자인 환경 권한

예를 들어, 대화식 플로차트를 생성하는 개인과 상호작용 전략을 정의하는 개인에 하나씩 두 개의 역할을 생성할 수 있습니다. 각 섹션에 권한 부여 역할이 나열됩니다.

대화식 플로차트 역할

사용자 정의 매크로

- 사용자 정의 매크로 추가
- 사용자 정의 매크로 편집
- 사용자 정의 매크로 사용

파생 필드

- 파생 필드 추가
- 파생 필드 편집
- 파생 필드 사용

플로차트 템플릿

- 템플릿 붙여넣기

세그먼트 템플릿

- 세그먼트 추가
- 세그먼트 편집

세션

- 세션 요약 보기
- 대화식 플로차트 보기
- 대화식 플로차트 추가
- 대화식 플로차트 편집
- 대화식 플로차트 복사
- 대화식 플로차트 실행 테스트
- 대화식 플로차트 배포

상호작용 전략 역할

캠페인

- 새 캠페인 요약
- 캠페인 대상 셀 관리
- 캠페인 상호작용 전략 보기

- 캠페인 상호작용 전략 편집
- 캠페인 상호작용 전략 추가
- 캠페인 상호작용 전략 배포 시작

오퍼

- 오퍼 요약 보기

세그먼트 템플릿

- 세그먼트 요약 보기

세션

- 대화식 플로차트 검토

제 3 장 Interact 데이터 소스 관리

Interact에서는 여러 데이터 소스가 제대로 기능해야 합니다. 일부 데이터 소스는 Interact가 기능하기 위해 필요한 정보를 포함하고 다른 데이터 소스를 사용자 데이터를 포함합니다.

다음 섹션은 Interact 데이터 소스를 제대로 구성하기 위해 필요한 정보와 데이터 소스 유지보수에 대한 몇 가지 제안사항을 포함하여 데이터 소스를 설명합니다.

Interact 데이터 소스에 대한 작업

Interact에서는 여러 데이터 세트가 기능해야 합니다.

- **Campaign 시스템 테이블** - Campaign의 모든 데이터 중에서 Campaign 시스템 테이블은 처리 규칙 및 대화식 채널과 같이 디자인 환경에서 생성하는 Interact 구성 요소에 대한 데이터를 포함합니다. 디자인 환경과 Campaign 시스템 테이블은 동일한 물리 데이터베이스 및 스키마를 공유합니다.
- **런타임 테이블** - (`systemTablesDataSource`) 디자인 환경, 컨택 및 응답 기록의 스테이징 테이블, 런타임 통계의 배포 데이터를 포함합니다.
- **프로파일 테이블** - (`prodUserDataSource`) 방문자를 스마트 세그먼트에 적절히 배치하기 위해 대화식 플로차트에 필요한 고객 데이터(실시간으로 수집된 정보 외에)를 포함합니다. 전적으로 실시간 데이터에 의존하는 경우에는 프로파일 테이블이 필요하지 않습니다. 프로파일 테이블을 사용 중이면 대화식 채널에 사용되는 대상 수준별로 최소 하나의 프로파일 테이블이 있어야 합니다.

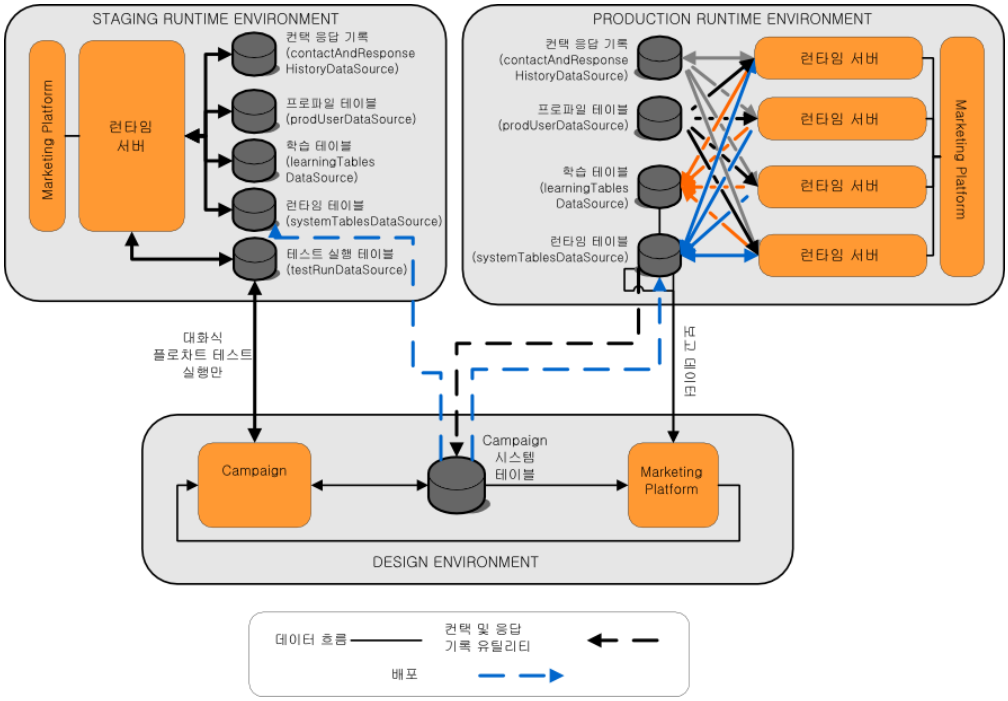
  프로파일 테이블은 오퍼 제외, 점수 재정의, 전역 및 개별 오퍼 지정에 대한 테이블을 포함하여 오퍼 제공을 기능 보강하는 데 사용되는 테이블도 포함할 수 있습니다.
- **테스트 실행 테이블** - (`testRunDataSource`) 상호작용 중 실시간 수집되는 정보를 모방하는 데이터를 포함하여, 방문자를 스마트 세그먼트에 배치하기 위해 대화식 플로차트에 필요한 모든 데이터의 샘플을 포함합니다. 이 테이블은 디자인 환경만의 테스트 실행 서버 그룹으로 지정된 서버 그룹에 필요합니다.
- **학습 테이블** - (`learningTablesDataSource`) 기본 제공 학습 유틸리티로 수집된 모든 데이터를 포함합니다. 이 테이블은 동적 속성을 정의하는 테이블을 포함할 수 있습니다. 학습을 사용하지 않거나 외부 학습 유틸리티를 사용하는 경우에는 학습 테이블이 필요하지 않습니다.

© Copyright IBM Corp. 2001, 2012

- **교차 세션 응답에 대한 컨택 및 응답 기록 -** (contactAndResponseHistoryDataSource) Campaign 컨택 기록 테이블이나 이 테이블의 사본입니다. 교차 세션 응답 기능을 사용하지 않으면 이 컨택 기록 테이블을 구성할 필요가 없습니다.

데이터베이스 및 응용 프로그램

다음 다이어그램은 가능한 Interact 데이터 소스 및 이 데이터 소스가 IBM® Unica 응용 프로그램에 관련된 방식을 보여줍니다.



- Campaign 및 테스트 실행 서버 그룹 모두 테스트 실행 테이블에 액세스합니다.
- 테스트 실행 테이블은 대화식 플로차트 테스트 실행에만 사용됩니다.
- 런타임 서버를 사용하여 Interact API를 포함한 배포를 테스트할 때 런타임 서버는 데이터의 프로파일 테이블을 사용합니다.
- 컨택 및 응답 기록 모듈을 구성한 경우 모듈은 백그라운드 추출, 변환, 로드(ETL) 프로세스를 사용하여 런타임 스테이징 테이블에서 Campaign 컨택 및 응답 기록 테이블로 데이터를 이동시킵니다.
- 보고 기능은 학습 테이블, 런타임 테이블, Campaign 시스템 테이블에서 데이터를 쿼리하여 Campaign에 보고서를 표시합니다.

운용 런타임 환경과 다른 테이블 세트를 사용하도록 테스트 런타임 환경을 구성해야 합니다. 스테이징과 운영 간 별도의 테이블로 실제 결과와 무관하게 테스트 결과를 보관할 수 있습니다. 컨택 및 응답 기록 모듈은 항상 실제 Campaign 컨택 및 응답 기록

테이블에 데이터를 삽입함에 유의하십시오(Campaign에는 테스트 컨택 및 응답 기록 테이블이 없음). 테스트 런타임 환경에 대한 별도의 학습 테이블이 있으며 보고서로 결과를 보려면 테스트 환경에 대한 학습 보고서를 실행하는 IBM Cognos® BI의 개별 인스턴스가 필요합니다.

Campaign 시스템 테이블

디자인 환경을 설치할 때에는 Campaign 시스템 테이블에 Interact별 테이블도 새로 생성합니다.

컨택 및 응답 기록 모듈을 설정하면 모듈이 런타임 테이블의 스테이징 테이블에서 Campaign 시스템 테이블의 컨택 및 응답 기록 테이블로 컨택 및 응답 기록을 복사합니다. 기본 테이블은 UA_ContactHistory, UA_DtlContactHist, UA_ResponseHistory이지만 컨택 및 응답 기록 모듈은 Campaign에 맵핑된 어느 테이블이나 컨택 및 응답 기록 테이블에 사용합니다.

전역 오퍼 테이블과 점수 재정의 테이블을 사용하여 오퍼를 할당하는 경우 대화식 채널의 처리 규칙에 포함되지 않은 오퍼를 사용 중이면 Campaign 시스템 테이블의 UACI_ICBatchOffers 테이블을 채워야 할 수 있습니다.

런타임 테이블

둘 이상의 대상 수준이 있는 경우 각 대상 수준마다 컨택 및 응답 기록 데이터에 대한 스테이징 테이블을 생성해야 합니다.

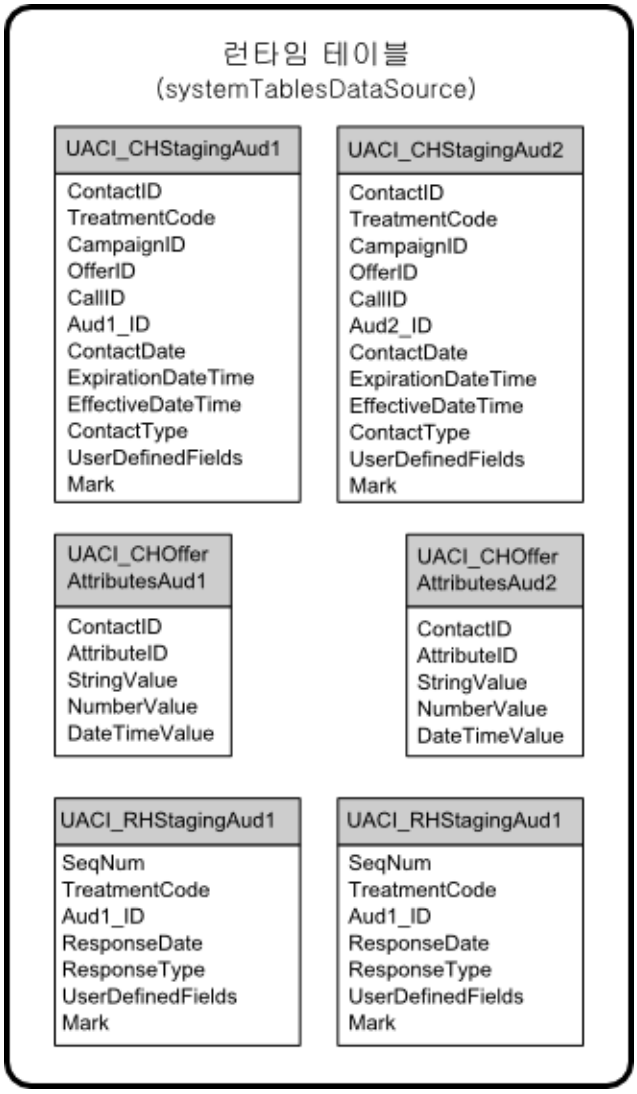
SQL 스크립트는 기본 대상 수준에 대한 다음 테이블을 생성합니다.

- UACI_CHStaging
- UACI_CHOfferAttrib
- UACI_RHStaging

런타임 테이블에 각 대상 수준에 대한 이 세 가지 테이블의 사본을 생성해야 합니다.

Campaign 컨택 및 응답 기록 테이블에 사용자 정의된 필드가 있으면 동일한 필드 이름과 유형을 UACI_CHStaging 및 UACI_RHStaging 테이블에 생성해야 합니다. 런타임 중 세션 데이터에 동일한 이름의 이름-값 쌍을 생성하여 이 필드를 채울 수 있습니다. 예를 들어, 컨택 및 응답 기록 테이블에는 catalogID 필드가 있습니다. catalogID 필드를 UACI_CHStaging 및 UACI_RHStaging 테이블에 모두 추가해야 합니다. 나중에 Interact API는 catalogID라는 이름-값 쌍으로 이벤트 매개변수를 정의해서 이 필드를 채웁니다. 세션 데이터는 프로파일 테이블, 임시 데이터, 학습 또는 Interact API로 제공할 수 있습니다.

다음 다이어그램은 Aud1 및 Aud2 대상의 샘플 테이블을 보여줍니다. 이 다이어그램에 런타임 데이터베이스의 모든 테이블이 포함되지는 않았습니다.



테이블의 모든 필드는 필수 필드입니다. `CustomerID` 및 `UserDefinedFields`를 Campaign 컨택 및 응답 기록 테이블과 일치하게 수정할 수 있습니다.

테스트 실행 테이블

테스트 실행 테이블은 대화식 플로차트의 테스트 실행에만 사용됩니다. 대화식 플로차트의 테스트 실행은 세그먼트 논리를 테스트해야 합니다. Interact 설치에 대한 하나의 테스트 실행 데이터베이스를 구성하기만 하면 됩니다. 테스트 실행 테이블은 독립형 데이터베이스에 있지 않아도 됩니다. 예를 들어, Campaign의 고객 데이터 테이블을 사용할 수 있습니다.

테스트 실행 테이블과 연관된 데이터베이스 사용자에게는 테스트 실행 결과 테이블을 추가할 CREATE 권한이 있어야 합니다.

테스트 실행 데이터베이스는 대화식 채널의 맵핑된 모든 테이블을 포함해야 합니다.

이 테이블은 대화식 플로차트에서 테스트하려는 시나리오를 실행할 데이터를 포함해야 합니다. 예를 들어, 대화식 플로차트에 음성 메일 시스템에 선택된 선택 사항에 기초하여 개인을 세그먼트로 정렬하는 논리가 있으면 가능한 모든 선택마다 최소 한 행이 있어야 합니다. 웹 사이트의 양식으로 작동하는 상호작용을 생성 중인 경우에는 누락되었거나 잘못된 양식의 데이터를 나타내는 행을 포함해야 합니다. 예를 들어, 이메일 주소의 값에 name@domaincom을 사용하십시오.

각 테스트 실행 테이블에는 해당 대상 수준에 대한 최소 하나의 ID 목록과 사용할 것으로 예상되는 실시간 데이터를 나타내는 열이 하나 있어야 합니다. 테스트 실행은 실시간 데이터에 대한 액세스가 없으므로 예상되는 실시간 데이터의 모든 부분에 대한 샘플 데이터를 제공해야 합니다. 예를 들어, lastPageVisited 속성에 저장된 방문한 마지막 웹 페이지의 이름이나 shoppingCartItemCount 속성에 저장된 장바구니의 항목 수와 같이 실시간으로 수집할 수 있는 데이터를 사용하려면 동일한 이름으로 열을 생성하고 샘플 데이터로 열을 채워야 합니다. 그러면 본래 동작이나 컨텍스트인 플로차트 논리의 분기를 테스트 실행할 수 있습니다.

대화식 플로차트의 테스트 실행은 큰 데이터 세트에 대한 작업에는 최적화되어 있지 않습니다. 상호작용 프로세스에서 테스트 실행에 사용되는 행 수를 제한할 수 있습니다. 하지만 제한하면 항상 첫 번째 행 세트가 선택됩니다. 다른 행 세트가 선택되게 하려면 테스트 실행 테이블의 다른 보기를 사용하십시오.

런타임에 대화식 플로차트의 처리량 성능을 테스트하려면 테스트 환경에 대한 프로파일 테이블을 포함하여 테스트 런타임 환경을 생성해야 합니다.

실제로, 대화식 플로차트의 테스트 실행에 대한 테스트 실행 테이블, 테스트 서버 그룹의 테스트 프로파일 테이블, 운용 프로파일 테이블 세트의 세 가지 테이블 세트가 테스트에 필요할 수 있습니다.

동적으로 생성된 테이블에 사용되는 기본 데이터 유형 재정의

Interact 런타임 환경은 두 가지 시나리오 즉, 플로차트의 테스트 실행 중 그리고 아직 존재하지 않는 테이블에 쓰는 스냅샷 프로세스의 실행 중 동적으로 테이블을 생성합니다. 이 테이블을 생성하기 위해 Interact는 지원되는 각 데이터베이스 유형의 하드 코딩된 데이터 유형에 의존합니다.

testRunDataSource 또는 prodUserDataSource에 uaci_column_types라는 대체 데이터 유형의 테이블을 생성하여 기본 데이터 유형을 재정의할 수 있습니다. Interact는 하드 코딩된 데이터 유형이 적용되지 않는 드문 경우에 이 추가 테이블을 사용합니다.

uaci_column_types 테이블이 정의될 때 Interact는 테이블 생성에 사용할 데이터 유형으로 열의 메타데이터를 사용합니다. uaci_column_types 테이블이 정의되지 않은 경우나 테이블을 읽으려 할 때 예외가 발생한 경우에는 기본 데이터 유형이 사용됩니다.

시작 시 런타임 시스템은 먼저 testRunDataSource에서 uaci_column_types 테이블을 확인합니다. uaci_column_types 테이블이 testDataSource에 존재하지 않는 경우나 prodUserDataSource가 다른 데이터베이스 유형이면 Interact는 prodUserDataSource에서 테이블을 확인합니다.

기본 데이터 유형 재정의

다음 단계에 따라 동적으로 생성된 테이블의 기본 데이터 유형을 재정의하십시오.

1. TestRunDataSource 또는 ProdUserDataSource에 다음 등록 정보의 테이블을 생성하십시오.

   테이블 이름: uaci_column_types

   열 이름:

   - uaci_float
   - uaci_number
   - uaci_datetime
   - uaci_string

   데이터베이스에 지원되는 적절한 데이터 유형을 사용하여 각 열을 정의하십시오.
2. Interact에 새 테이블이 인식되도록 런타임 서버를 다시 시작하십시오.

   중요사항: uaci_column_types 테이블을 변경할 때마다 런타임 서버를 다시 시작해야 합니다.

동적으로 생성된 테이블의 기본 데이터 유형

다음 테이블에는 Interact 런타임 시스템이 실수, 숫자, 날짜/시간, 문자열 열의 지원되는 각 데이터베이스에 기본적으로 사용하는 하드 코딩된 데이터 유형이 나열됩니다.

표 1. 동적으로 생성된 테이블의 기본 데이터 유형

데이터베이스	기본 데이터 유형
DB2®	• float • bigint • timestamp • varchar

표 1. 동적으로 생성된 테이블의 기본 데이터 유형 (계속)

데이터베이스	기본 데이터 유형
Informix®	• float • int8 • DATETIME YEAR TO FRACTION • char2
Oracle	• float • number(19) • timestamp • varchar2
SQL Server	• float • bigint • datetime • nvarchar

프로파일 데이터베이스

프로파일 데이터베이스의 컨텐츠는 대화식 플로차트 및 Interact API 구성에 필요한 데이터에 전적으로 의존합니다. Interact는 각 데이터베이스마다 특정 유형이나 데이터를 포함하도록 요구하거나 권장합니다.

프로파일 데이터베이스는 다음을 포함해야 합니다.

- 대화식 채널의 맵핑된 모든 테이블.

  이 테이블은 운용 환경에서 대화식 플로차트를 실행하는 데 필요한 모든 데이터를 포함합니다. 다음 테이블을 선택, 능률화, 제대로 색인화해야 합니다. 차원 데이터에 액세스하는 성능 비용이 있으므로 가능할 때마다 비정규화된 스키마를 사용해야 합니다. 최소한 대상 수준 ID 필드의 프로파일 테이블을 색인화해야 합니다. 차원 테이블에서 검색된 다른 필드가 있는 경우 데이터베이스 페치 시간을 줄이도록 적절하게 이 필드를 색인화해야 합니다. 프로파일 테이블의 대상 ID는 Campaign에 정의된 대상 ID와 일치해야 합니다.

- enableScoreOverrideLookup 구성 등록 정보를 true로 설정하면 최소 한 대상 수준에 대한 점수 재정의 테이블을 포함해야 합니다. 점수 재정의 테이블 이름은 scoreOverrideTable 등록 정보로 정의합니다.

  점수 재정의 테이블은 개별 고객 대 오퍼 쌍을 포함할 수 있습니다. 프로파일 데이터베이스에 대해 aci_usertab SQL 스크립트를 실행하여 샘플 점수 재정의 테이블, UACI_ScoreOverride를 생성할 수 있습니다. 대상 ID 열에도 이 테이블을 색인화해야 합니다.

enableScoreOverrideLookup 등록 정보를 false로 설정하는 경우에는 점수 재정의 테이블을 포함할 필요가 없습니다.

- enableDefaultOfferLookup 구성 등록 정보를 true로 설정하는 경우 전역 오퍼 테이블(UACI_DefaultOffers)을 포함해야 합니다. 프로파일 데이터베이스에 대해 aci_usertab SQL 스크립트를 실행하여 전역 오퍼 테이블을 생성할 수 있습니다.

  전역 오퍼 테이블은 대상 대 오퍼 쌍을 포함할 수 있습니다.

- enableOfferSuppressionLookup 등록 정보를 true로 설정하면 최소 하나의 대상 수준에 대한 오퍼 제외 테이블을 포함해야 합니다. 오퍼 제외 테이블 이름은 offerSuppressionTable 등록 정보로 정의합니다.

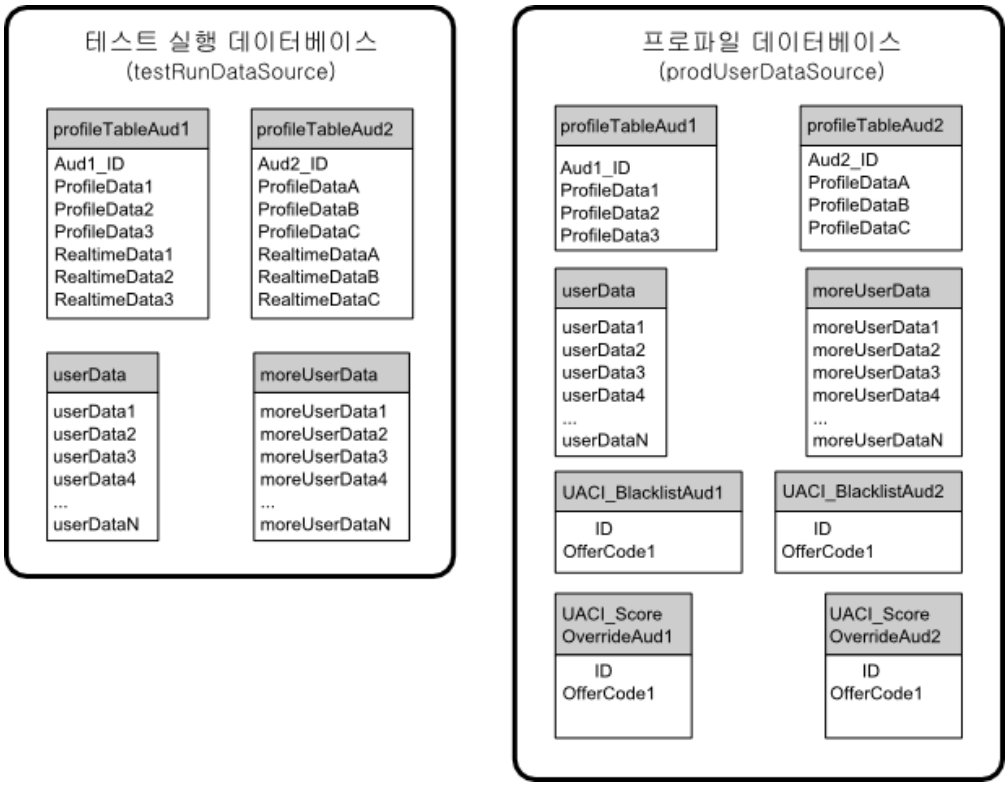
  오퍼 제외 테이블은 모든 대상 구성원의 항목이 필요하지 않아도 대상 구성원의 표시 안하는 각 오퍼에 대한 한 행을 포함할 수 있습니다. 프로파일 데이터베이스에 대해 aci_usertab SQL 스크립트를 실행하여 샘플 오퍼 제외 테이블, UACI_BlackList를 생성할 수 있습니다.

  enableOfferSuppressionLookup 등록 정보를 false로 설정하는 경우에는 오퍼 제외 테이블을 포함할 필요가 없습니다.

이 테이블에 대용량 데이터가 있는 경우 성능이 저하될 수 있습니다. 최상의 결과를 위해서는 런타임에 사용되는 대용량 데이터가 있는 테이블의 대상 수준 열에 적절한 색인을 두십시오.

위에서 참조된 모든 구성 등록 정보는 **Interact > 프로파일** 또는 **Interact > 프로파일 > 대상 수준 > *AudienceLevel*** 범주에 있습니다. aci_usertab SQL 스크립트는 런타임 환경 설치 디렉토리의 ddl 디렉토리에 있습니다.

다음 다이어그램은 대상 수준 Aud1 및 Aud2의 테스트 실행과 프로파일 데이터베이스에 대한 예 테이블을 보여줍니다.



학습 테이블

Interact 기본 제공 학습을 사용 중인 경우 학습 테이블을 구성해야 합니다. 이 테이블은 기본 제공 학습 기능으로 학습하는 모든 데이터를 포함합니다.

동적 학습 속성을 사용 중이면 UACI_AttributeList 테이블을 채워야 합니다.

학습은 중간 스테이징 테이블에 쓰고 스테이징 테이블에서 학습 테이블로 정보를 집계하는 과정과 관련됩니다. Interact > offerserving > 기본 제공 학습 구성 범주의 insertRawStatsIntervalInMinutes 및 aggregateStatsIntervalInMinutes 구성 등록 정보는 학습 테이블을 채우는 빈도를 판별합니다.

insertRawStatsIntervalInMinutes 속성은 각 고객 및 오퍼에 대한 수락과 컨택 정보가 메모리에서 스테이징 테이블, UACI_OfferStatsTX 및 UACI_OfferAllTx로 이동하는 빈도를 판별합니다. 스테이징 테이블에 저장된 정보는 aggregateStatsIntervalInMinutes 구성 등록 정보로 판별된 주기적 간격으로 집계되어 UACI_OfferStats 및 UACI_OfferStatsAll 테이블에 이동됩니다.

Interact 기본 제공 학습은 이 데이터를 사용하여 오퍼의 최종 점수를 계산합니다.

교차 세션 응답 추적의 컨택 기록

교차 세션 응답 기능을 사용하는 경우 런타임 환경에 Campaign 컨택 기록 테이블에 대한 읽기 전용 액세스 권한이 필요합니다. Campaign 시스템 테이블을 보도록 런타임 환경을 구성하거나 Campaign 컨택 기록 테이블의 사본을 생성할 수 있습니다. 테이블의 사본을 생성하면 사본을 최신 상태로 유지하는 프로세스를 관리해야 합니다. 컨택 및 응답 기록 모듈은 컨택 기록 테이블의 사본을 업데이트하지 않습니다.

이 컨택 기록 테이블에 대해 `aci_crhtab` SQL 스크립트를 실행하여 교차 세션 응답 추적 기능에 필요한 테이블을 추가해야 합니다.

Interact 기능 스크립트 사용

Interact에서 사용 가능한 몇 가지 선택적 기능을 사용하려면 프로파일 데이터베이스의 특정 테이블을 변경해야 합니다. Interact 설치는 디자인 환경과 런타임 환경 모두 ddl 스크립트를 포함합니다. 이 스크립트는 테이블에 필요한 특정 열을 추가합니다.

이러한 기능을 설정하려면 해당 데이터베이스 또는 테이블에 대해 해당 스크립트를 실행하십시오.

`dbType`은 데이터베이스 유형입니다(예: Microsoft SQL Server의 경우 `sqlsvr`).

기능 이름	기능 스크립트	실행 대상	변경
전역 오퍼, 오퍼 제외, 점수 재정의	런타임 환경 설치 디렉토리\ddl\aci_usrtab_*dbType*.sql	프로파일 데이터베이스 (userProdDataSource)	`DefaultOffers`, `UACI_BlackList`, `UACI_ScoreOverride` 테이블을 작성합니다.
점수 부여	런타임 환경 설치 디렉토리\ddl\acifeatures\aci_scoringfeature_*dbType*.sql	프로파일 데이터베이스의 점수 재정의 테이블 (userProdDataSource)	`LikelihoodScore` 및 `AdjExploreScore` 열을 추가합니다.
학습	디자인 환경 설치 디렉토리\ddl\acifeatures\aci_lrnfeature_*dbType*.sql	컨택 기록 테이블을 포함하는 Campaign 데이터베이스	`UA_DtlContactHist` 테이블에 `RTSelectionMethod` 열을 추가합니다.

컨택 및 응답 기록 추적 정보

컨택 및 응답 기록을 Campaign 컨택 및 응답 기록 테이블에 레코딩하도록 런타임 환경을 구성할 수 있습니다. 런타임 서버는 컨택 및 응답 기록을 스테이징 테이블에 저장합니다. 컨택 및 응답 기록 모듈은 이 데이터를 스테이징 테이블에서 Campaign 컨택 및 응답 기록 테이블로 복사합니다.

컨택 및 응답 기록 모듈은 디자인 환경의 구성 페이지에서 interactInstalled 및 contactAndResponseHistTracking > isEnabled 등록 정보를 예로 설정한 경우에만 작동합니다.

교차 세션 응답 추적 모듈을 사용 중인 경우 컨택 및 응답 기록 모듈은 별개의 엔티티입니다.

컨택 및 응답 유형 구성

다음 테이블에 표시된 대로 Interact에서 하나의 컨택 유형과 두 개의 응답 유형을 레코딩할 수 있습니다. 이 모든 등록 정보는 contactAndResponseHistTracking 범주에 있습니다.

이벤트	컨택/응답 유형	구성 등록 정보
오퍼 컨택 로그	컨택	연락된 고객 수
오퍼 수락 로그	응답	수락
오퍼 거부 로그	응답	거부

postEvent 메소드로 추가 사용자 정의 응답 유형을 레코딩할 수도 있습니다.

Campaign 시스템 테이블에 있는 UA_UsrResponseType 테이블의 CountsAsResponse 열이 제대로 구성되었는지도 확인해야 합니다. 이 모든 응답 유형이 UA_UsrResponseType 테이블에 존재해야 합니다.

UA_UsrResponseType의 올바른 항목이려면 CountsAsResponse를 포함하여 테이블의 모든 열의 값을 정의해야 합니다. CountsAsResponse의 올바른 값은 0, 1 또는 2입니다. 0은 응답 없음, 1은 응답, 2는 거부를 나타냅니다. 이 응답은 보고에 사용됩니다.

추가 응답 유형

Interact에서 Interact API의 postEvent 메소드를 사용하여 오퍼에 대한 "수락" 또는 "거부" 작업을 로그하는 이벤트를 트리거할 수 있습니다. postEvent 호출로 탐색, 고려, 커밋 또는 이행과 같은 추가 응답 유형을 레코딩하도록 시스템을 기능 보강할 수도 있습니다. 이 모든 응답 유형은 Campaign 시스템 테이블의 UA_UsrResponseType 테이블에 존재해야 합니다. postEvent 메소드에 특정 이벤트 매개변수를 사용하여 추가 응답 유형을 레코딩하고 학습에 수락을 포함해야 하는지 여부를 정의할 수 있습니다.

추가 응답 유형을 로그하려면 다음 이벤트 매개변수를 추가해야 합니다.

- **UACIRESPONSETYPECODE** - 응답 유형 코드를 나타내는 문자열입니다. 이 값은 UA_UsrResponseType 테이블의 올바른 항목이어야 합니다.

UA_UsrResponseType의 올바른 항목이려면 CountsAsResponse를 포함하여 테이블의 모든 열을 정의해야 합니다. CountsAsResponse의 올바른 값은 0, 1 또는 2입니다. 0은 응답 없음, 1은 응답, 2는 거부를 나타냅니다. 이 응답은 보고에 사용됩니다.

- **UACILOGTOLEARNING** - 1 또는 0 값이 있는 숫자입니다. 1은 Interact가 학습에 대한 수락으로 이벤트를 로그해야 함을 나타냅니다. 0은 Interact가 학습에 대한 이벤트를 로그하면 안됨을 나타냅니다. 이 매개변수를 사용하여 학습에 영향을 주지 않고 여러 응답 유형을 로깅하는 몇 개의 postEvent 메소드를 작성할 수 있습니다. UACILOGTOLEARNING을 정의하지 않으면 Interact는 기본값 0을 가정합니다.

로그하려는 각 응답 유형마다 하나씩 오퍼 수락 로그 작업으로 여러 이벤트를 생성하거나 개별 응답 유형을 로그하기 위해 사용하는 모든 postEvent 호출에 사용할 오퍼 수락 로그 작업으로 단일 이벤트를 생성할 수 있습니다.

예를 들어 각 응답 유형마다 오퍼 수락 로그 작업으로 하나의 이벤트를 생성하십시오. UA_UsrResponseType 테이블에 [이름(코드)]로 사용자 정의 응답을 정의합니다(예: 탐색(EXP), 고려(CON), 커밋(CMT)). 그런 다음 세 개의 이벤트를 생성하고 이름을 LogAccept_Explore, LogAccept_Consider, LogAccept_Commit으로 지정합니다. 세 가지 모든 이벤트는 정확히 동일하지만(오퍼 수락 로그 작업이 있음) API에 대해 작업하는 사용자가 구별할 수 있도록 이름은 서로 다릅니다.

또는 모든 사용자 정의 응답 유형에 사용하는 오퍼 수락 로그 작업으로 단일 이벤트를 생성할 수 있습니다. 예를 들어, 이름을 LogCustomResponse라 지정하십시오.

API에 대해 작업할 때에는 이벤트 간에 기능적 차이는 없지만 이름 지정 규칙으로 코드를 명확히 할 수 있습니다. 각 사용자 정의 응답에 별도의 이름을 부여하는 경우에는 채널 이벤트 활동 요약 보고서에 보다 정확한 정보가 표시됩니다.

먼저 모든 이름-값 쌍을 설정하십시오.

```
//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);
```

이 첫 번째 예제는 개별 이벤트 사용을 보여줍니다.

```
//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);
```

이 두 번째 예제는 이벤트를 하나만 사용하는 경우를 보여줍니다.

```
//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);
```

두 예제 모두 정확히 동일한 작업을 수행하지만 한 버전이 다른 버전에 비해 읽기 쉬울 수 있습니다.

런타임 환경 스테이징 테이블을 Campaign 기록 테이블에 맵핑

다음 테이블은 런타임 환경 스테이징 테이블을 Campaign 기록 테이블에 맵핑하는 방법을 표시합니다. 각 대상 수준마다 이 테이블 중 하나가 있어야 함을 기억하십시오. 표시된 테이블 이름은 런타임 테이블 및 Campaign 시스템 테이블에서 기본 대상에 대해 작성된 샘플 테이블입니다.

표 2. 컨택 기록

UACI_CHStaging **Interact** 컨택 기록 스테이징 테이블 열 이름	**Campaign** 컨택 기록 테이블	테이블 열 이름
ContactID	해당 없음	해당 없음
TreatmentCode	UA_Treatment	TreatmentCode
CampaignID	UA_Treatment	CampaignID
OfferID	UA_Treatment	OfferID
CellID	UA_Treatment	CellID
CustomerID	UA_DtlContactHist	CustomerID
ContactDate	UA_DtlContactHist	ContactDateTime
ExpirationDateTime	UA_Treatment	ExpirationDateTime
EffectiveDateTime	UA_Treatment	EffectiveDateTime
ContactType	UA_DtlContactHist	ContactStatusID
UserDefinedFields	UA_DtlContactHist	UserDefinedFields

ContactID는 UACI_CHOfferAtrib를 UACI_CHStaging과 조인하는 키입니다.

표 3. 오퍼 속성

UACI_CHOfferAttrib **Interact** 컨택 기록 스테이징 테이블 열 이름	**Campaign** 컨택 기록 테이블	테이블 열 이름
ContactID	해당 없음	해당 없음
AttributeID	UA_OfferHistAttrib	AttributeID
StringValue	UA_OfferHistAttrib	StringValue
NumberValue	UA_OfferHistAttrib	NumberValue
DateTimeValue	UA_OfferHistAttrib	DateTimeValue

ContactID는 UACI_CHOfferAtrib를 UACI_CHStaging과 조인하는 키입니다.

표 4. 응답 기록

UACI_RHStaging **Interact** 응답 기록 스테이징 테이블 열 이름	**Campaign** 응답 기록 테이블	테이블 열 이름
SeqNum	해당 없음	해당 없음
TreatmentCode	UA_ResponseHistory	TreatmentInstID
CustomerID	UA_ResponseHistory	CustomerID
ResponseDate	UA_ResponseHistory	ResponseDateTime
ResponseType	UA_ResponseHistory	ResponseTypeID
UserDefinedFields	UA_ResponseHistory	UserDefinedFields

SeqNum은 컨택 및 응답 기록 모듈이 데이터를 식별하는 데 사용하는 키이지만 Campaign 응답 테이블에 기록되지 않습니다.

userDefinedFields 열은 사용자가 선택하는 데이터를 포함할 수 있습니다. 스테이징 테이블에 열을 추가하면, 컨택 및 응답 기록 모듈이 UA_DtlContactHist 또는 UA_ResponseHistory 테이블의 동일 이름의 열에 기록합니다. 예를 들어, UACI_CHStaging 테이블에 linkFrom 열을 추가하면 컨택 및 응답 기록 모듈이 해당 데이터를 UA_DtlContactHist 테이블의 linkFrom 열로 복사합니다.

중요사항: Campaign 컨택 및 응답 기록 테이블에 추가 열이 있으면, 스테이징 테이블에 일치하는 열을 추가해야 컨택 및 응답 기록 테이블을 실행할 수 있습니다.

런타임 세션 데이터의 이름-값 쌍과 동일한 이름을 가진 열을 작성하여 스테이징 테이블의 기존 열을 채우십시오. 예를 들어, NumberItemsInWishList 및 NumberItemsInShoppingCart 이름-값 쌍을 작성하는 경우 오퍼 수락 로그 또는 오퍼 거부 로그 이벤트가 발생할 때 NumberItemsInWishList 및 NumberItemsInShoppingCart 열이 UACI_RHStaging 테이블에 있으면 런타임 환경이 해당 테이블을 채웁니다. 런타임 환경은 오퍼 컨택 로그 이벤트가 발생할 때 UACI_CHStaging 테이블을 채웁니다.

이러한 사용자 정의된 필드를 사용하여 오퍼를 제공하는 데 사용되는 점수를 포함시킬 수 있습니다. 런타임 테이블의 UACI_CHStaging 테이블과 Campaign 시스템 테이블의 UA_DtlContactHist 테이블 모두에 FinalScore 열을 추가하십시오. 기본 제공 학습을 사용 중인 경우 Interact가 FinalScore 열을 오퍼에 사용되는 최종 점수로 자동으로 채웁니다.

사용자 정의된 학습 모듈을 빌드하는 경우, ITreatment 인터페이스의 setActualValueUsed 메소드 및 ILearning 인터페이스의 logEvent 메소드를 사용할 수 있습니다.

학습을 사용하지 않는 경우, 런타임 테이블의 UACI_CHStaging 테이블과 Campaign 시스템 테이블의 UA_DtlContactHist 테이블 모두에 Score 열을 추가하십시오. Interact가 Score 열을 오퍼에 사용되는 점수로 자동으로 채웁니다.

컨택 및 응답 기록 모듈에 대한 JMX 모니터링 구성

디자인 환경에 대한 Marketing Platform에서 Campaign > 모니터링 범주의 다음 구성 등록 정보를 편집하십시오.

구성 등록 정보	설정
monitorEnabledForInteract	**True**
port	JMX 서비스의 포트 번호입니다.
protocol	**JMXMP** 또는 **RMI** JMXMP 프로토콜을 선택하더라도 컨택 및 응답 기록 모듈에 대한 보안이 사용되지 않습니다.

JMX 모니터링 도구에서 컨택 및 응답 기록 데이터를 보면 속성이 처음에는 파티션별로 그리고 그 다음에는 대상 수준별로 구성되어 있습니다.

JMXMP 프로토콜에 대한 컨택 및 응답 기록 모듈 모니터링의 기본 주소는 service:jmx:jmxmp://*CampaignServer*:*port*/campaign입니다.

RMI 프로토콜에 대한 컨택 및 응답 기록 모듈 모니터링의 기본 주소는 service:jmx:rmi:///jndi/rmi://*CampaignServer*:*port*/campaign입니다.

교차 세션 응답 추적 정보

방문자는 항상 한 번의 접촉점 방문으로 트랜잭션을 완료할 수는 없습니다. 고객이 웹 사이트에서 장바구니에 항목을 추가하고 2일 후까지 판매를 완료하지 않을 수 있습니다. 런타임 세션을 무한정 활성화하는 것은 실현 불가능합니다. 교차 세션 응답 추적을 사용하여 한 세션에서는 오퍼 프리젠테이션을 추적하고 다른 세션에서는 이를 응답과 일치시킬 수 있습니다.

Interact 교차 세션 응답 추적은 기본적으로 처리 코드 또는 오퍼 코드를 일치시킬 수 있습니다. 또한 사용자가 선택한 사용자 정의 코드를 일치시키도록 구성할 수 있습니다. 교차 세션 응답은 사용 가능한 데이터를 일치시킵니다. 예를 들어, 웹 사이트에 한 주 동안 할인 상품을 표시할 때 생성된 판촉 코드가 있는 오퍼가 포함되어 있습니다. 사용자가 장바구니에 항목을 추가하지만 3일 후까지 구매를 완료하지 않을 수 있습니다. postEvent 호출을 사용하여 수락 이벤트를 로그하면, 판촉 코드만 포함시킬 수 있습니다. 런타임은 현재 세션에서 일치시킬 처리 또는 오퍼 코드를 찾을 수 없기 때문에 런타임은 교차 세션(XSessResponse) 스테이징 테이블에 사용 가능한 정보와 함께 수락 이벤트를 배치합니다. CrossSessionResponse 서비스는 XSessResponse 테이블을 정기적으로 읽고 해당 레코드를 사용 가능한 컨택 기록 데이터와 일치시키려고 시도합니다. CrossSessionResponse 서비스는 판촉 코드를 컨택 기록과 일치시키고 적절한 응답을 로그하는 데 필요한 모든 데이터를 수집합니다. 그런 다음 CrossSessionResponse 서비스는 응답을 응답 스테이징 테이블 및 학습 테이블(학습이 활성화된 경우)에 기록합니다. 그러면 컨택 및 응답 기록 모듈이 응답을 Campaign 컨택 및 응답 기록 테이블에 기록합니다.

교차 세션 응답 추적 데이터 소스 구성

Interact 교차 세션 응답 추적은 런타임 환경의 세션 데이터를 Campaign 컨택 및 응답 기록과 일치시킵니다. 기본적으로 교차 세션 응답 추적은 처리 코드 또는 오퍼 코드를 일치시킵니다. 사용자 정의 또는 대체 코드를 일치시키도록 런타임 환경을 구성할 수 있습니다.

- 대체 코드를 일치시키도록 선택하면, Interact 런타임 테이블의 UACI_TrackingType 테이블에 정의해야 합니다.

- 런타임 환경은 Campaign 컨택 기록 테이블에 액세스할 수 있어야 합니다. Campaign 컨택 기록 테이블에 액세스할 수 있도록 런타임 환경을 구성하거나 런타임 환경에 컨택 기록 테이블 복사본을 작성하면 이렇게 할 수 있습니다.

  이 액세스는 읽기 전용이며, 컨택 및 응답 기록 유틸리티와 별개입니다.

  테이블 복사본을 작성하는 경우, 컨택 기록 복사본의 데이터가 정확한지 확인하는 것은 사용자의 책임입니다. purgeOrphanResponseThresholdInMinutes 등록 정보를 사용하여 CrossSessionResponse 서비스가 일치하지 않는 응답을 보유하는 시간 길이를 사용자가 컨택 기록 테이블 복사본의 데이터 새로 고침 빈도와 일치하도록 구성할 수 있습니다. 컨택 및 응답 기록 모듈을 사용 중인 경우, ETL 업데이트를 조정하여 최신 데이터를 보유하고 있는지 확인해야 합니다.

교차 세션 응답 추적을 위한 컨택 및 응답 기록 테이블 구성

컨택 기록 테이블 복사본을 작성하건 Campaign 시스템 테이블의 실제 테이블을 사용하건 다음 단계를 수행해야 합니다.

1. 컨택 및 응답 기록 테이블을 Campaign에서 올바로 맵핑해야 합니다.
2. Interact 디자인 환경 설치 디렉토리의 interactDT/ddl/acifeatures 디렉토리에 있는 aci_lrnfeature SQL 스크립트를 Campaign 시스템 테이블의 UA_DtlContactHist 및 UA_ResponseHistory 테이블에 대해 실행해야 합니다.

   이 스크립트는 UA_DtlContactHist 및 UA_ResponseHistory 테이블에 RTSelectionMethod 열을 추가합니다. 각 대상 수준마다 이러한 테이블에 대해 aci_lrnfeature 스크립트를 실행하십시오. 각 대상 수준마다 올바른 테이블에 대해 작업하도록 이 스크립트를 필요에 따라 편집하십시오.
3. 컨택 기록 테이블을 런타임 환경으로 복사하려면 지금 복사하십시오.
4. Interact 런타임 환경 설치 디렉토리의 ddl 디렉토리에 있는 aci_crhtab SQL 스크립트를 컨택 및 응답 기록 데이터 소스에 대해 실행하십시오.

   이 스크립트는 UACI_XsessResponse 및 UACI_CRHTAB_Ver 테이블을 작성합니다.
5. 각 대상 수준마다 UACI_XsessResponse 테이블의 버전을 작성하십시오.

교차 세션 응답 추적 지원을 위해 런타임 환경이 액세스할 수 있는 Campaign 컨택 기록 테이블 복사본을 작성 중인 경우, 다음 지침을 사용하십시오.

- 교차 세션 응답 추적에는 이러한 테이블에 대한 읽기 전용 액세스가 필요합니다.
- 교차 세션 응답 추적에는 Campaign 컨택 기록의 다음 테이블이 필요합니다.
  - UA_DtlContactHist(각 대상 수준마다)
  - UA_Treatment

이러한 테이블의 데이터를 정기적으로 업데이트하여 정확한 응답 추적을 보장해야 합니다.

교차 세션 응답 추적의 성능을 향상시키기 위해 컨택 기록 데이터를 복사하거나 Campaign 컨택 기록 테이블에 보기를 구성하여 컨택 기록 데이터의 양을 제한할 수 있습니다. 예를 들어, 오퍼가 30일 동안만 유효한 비즈니스 사례가 있으면 컨택 기록 데이터를 지난 30일로 제한해야 합니다.

컨택 및 응답 기록 모듈이 실행되어야 교차 세션 응답 추적의 결과를 볼 수 있습니다. 예를 들어, 기본 processSleepIntervalInMinutes가 60분입니다. 따라서 최소 1시간이 지나야 교차 세션 응답이 Campaign 응답 기록에 나타납니다.

UACI_TrackingType 테이블

UACI_TrackingType 테이블은 런타임 환경 테이블의 일부입니다. 이 테이블은 교차 세션 응답 추적과 함께 사용되는 추적 코드를 정의합니다. 추적 코드는 런타임 환경이 런타임 세션의 현재 오퍼를 컨택 및 응답 기록과 일치시키는 데 사용하는 메소드를 정의합니다.

열	유형	설명
TrackingCodeType	int	추적 코드 유형을 나타내는 숫자입니다. SQL 명령은 이 숫자를 참조하여 세션 데이터의 정보를 컨택 및 응답 기록 테이블과 일치시킵니다.
Name	varchar(64)	추적 코드 유형의 이름입니다. UACI_TrackingCodeType 예약 매개변수를 postEvent 메소드와 함께 사용하여 세션 데이터로 전달됩니다.
Description	varchar(512)	추적 코드 유형에 대한 간단한 설명입니다. 이 필드는 선택사항입니다.

다음 테이블에 표시된 대로 기본적으로 런타임 환경에는 두 개의 추적 코드가 정의되어 있습니다. 대체 코드의 경우, 고유 TrackingCodeType을 정의해야 합니다.

TrackingCodeType	이름	설명
1	처리 코드	UACI 생성 처리 코드
2	오퍼 코드	UAC 캠페인/오퍼 코드

UACI_XSessResponse

각 대상 수준마다 이 테이블의 한 인스턴스가 Interact 교차 세션 응답 추적에 사용 가능한 컨택 및 응답 기록 데이터 소스에 있어야 합니다.

열	유형	설명
SeqNumber	bigint	데이터 행 ID입니다. CrossSessionResponse 서비스는 SeqNumber 순서로 모든 레코드를 처리합니다.

열	유형	설명
ICID	bigint	대화식 채널 ID입니다.
AudienceID	bigint	이 대상 수준의 대상 ID입니다. 이 열의 이름은 Campaign에 정의된 대상 ID와 일치해야 합니다. 샘플 테이블은 CustomerID 열을 포함합니다.
TrackingCode	varchar(64)	postEvent 메소드의 UACIOfferTrackingCode 매개변수가 전달한 값입니다.
TrackingCodeType	int	추적 코드의 숫자 표시입니다. 이 값은 UACI_TrackingType 테이블의 올바른 항목이어야 합니다.
OfferID	bigint	Campaign에 정의된 오퍼 ID입니다.
ResponseType	int	이 레코드의 응답 유형입니다. 이 값은 UA_UsrResponseType 테이블의 올바른 항목이어야 합니다.
ResponseTypeCode	varchar(64)	이 레코드의 응답 유형 코드입니다. 이 값은 UA_UsrResponseType 테이블의 올바른 항목이어야 합니다.
ResponseDate	datetime	응답 날짜입니다.
Mark	bigint	이 필드 값은 레코드 상태를 식별합니다. • 1 - 진행 중 • 2 - 성공 • NULL - 재시도 • -1 - 레코드가 purgeOrphanResponseThresholdInMinutes분이 넘도록 데이터베이스에 있었습니다. 데이터베이스 관리자의 이 테이블 유지보수의 일부로 이 필드에서 일치하지 않는 레코드(즉, 값이 -1인 모든 레코드)를 확인할 수 있습니다. 값이 2인 모든 레코드는 CrossSessionResponse 서비스에 의해 자동으로 제거됩니다.
UsrDefinedFields	char(18)	오퍼 응답을 컨택 및 응답 기록과 일치시킬 때 포함시킬 사용자 정의 필드입니다. 예를 들어, 판촉 코드를 일치시키려면 판촉 코드 사용자 정의된 필드를 포함시키십시오.

교차 세션 응답 추적 사용

교차 세션 응답 추적을 최대한 활용하려면 컨택 및 응답 기록 모듈을 구성해야 합니다.

교차 세션 응답 추적을 사용하려면 Campaign 컨택 및 응답 기록 테이블에 대한 읽기 액세스 권한을 갖도록 런타임 환경을 구성해야 합니다. 디자인 환경의 실제 Campaign 컨택 및 응답 기록 테이블 또는 런타임 환경 데이터 소스의 테이블 복사본에서 읽을 수 있습니다. 이는 컨택 및 응답 기록 모듈 구성과 별개입니다.

처리 코드 또는 오퍼 코드가 아닌 어떤 것을 일치시키는 경우, UACI_TrackingType 테이블에 추가해야 합니다.

1. 런타임 환경에 액세스할 수 있는 컨택 및 응답 기록 테이블에 XSessResponse 테이블을 작성하십시오.

2. 런타임 환경의 contactAndResponseHistoryDataSource 범주에 등록 정보를 정의하십시오.
3. 각 대상 수준마다 crossSessionResponseTable 등록 정보를 정의하십시오.
4. 각 대상 수준마다 OverridePerAudience 범주를 작성하십시오.

교차 세션 응답 오퍼 일치

기본적으로 교차 세션 응답 추적은 처리 코드 또는 오퍼 코드를 일치시킵니다. crossSessionResponse 서비스는 SQL 명령을 사용하여 세션 데이터의 처리 코드, 오퍼 코드 또는 사용자 정의 코드를 Campaign 컨택 및 응답 기록 테이블과 일치시킵니다. 이 SQL 명령을 편집하여 추적 코드, 오퍼 코드 또는 사용자 정의 코드 사용자 정의를 일치시킬 수 있습니다.

처리 코드별 일치

처리 코드별 일치를 위한 SQL은 이 대상 수준에 대한 XSessResponse 테이블의 모든 열과 OfferIDMatch 열을 리턴해야 합니다. OfferIDMatch 열의 값은 XSessResponse 레코드의 처리 코드와 함께 제공되는 offerId여야 합니다.

다음은 처리 코드를 일치시키는 기본적으로 생성된 SQL 명령 샘플입니다. Interact는 대상 수준에 올바른 테이블 이름을 사용하기 위한 SQL을 생성합니다. 이 SQL은 Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byTreatmentCode > SQL 등록 정보가 **시스템 생성 SQL 사용**으로 설정된 경우 사용됩니다.

```
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
```

UACI_XsessResponse, UA_DtlContactHist, CustomerID, UA_ContactHistory 값은 사용자 설정에 의해 Interact에 정의됩니다. 예를 들어, UACI_XsessResponse는 Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable 구성 등록 정보에 의해 정의됩니다.

컨택 및 응답 기록 테이블을 사용자 정의한 경우, 테이블에 대한 작업을 하려면 이 SQL을 수정해야 합니다. Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byTreatmentCode > OverrideSQL 등록 정보에 SQL 재정의를 정의하십시오. 일부 재정의 SQL을 제공하는 경우, SQL 등록 정보도 **재정의 SQL**로 변경해야 합니다.

오퍼 코드별 일치

오퍼 코드별 일치를 위한 SQL은 이 대상 수준에 대한 XSessResponse 테이블의 모든 열과 TreatmentCodeMatch 열을 리턴해야 합니다. TreatmentCodeMatch 열의 값은 XSessResponse 레코드의 오퍼 ID(및 오퍼 코드)와 함께 제공되는 처리 코드여야 합니다.

다음은 오퍼 코드를 일치시키는 기본적으로 생성된 SQL 명령 샘플입니다. Interact는 대상 수준에 올바른 테이블 이름을 사용하기 위한 SQL을 생성합니다. 이 SQL은 Interact > services > crossSessionResponse > OverridePerAudience > *AudienceLevel* > TrackingCodes > byOfferCode > SQL 등록 정보가 **시스템 생성 SQL 사용**으로 설정된 경우 사용됩니다.

```
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
```

UACI_XsessResponse, UA_DtlContactHist, CustomerID, UA_ContactHistory 값은 사용자 설정에 의해 Interact에 정의됩니다. 예를 들어, UACI_XsessResponse는 Interact > profile > Audience Levels > [AudienceLevelName] > crossSessionResponseTable 구성 등록 정보에 의해 정의됩니다.

컨택 및 응답 기록 테이블을 사용자 정의한 경우, 테이블에 대한 작업을 하려면 이 SQL을 수정해야 합니다. Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byOfferCode > OverrideSQL 등록 정보에 SQL 재정의를 정의하십시오. 일부 재정의 SQL을 제공하는 경우, SQL 등록 정보도 **재정의 SQL**로 변경해야 합니다.

대체 코드별 일치

사용자가 선택한 일부 대체 코드별 일치를 위한 SQL 명령을 정의할 수 있습니다. 예를 들어, 오퍼 또는 처리 코드와 별개인 판촉 코드 또는 제품 코드가 있을 수 있습니다.

Interact 런타임 환경 테이블의 UACI_TrackingType 테이블에 이 대체 코드를 정의해야 합니다.

Interact > services > crossSessionResponse > OverridePerAudience > *(AudienceLevel)* > TrackingCodes > byAlternateCode > OverrideSQL 등록 정보에 이 대상 수준에 대한 XSessResponse 테이블의 모든 열과 TreatmentCodeMatch 및 OfferIDMatch 열을 리턴하는 SQL 또는 스토어드 프로시저를 제공해야 합니다. OfferIDMatch 대신에 offerCode를 선택적으로 리턴할 수 있습니다(N 파트 오퍼 코드의 경우 offerCode1, offerCode2, ... offerCode*N* 형식). TreatmentCodeMatch 열 및 OfferIDMatch 열(또는 오퍼 코드 열)의 값은 XSessResponse 레코드의 TrackingCode에 해당해야 합니다.

예를 들어, 다음 SQL 의사 코드는 XSessResponse 테이블의 AlternateCode 열을 일치시킵니다.

```
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>
```

여기서 <*x*>는 UACI_TrackingType 테이블에 정의된 추적 코드입니다.

런타임 환경에서 데이터베이스 로드 유틸리티 사용

기본적으로 런타임 환경은 세션 데이터의 컨택 및 응답 기록 데이터를 스테이징 테이블에 기록합니다. 그러나 매우 활성화된 운용 시스템에서는 런타임이 스테이징 테이블에 기록하기 전에 데이터를 모두 캐시하는 데 필요한 메모리 양이 엄청나게 많습니다. 성능 향상을 위해 데이터베이스 로드 유틸리티를 사용하도록 런타임을 구성할 수 있습니다.

데이터베이스 로드 유틸리티를 사용하면, 런타임이 스테이징 테이블에 기록하기 전에 컨택 및 응답 기록을 메모리에 모두 보유하지 않고 스테이징 테이블에 데이터를 기록합니다. `externalLoaderStagingDirectory` 등록 정보를 사용하여 스테이징 파일이 포함된 디렉토리의 위치를 정의하십시오. 이 디렉토리에는 몇 개의 하위 디렉토리가 있습니다. 첫 번째 하위 디렉토리는 `contactHist` 및 `respHist` 디렉토리를 포함하는 런타임 인스턴스 디렉토리입니다. `contactHist` 및 `respHist` 디렉토리에는 *audienceLevelName.uniqueID.currentState* 형식으로 고유하게 이름 지정된 하위 디렉토리(스테이징 파일을 포함)가 있습니다.

현재 상태	설명
CACHE	디렉토리 컨텐츠를 파일에 현재 기록 중입니다.
READY	디렉토리 컨텐츠를 처리할 준비가 되었습니다.
RUN	디렉토리 컨텐츠를 데이터베이스에 현재 기록 중입니다.
PROCESSED	디렉토리 컨텐츠가 데이터베이스에 기록되었습니다.
ERROR	디렉토리 컨텐츠를 데이터베이스에 기록하는 동안 오류가 발생했습니다.
ATTN	디렉토리 컨텐츠에 주의가 필요합니다. 즉, 데이터베이스에 이 디렉토리 컨텐츠 기록을 완료하려면 몇 가지 수동 단계를 수행해야 합니다.
RERUN	디렉토리 컨텐츠를 데이터베이스에 기록할 준비가 되었습니다. 문제점을 정정한 후에는 디렉토리 이름을 `ATTN` 또는 `ERROR`에서 `RERUN`으로 변경해야 합니다.

응용 프로그램 서버 시작 스크립트에 interact.runtime.instance.name JVM 등록 정보를 정의하여 런타임 인스턴스 디렉토리를 정의할 수 있습니다. 예를 들어, 웹 응용 프로그램 서버 시작 스크립트에 `-Dinteract.runtime.instance.name=instance2`를 추가할 수 있습니다. 설정하지 않은 경우 기본 이름은 `DefaultInteractRuntimeInstance`입니다.

`samples` 디렉토리에는 자체 데이터베이스 로드 유틸리티 컨트롤 파일 작성을 지원하는 샘플 파일이 들어 있습니다.

런타임 환경에 데이터베이스 로드 유틸리티 설정

데이터베이스 로드 유틸리티의 명령이나 컨트롤 파일을 사용하도록 런타임 환경을 구성하려면 먼저 이를 정의해야 합니다. 이 파일은 같은 서버 그룹에서 모든 런타임 서버의 동일한 위치에 있어야 합니다.

Interact는 Interact 런타임 서버 설치의 loaderService 디렉토리에 샘플 명령과 컨트롤 파일을 제공합니다.

1. 런타임 환경 사용자에게 Marketing Platform에 정의된 런타임 테이블 데이터 소스에 대한 로그인 신임 정보가 있는지 확인하십시오.

   Marketing Platform의 데이터 소스 이름은 systemTablesDataSource여야 합니다.

2. Interact > general > systemTablesDataSource > loaderProperties 구성 등록 정보를 정의하십시오.
3. Interact > 서비스 >externalLoaderStagingDirectory 등록 정보를 정의하십시오.
4. 필요에 따라 Interact > 서비스 > responseHist > fileCache 구성 등록 정보를 수정하십시오.
5. 필요에 따라 Interact > 서비스 > contactHist > fileCache 구성 등록 정보를 수정하십시오.
6. 런타임 서버를 다시 시작하십시오.

제 4 장 오퍼 제공

제공할 오퍼 선택 방법을 향상시키기 위해 여러 가지 방법으로 Interact를 선택할 수 있습니다. 다음 섹션에서는 이러한 선택적 기능을 자세히 설명합니다.

오퍼 자격

Interact의 목적은 적합한 오퍼를 제공하는 것입니다. 간단히 말해 Interact는 방문자, 채널, 상황을 기준으로 적합한 오퍼 중에서 최적 오퍼를 제공합니다.

처리 규칙은 Interact가 고객에게 적합한 오퍼를 판별하는 방법의 시작일 뿐입니다. Interact에는 런타임이 환경이 제공할 오퍼를 판별하는 방법을 향상시키기 위해 구현할 수 있는 몇 가지 선택적 기능이 있습니다. 이 중 어떠한 기능도 고객에게 오퍼가 제공되도록 보장하지 않습니다. 이러한 기능은 오퍼가 고객에게 제공하기에 적합하다는 가능성에 영향을 줍니다. 이러한 기능을 필요한 만큼 얼마든지 또는 일부를 사용하여 환경에 맞는 최상의 솔루션을 구현할 수 있습니다.

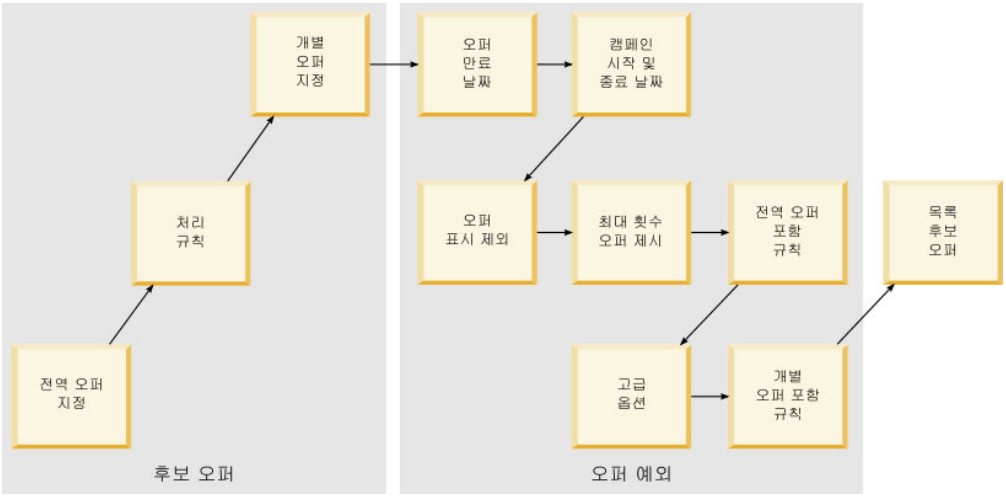
오퍼 자격에 영향을 줄 수 있는 세 가지 기본 영역이 있는데, 후보 오퍼 목록 생성, 마케팅 점수 판별, 학습입니다.

후보 오퍼 목록 생성

후보 오퍼 목록 생성에는 두 개의 주요 스테이지가 있습니다. 첫 번째 스테이지는 고객에게 적합한 모든 가능한 오퍼 목록을 생성하는 것입니다. 두 번째 스테이지는 고객에게 더 이상 적합하지 않은 모든 오퍼를 필터링하는 것입니다. 두 스테이지 모두에서 후보 오퍼 목록 생성에 영향을 줄 수 있는 몇 개의 위치가 있습니다.

다음 다이어그램은 후보 오퍼 목록 생성 스테이지를 표시합니다. 화살표는 우선 순위 순서를 표시합니다. 예를 들어, 오퍼가 **최대 오퍼 제공 횟수** 필터를 통과하지만 **전역 오퍼 포함 규칙** 필터를 통과하지 못하면 런타임 환경이 오퍼를 제외합니다.

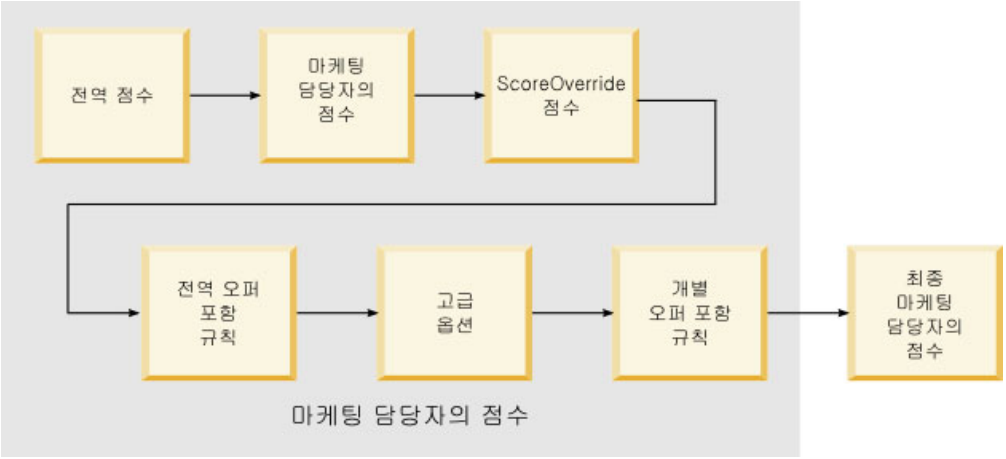
© Copyright IBM Corp. 2001, 2012



- **전역 오퍼 지정** - 전역 오퍼 테이블을 사용하여 대상 수준별 전역 오퍼를 정의할 수 있습니다.
- **처리 규칙** - 상호작용 전략 탭을 사용하여 상호작용 지점별로 세그먼트별 오퍼를 정의하는 기본 방법입니다.
- **개별 오퍼 지정** - 점수 재정의 테이블을 사용하여 고객별 특정 오퍼 지정을 정의할 수 있습니다.
- **오퍼 만료 날짜** - Campaign에서 오퍼를 작성할 때 만료 날짜를 정의할 수 있습니다. 오퍼 만료 날짜가 지났으면 런타임 환경이 오퍼를 제외합니다.
- **캠페인 시작 및 종료 날짜** - Campaign에서 캠페인을 작성할 때 캠페인 시작 및 종료 날짜를 정의할 수 있습니다. 캠페인 시작 날짜가 발생하지 않았거나 캠페인 종료 날짜가 지났으면 런타임 환경이 오퍼를 제외합니다.
- **오퍼 제외** - 오퍼 제외 테이블을 사용하여 특정 대상 구성원에 대한 오퍼 제외를 정의할 수 있습니다.
- **최대 오퍼 제공 횟수** - 대화식 채널을 정의할 때 세션당 고객에게 오퍼를 제공할 최대 횟수를 정의하십시오. 오퍼가 이미 이 횟수만큼 제공되었으면 런타임 환경이 오퍼를 제외합니다.
- **전역 오퍼 포함 규칙** - 전역 오퍼 테이블을 사용하여 대상 수준에서 오퍼를 필터하기 위한 부울 표현식을 정의할 수 있습니다. 결과가 거짓이면 런타임 환경이 오퍼를 제외합니다.
- **고급 옵션** - 처리 규칙의 **다음 표현식이 참이면 이 규칙을 적합한 것으로 간주** 고급 옵션을 사용하여 세그먼트 수준에서 오퍼를 필터할 수 있습니다. 결과가 거짓이면 런타임 환경이 오퍼를 제외합니다.
- **개별 오퍼 포함 규칙** - 점수 재정의 테이블을 사용하여 고객 수준에서 오퍼를 필터하기 위한 부울 표현식을 정의할 수 있습니다. 결과가 거짓이면 런타임 환경이 오퍼를 제외합니다.

마케팅 점수 계산

마케팅 점수에 영향을 주거나(계산을 사용) 재정의하는 여러 가지 방법이 있습니다. 다음 다이어그램은 마케팅 점수에 영향을 주거나 재정의할 수 있는 여러 스테이지를 표시합니다. 화살표는 우선 순위 순서를 표시합니다. 예를 들어, 처리 규칙에 대한 고급 옵션에 마케팅 점수를 판별하기 위한 표현식을 정의하고 점수 재정의 테이블에 표현식을 정의하는 경우 점수 재정의 테이블의 표현식이 우선합니다.



- **전역 점수** - 전역 오퍼 테이블을 사용하여 대상 수준당 점수를 정의할 수 있습니다.
- **마케팅 담당자의 점수** - 처리 규칙의 슬라이더를 사용하여 세그먼트당 점수를 정의할 수 있습니다.
- **점수 재정의 점수** - 점수 재정의 테이블을 사용하여 고객당 점수를 정의할 수 있습니다.
- **전역 오퍼 포함 점수** - 전역 오퍼 테이블을 사용하여 대상 수준당 점수를 계산하는 표현식을 정의할 수 있습니다.
- **고급 옵션** - 처리 규칙의 **다음 표현식을 마케팅 점수로 사용** 고급 옵션을 사용하여 세그먼트당 점수를 계산하는 표현식을 정의할 수 있습니다.
- **점수 재정의 오퍼 포함 규칙** - 점수 재정의 테이블을 사용하여 고객당 점수를 계산하는 표현식을 정의할 수 있습니다.

학습에 영향력 행사

Interact 기본 제공 학습 모듈을 사용 중인 경우, 학습 속성 또는 신뢰 수준 목록과 같은 표준 학습 구성을 벗어나 학습 출력에 영향을 줄 수 있습니다. 나머지 구성 요소를 사용하는 동안 학습 알고리즘 구성 요소를 재정의할 수 있습니다.

기본 오퍼 및 점수 재정의 테이블의 `LikelihoodScore` 및 `AdjExploreScore` 열을 사용하여 학습을 재정의할 수 있습니다. `aci_scoringfeature` 기능 스크립트를 사용하여 기본 오퍼 및 점수 재정의 테이블에 이 열을 추가할 수 있습니다. 이러한 재정의를 올바로 사용하려면 Interact 기본 제공 학습을 철저히 이해해야 합니다.

학습 모듈은 후보 오퍼 목록 및 후보 오퍼당 마케팅 점수를 사용하며 최종 계산에서 이를 사용합니다. 오퍼 목록은 고객이 오퍼를 수락할 가능성(수락 가능성)을 계산하는 데 학습 속성과 함께 사용됩니다. 학습 알고리즘은 탐색과 이용 간의 균형을 맞추기 위해 이러한 가능성과 프리젠테이션 기록 수를 사용하여 오퍼 가중치를 판별합니다. 마지막으로, 기본 제공 학습은 오퍼 가중치를 사용하고 최종 마케팅 점수를 곱해 최종 점수를 리턴합니다. 오퍼는 이 최종 점수별로 정렬됩니다.

오퍼 억제 정보

런타임 환경이 오퍼를 억제하는 몇 가지 방법이 있습니다.

- 대화식 채널의 **1회 방문 시 오퍼를 표시할 최대 횟수** 요소.

  대화식 채널을 작성하거나 편집할 때 **1회 방문 시 오퍼를 표시할 최대 횟수**를 정의하십시오.
- 오퍼 제외 테이블 사용.

  프로파일 데이터베이스에 오퍼 제외 테이블을 작성할 수 있습니다.
- 만료 날짜가 지난 오퍼.
- 만료된 캠페인의 오퍼.
- 오퍼 포함 규칙(처리 규칙 고급 옵션)을 통과하지 못하기 때문에 제외된 오퍼.
- Interact 세션에서 명시적으로 이미 수락되거나 거부된 오퍼. 고객이 오퍼를 명시적으로 수락하거나 거부하면, 해당 오퍼가 세션 기간 동안 억제됩니다.

오퍼 제외 테이블 사용

억제된 오퍼 목록을 참조하도록 Interact를 구성할 수 있습니다.

1. 모든 대상에 대해 대상 ID 및 오퍼 ID를 포함하는 새 테이블(`offerSuppressionTable`)을 작성하십시오.
2. `enableOfferSuppressionLookup` 등록 정보를 **true**로 설정하십시오.
3. `offerSuppressionTable` 등록 정보를 해당 대상에 대한 오퍼 제외 테이블 이름으로 설정하십시오.

오퍼 제외 테이블

오퍼 제외 테이블을 사용하여 특정 대상 ID에 대한 오퍼를 표시하지 않을 수 있습니다. 예를 들어, 대상이 Customer이면 John Smith 고객에 대한 오퍼를 표시하지 않을 수 있습니다. 한 개 이상의 대상 수준에 대해 이 테이블의 버전이 운용 프로파일 데이터베이스에 있어야 합니다. 프로파일 데이터베이스에 대해 `aci_usertab` SQL 스크립

트를 실행하여 오퍼 제외 테이블 샘플(UACI_Blacklist)을 작성할 수 있습니다. aci_usertab SQL 스크립트는 런타임 환경 설치 디렉토리의 ddl 디렉토리에 있습니다.

각 행에 대해 AudienceID 및 OfferCode1 필드를 정의해야 합니다. 대상 ID 또는 오퍼 코드가 여러 열로 구성된 경우 추가 열을 추가할 수 있습니다. 이 열은 Campaign에 정의된 열 이름과 일치해야 합니다. 예를 들어, HHold_ID 및 MemberNum 필드로 Customer 대상을 정의하는 경우 오퍼 제외 테이블에 HHold_ID 및 MemberNum을 추가해야 합니다.

이름	설명
AudienceID	(필수) 이 열의 이름은 Campaign에 대상 ID를 정의하는 열 이름과 일치해야 합니다. 대상 ID가 여러 열로 구성된 경우, 이 테이블에 열을 추가할 수 있습니다. 각 행은 기본 오퍼를 지정할 대상 ID(예: customer1)를 포함해야 합니다.
OfferCode1	(필수) 재정의할 오퍼의 오퍼 코드입니다. 오퍼 코드가 여러 필드로 구성된 경우, 추가 열(예: OfferCode2 등)을 추가할 수 있습니다.

전역 오퍼 및 개별 지정

상호작용 전략 탭에 구성된 처리 규칙을 벗어나 특정 오퍼를 지정하도록 런타임 환경을 구성할 수 있습니다. 임의의 대상 수준 구성원에 대해 전역 오퍼를 정의하고 특정 대상 구성원에 대해 개별 지정을 정의할 수 있습니다. 예를 들어, 모든 가정에 대해 전역 오퍼를 정의하여 다른 사람이 사용할 수 없는 시기를 확인한 후 특정 Smith 가정에 대해 개별 오퍼 지정을 작성할 수 있습니다.

영역, 셀, 오퍼 포함 규칙을 통해 전역 오퍼와 개별 지정을 모두 제한할 수 있습니다. 전역 오퍼와 개별 지정 모두 운용 프로파일 데이터베이스의 특정 테이블에 데이터를 추가하여 구성됩니다.

전역 오퍼와 개별 지정이 올바로 작동하려면 참조된 모든 셀 및 오퍼 코드가 배포에 있어야 합니다. 필수 데이터를 사용할 수 있게 하려면 기본 셀 코드 및 UACI_ICBatchOffers 테이블을 구성해야 합니다.

기본 셀 코드 정의

전역 또는 개별 오퍼 지정에 기본 오퍼 또는 점수 재정의 테이블을 사용하는 경우, IndividualTreatment 범주의 각 대상 수준 및 테이블 유형에 대해 DefaultCellCode 등록 정보를 정의하여 기본 셀 코드를 정의해야 합니다.

기본 오퍼 또는 점수 재정의 테이블의 특정 행에 셀 코드를 정의하지 않은 경우 DefaultCellCode가 사용됩니다. 보고는 이 기본 셀 코드를 사용합니다.

DefaultCellCode는 Campaign에 정의된 셀 코드 형식과 일치해야 합니다. 이 셀 코드는 보고에 나타나는 모든 오퍼 지정에 사용됩니다. 고유 기본 셀 코드를 정의하는 경우, 기본 오퍼 또는 정의 테이블에서 지정한 오퍼를 쉽게 식별할 수 있습니다.

UACI_ICBatchOffers 테이블 정의

기본 오퍼 또는 점수 재정의 테이블을 사용하는 경우, 모든 오퍼 코드가 배포에 있는지 확인해야 합니다. 기본 오퍼 또는 점수 재정의 테이블에서 사용하는 모든 오퍼가 처리 규칙에서 사용된다고 알고 있으면, 오퍼가 배포에 있습니다. 그러나 처리 규칙에서 사용되지 않는 모든 오퍼는 UACI_ICBatchOffers 테이블에 정의해야 합니다.

UACI_ICBatchOffers 테이블은 Campaign 시스템 테이블에 있습니다.

UACI_ICBatchOffers 테이블을 기본 오퍼 또는 점수 재정의 테이블에서 사용되는 오퍼 코드로 채워야 합니다. 이 테이블의 형식은 다음과 같습니다.

열 이름	유형	설명
ICName	varchar(64)	오퍼가 연관된 대화식 채널 이름입니다. 두 개의 서로 다른 대화식 채널에서 동일한 오퍼를 사용 중인 경우, 각 대화식 채널에 행을 제공해야 합니다.
OfferCode1	varchar(64)	오퍼 코드의 첫 번째 파트입니다.
OfferCode2	varchar(64)	오퍼 코드의 두 번째 파트입니다(필요한 경우).
OfferCode3	varchar(64)	오퍼 코드의 세 번째 파트입니다(필요한 경우).
OfferCode4	varchar(64)	오퍼 코드의 네 번째 파트입니다(필요한 경우).
OfferCode5	varchar(64)	오퍼 코드의 다섯 번째 파트입니다(필요한 경우).

전역 오퍼 테이블 정보

전역 오퍼 테이블을 사용하여 대상 수준에서 처리를 정의할 수 있습니다. 예를 들어, 대상 가정의 모든 구성원에 대한 전역 오퍼를 정의할 수 있습니다.

Interact 오퍼 제공의 다음 요소에 대한 전역 설정을 정의할 수 있습니다.

- 전역 오퍼 지정
- 전역 마케팅 담당자의 점수(숫자 기준 또는 표현식 기준)
- 오퍼를 필터할 부울 표현식
- 학습 가능성 및 가중치(Interact 기본 제공 학습을 사용 중인 경우)
- 전역 학습 재정의

전역 오퍼 테이블 사용

처리 규칙에 정의된 것을 벗어나 대상 수준에 대해 전역 오퍼를 지정하도록 런타임 환경을 구성할 수 있습니다.

1. 프로파일 데이터베이스에 UACI_DefaultOffers 테이블을 작성하십시오.

aci_usrtab ddl 파일을 사용하여 올바른 열을 가진 UACI_DefaultOffers 테이블을 작성할 수 있습니다.

2. enableDefaultOfferLookup 등록 정보를 **true**로 설정하십시오.

전역 오퍼 테이블

전역 오퍼 테이블은 프로파일 데이터베이스에 있어야 합니다. 프로파일 데이터베이스에 대해 aci_usertab SQL 스크립트를 실행하여 UACI_DefaultOffers 전역 오퍼 테이블을 작성할 수 있습니다. aci_usertab SQL 스크립트는 런타임 환경 설치 디렉토리의 ddl 디렉토리에 있습니다.

각 행에 대해 AudienceLevel 및 OfferCode1 필드를 정의해야 합니다. 기타 필드는 오퍼 지정을 추가로 제한하거나 대상 수준에서 기본 제공 학습에 영향을 주기 위한 선택 사항입니다.

최상의 성능을 얻으려면 이 테이블의 대상 수준 열에 인덱스를 작성해야 합니다.

이름	유형	설명
AudienceLevel	varchar(64)	(필수) 기본 오퍼를 지정하는 대상 수준의 이름입니다(예: customer 또는 household). 이 이름은 Campaign에 정의된 대상 수준과 일치해야 합니다.
OfferCode1	varchar(64)	(필수) 기본 오퍼의 오퍼 코드입니다. 오퍼 코드가 여러 필드로 구성된 경우, 추가 열(예: OfferCode2 등)을 추가할 수 있습니다. 전역 오퍼 지정을 제공하기 위해 이 오퍼를 추가하는 경우, UACI_ICBatchOffers 테이블에 이 오퍼를 추가해야 합니다.
Score	float	이 오퍼 지정에 대한 마케팅 점수를 정의할 숫자입니다.
OverrideTypeID	int	1로 설정된 경우, 오퍼가 오퍼 후보 목록에 없으면 목록에 이 오퍼를 추가하고 오퍼에 대한 점수 데이터를 사용하십시오. 일반적으로 전역 오퍼 지정을 제공하려면 1을 사용하십시오. 0, *null* 또는 1이 아닌 임의의 숫자로 설정된 경우, 오퍼가 오퍼 후보 목록에 있는 경우에만 오퍼에 대한 데이터를 사용하십시오. 대부분의 경우, 처리 규칙 또는 개별 지정은 이 설정을 재정의합니다.

이름	유형	설명
Predicate	varchar(4000)	처리 규칙에 대한 고급 옵션으로 이 열에 표현식을 입력할 수 있습니다. 처리 규칙에 대한 고급 옵션 작성 시 사용 가능한 동일한 변수 및 매크로를 사용할 수 있습니다. 이 열의 동작은 EnableStateID 열의 값에 따라 다릅니다. • EnableStateID가 2이면, 이 열은 처리 규칙에 대한 고급 옵션의 **다음 표현식이 참이면 이 규칙을 적합한 것으로 간주** 옵션과 동일하게 작동하여 이 오퍼 지정을 제한합니다. 이 열은 부울 표현식을 포함해야 하며, 이 오퍼를 포함하려면 참으로 해석되어야 합니다. 잘못해서 숫자로 해석되는 표현식을 정의하면, 0이 아닌 임의의 숫자가 참으로 간주되고 0은 거짓으로 간주됩니다. • EnableStateID가 3이면, 이 열은 처리 규칙에 대한 고급 옵션의 **다음 표현식을 마케팅 점수로 사용** 옵션과 동일하게 작동하여 이 오퍼를 제한합니다. 이 열은 숫자로 해석되는 표현식을 포함해야 합니다. • EnableStateID가 1이면, Interact가 이 열의 모든 값을 무시합니다.
FinalScore	float	반환된 오퍼 최종 목록을 정렬하는 데 사용되는 최종 점수를 재정의할 숫자입니다. 이 열은 기본 제공 학습 모듈을 활성화한 경우 사용됩니다. 이 열을 사용하도록 자체 학습을 구현할 수 있습니다.
CellCode	varchar(64)	이 기본 오퍼를 지정할 대화식 세그먼트에 대한 셀 코드입니다. 셀 코드가 여러 필드로 구성된 경우, 추가 열을 추가할 수 있습니다. OverrideTypeID가 0 또는 null인 경우 셀 코드를 제공해야 합니다. 셀 코드를 포함시키지 않으면, 런타임 환경이 이 데이터 행을 무시합니다. OverrideTypeID가 1이면, 이 열에 셀 코드를 제공하지 않아도 됩니다. 셀 코드를 제공하지 않는 경우, 런타임 환경은 보고 목적으로 이 대상 수준 및 테이블에 대해 DefaultCellCode 등록 정보에 정의된 셀 코드를 사용합니다.
Zone	varchar(64)	이 오퍼 지정을 적용할 영역 이름입니다. NULL이면 모든 영역에 적용됩니다.

이름	유형	설명
EnableStateID	int	이 열의 값은 Predicate 열의 동작을 정의합니다. • **1** - Predicate 열을 사용하지 않습니다. • **2** - Predicate를 부울로 사용하여 오퍼를 필터합니다. 처리 규칙의 **다음 표현식이 참이면 이 규칙을 적합한 것으로 간주** 고급 옵션과 동일한 규칙을 따릅니다. • **3** - Predicate를 사용하여 마케팅 담당자의 점수를 정의합니다. 처리 규칙의 **다음 표현식을 마케팅 점수로 사용** 고급 옵션과 동일한 규칙을 따릅니다. 이 열이 Null이거나 2 또는 3 이외의 값인 모든 행은 Predicate 열을 무시합니다.
LikelihoodScore	float	이 열은 기본 제공 학습에 영향을 주기 위해서만 사용됩니다. aci_scoringfeature ddl을 사용하여 이 열을 추가할 수 있습니다.
AdjExploreScore	float	이 열은 기본 제공 학습에 영향을 주기 위해서만 사용됩니다. aci_scoringfeature ddl을 사용하여 이 열을 추가할 수 있습니다.

점수 재정의 정보

점수 재정의 테이블을 사용하여 대상 ID 또는 개별 수준에서 처리를 정의할 수 있습니다. 예를 들어, 대상 수준이 방문자이면 특정 방문자에 대한 재정의를 작성할 수 있습니다.

Interact 오퍼 제공의 다음 요소에 대한 재정의를 정의할 수 있습니다.

- 개별 오퍼 지정
- 개별 마케팅 담당자의 점수(숫자 기준 또는 표현식 기준)
- 오퍼를 필터할 부울 표현식
- 학습 가능성 및 가중치(기본 제공 학습을 사용 중인 경우)
- 개별 학습 재정의

점수 재정의 테이블 사용

마케팅 점수 대신 모델링 응용 프로그램에서 생성된 점수를 사용하도록 Interact를 구성할 수 있습니다.

1. 재정의를 제공할 각 대상 수준에 대해 점수 재정의 테이블을 작성하십시오.

   aci_usrtab ddl 파일을 사용하여 올바른 열을 가진 점수 재정의 테이블 샘플을 작성할 수 있습니다.

2. enableScoreOverrideLookup 등록 정보를 **true**로 설정하십시오.

3. scoreOverrideTable 등록 정보를 재정의를 제공할 각 대상 수준의 점수 재정의 테이블 이름으로 설정하십시오.

   모든 대상 수준에 점수 재정의 테이블을 제공하지는 않아도 됩니다.

점수 재정의 테이블

점수 재정의 테이블은 운용 프로파일 데이터베이스에 있어야 합니다. 프로파일 데이터베이스에 대해 aci_usertab SQL 스크립트를 실행하여 점수 재정의 테이블 샘플(UACI_ScoreOverride)을 작성할 수 있습니다. aci_usertab SQL 스크립트는 런타임 환경 설치 디렉토리의 ddl 디렉토리에 있습니다.

각 행에 대해 *AudienceID*, OfferCode1 및 Score 필드를 정의해야 합니다. 기타 필드의 값은 개별 오퍼 지정을 추가로 제한하거나 기본 제공 학습에 대한 점수 재정의 정보를 제공하기 위한 선택 사항입니다.

이름	유형	설명
AudienceID	varchar(64)	(필수) 이 열의 이름은 Campaign에 대상 ID를 정의하는 열 이름과 일치해야 합니다. aci_usertab ddl 파일이 작성한 샘플 테이블은 이 열을 CustomerID 열로 작성합니다. 대상 ID가 여러 열로 구성된 경우, 이 테이블에 열을 추가할 수 있습니다. 각 행은 개별 오퍼를 지정할 대상 ID(예: customer1)를 포함해야 합니다. 최상의 성능을 얻으려면 이 열에 인덱스를 작성해야 합니다.
OfferCode1	varchar(64)	(필수) 오퍼의 오퍼 코드입니다. 오퍼 코드가 여러 필드로 구성된 경우, 추가 열(예: OfferCode2 등)을 추가할 수 있습니다. 개별 오퍼 지정을 제공하기 위해 이 오퍼를 추가하는 경우, UACI_ICBatchOffers 테이블에 이 오퍼를 추가해야 합니다.
Score	float	이 오퍼 지정에 대한 마케팅 점수를 정의할 숫자입니다.
OverrideTypeID	int	0이나 *null*(또는 1이 아닌 임의의 숫자)로 설정된 경우, 오퍼가 오퍼 후보 목록에 있는 경우에만 오퍼에 대한 데이터를 사용하십시오. 일반적으로 점수 재정의를 제공하려면 0을 사용하십시오. 셀 코드를 제공해야 합니다. 1로 설정된 경우, 오퍼가 오퍼 후보 목록에 없으면 목록에 이 오퍼를 추가하고 오퍼에 대한 점수 데이터를 사용하십시오. 일반적으로 개별 오퍼 지정을 제공하려면 1을 사용하십시오.

이름	유형	설명
Predicate	varchar(4000)	처리 규칙에 대한 고급 옵션으로 이 열에 표현식을 입력할 수 있습니다. 처리 규칙에 대한 고급 옵션 작성 시 사용 가능한 동일한 변수 및 매크로를 사용할 수 있습니다. 이 열의 동작은 EnableStateID 열의 값에 따라 다릅니다. • EnableStateID가 2이면, 이 열은 처리 규칙에 대한 고급 옵션의 **다음 표현식이 참이면 이 규칙을 적합한 것으로 간주** 옵션과 동일하게 작동하여 이 오퍼 지정을 제한합니다. 이 열은 부울 표현식을 포함해야 하며, 이 오퍼를 포함하려면 참으로 해석되어야 합니다. 잘못해서 숫자로 해석되는 표현식을 정의하면, 0이 아닌 임의의 숫자가 참으로 간주되고 0은 거짓으로 간주됩니다. • EnableStateID가 3이면, 이 열은 처리 규칙에 대한 고급 옵션의 **다음 표현식을 마케팅 점수로 사용** 옵션과 동일하게 작동하여 이 오퍼를 제한합니다. 이 열은 숫자로 해석되는 표현식을 포함해야 합니다. • EnableStateID가 1이면, Interact가 이 열의 모든 값을 무시합니다.
FinalScore	float	반환된 오퍼 최종 목록을 정렬하는 데 사용되는 최종 점수를 재정의할 숫자입니다. 이 열은 기본 제공 학습 모듈을 활성화한 경우 사용됩니다. 이 열을 사용하도록 자체 학습을 구현할 수 있습니다.
CellCode	varchar(64)	이 오퍼를 지정할 대화식 세그먼트에 대한 셀 코드입니다. 셀 코드가 여러 필드로 구성된 경우, 추가 열을 추가할 수 있습니다. OverrideTypeID가 0 또는 null인 경우 셀 코드를 제공해야 합니다. 셀 코드를 포함시키지 않으면, 런타임 환경이 이 데이터 행을 무시합니다. OverrideTypeID가 1이면, 이 열에 셀 코드를 제공하지 않아도 됩니다. 셀 코드를 제공하지 않는 경우, 런타임 환경은 보고 목적으로 이 대상 수준 및 테이블에 대해 DefaultCellCode 등록 정보에 정의된 셀 코드를 사용합니다.
Zone	varchar(64)	이 오퍼 지정을 적용할 영역 이름입니다. NULL이면 모든 영역에 적용됩니다.

이름	유형	설명
EnableStateID	int	이 열의 값은 Predicate 열의 동작을 정의합니다. • **1** - Predicate 열을 사용하지 않습니다. • **2** - Predicate를 부울로 사용하여 오퍼를 필터합니다. 처리 규칙의 **다음 표현식이 참이면 이 규칙을 적합한 것으로 간주** 고급 옵션과 동일한 규칙을 따릅니다. • **3** - Predicate를 사용하여 마케팅 담당자의 점수를 정의합니다. 처리 규칙의 **다음 표현식을 마케팅 점수로 사용** 고급 옵션과 동일한 규칙을 따릅니다. 이 열이 Null이거나 2 또는 3 이외의 값인 모든 행은 Predicate 열을 무시합니다.
LikelihoodScore	float	이 열은 기본 제공 학습에 영향을 주기 위해서만 사용됩니다. aci_scoringfeature ddl을 사용하여 이 열을 추가할 수 있습니다.
AdjExploreScore	float	이 열은 기본 제공 학습에 영향을 주기 위해서만 사용됩니다. aci_scoringfeature ddl을 사용하여 이 열을 추가할 수 있습니다.

Interact 기본 제공 학습 개요

올바른 세그먼트에 올바른 오퍼를 제안하기 위해 할 수 있는 모든 것을 수행하는 동안 방문자의 실제 선택 사항에서 항상 뭔가를 학습할 수 있습니다. 방문자의 실제 동작은 전략에 영향을 주어야 합니다. 응답 기록을 가져와 일부 모델링 도구를 통해 실행하여 대화식 플로차트에 포함시킬 수 있는 점수를 가져올 수 있습니다. 그러나 이 데이터는 실시간이 아닙니다.

Interact는 방문자의 작업에서 실시간으로 학습할 수 있도록 다음 두 가지 옵션을 제공합니다.

- 기본 제공 학습 모듈 - 런타임 환경에는 Naive Bayesian 기반 학습 모듈이 있습니다. 이 모듈은 사용자가 선택하는 고객 속성을 모니터하고 해당 데이터를 사용하여 제공할 오퍼를 쉽게 선택합니다.
- 학습 API - 런타임 환경에는 자체 학습 모듈을 작성할 수 있도록 학습 API도 있습니다.

학습을 사용하지 않아도 됩니다. 기본적으로 학습은 비활성화되어 있습니다.

Interact 학습 이해

Interact 학습 모듈은 오퍼 및 방문자 속성에 대한 방문자의 응답을 모니터합니다. 학습 모듈에는 다음 두 가지 일반 모드가 있습니다.

- **탐색** - 학습 모듈은 이용 중에 나중에 사용되는 추정을 최적화하기에 충분한 응답 데이터를 수집할 수 있도록 오퍼를 제공합니다. 탐색 중에 제공된 오퍼는 최적 선택을 반드시 반영하지는 않습니다.
- **이용** - 탐색 단계에서 충분한 데이터가 수집된 후 학습 모듈은 가능성을 사용하여 제공할 오퍼를 쉽게 선택합니다.

학습 모듈은 두 개의 등록 정보를 기준으로 탐색과 이용 사이를 오갑니다(즉, confidenceLevel 등록 정보를 사용하여 구성하는 신뢰 수준과 percentRandomSelection 등록 정보를 사용하여 구성하는 학습 모듈의 무작위 오퍼 제공 가능성).

confidenceLevel을 중재에서 오퍼 점수가 사용되기 전에 학습 모듈의 신뢰 정도를 표시하는 비율로 설정하십시오. 처음에, 학습 모듈에 작업할 데이터가 없으면 학습 모듈은 마케팅 점수에 전적으로 의존합니다. minPresentCountThreshold가 정의한 만큼 모든 오퍼가 제공된 후 학습 모듈은 탐색 모드에 진입합니다. 작업할 데이터가 많지 않으면 학습 모듈은 계산한 비율이 올바르다고 확신하지 않습니다. 따라서 탐색 모드를 유지합니다.

학습 모듈은 각 오퍼에 가중치를 할당합니다. 가중치를 계산하기 위해 학습 모듈은 기록 수락 데이터 및 현재 세션 데이터는 물론 구성된 신뢰 수준을 입력으로 사용하는 공식을 사용합니다. 이 공식은 내재적으로 탐색과 이용 사이에서 균형을 유지하고 해당 가중치를 반환합니다.

시스템이 초기 단계 동안 최상으로 수행하는 오퍼로 기울지 않도록 Interact는 percentRandomSelection%의 무작위 오퍼를 제공합니다. 이는 학습 모듈이 성공 가능성이 가장 낮은 오퍼를 권장하여 다른 오퍼가 노출이 많으면 성공 가능성이 높은지 여부를 판별하도록 강제합니다. 예를 들어, percentRandomSelection을 5로 구성하면 이는 학습 모듈이 5%의 무작위 오퍼를 제공하고 계산에 응답 데이터를 추가함을 의미합니다.

학습 모듈은 다음과 같은 방법으로 제공되는 오퍼를 판별합니다.

1. 방문자가 오퍼를 선택할 가능성을 계산합니다.
2. 1단계의 가능성을 사용하여 오퍼 가중치를 계산하고 탐색 모드여야 하는지 이용 모드여야 하는지 여부를 판별합니다.
3. 2단계의 오퍼 가중치 및 마케팅 점수를 사용하여 각 오퍼의 최종 점수를 계산합니다.
4. 3단계에서 판별한 점수별로 오퍼를 정렬하고 요청된 상위 오퍼 수를 리턴합니다.

예를 들어, 학습 모듈은 방문자가 오퍼 A를 수락할 가능성이 30%이고 오퍼 B를 수락할 가능성이 70%이며 이 정보를 이용해야 한다고 판별합니다. 처리 규칙에서 오퍼 A

의 마케팅 점수는 75이고 오퍼 B의 경우는 55입니다. 그러나 3단계에서의 계산은 오퍼 B의 최종 점수가 오퍼 A보다 높습니다. 따라서 런타임 환경은 오퍼 B를 권장합니다.

학습은 recencyWeightingFactor 등록 정보 및 recencyWeightingPeriod 등록 정보도 기준으로 합니다. 이러한 등록 정보를 사용하여 이전 데이터보다 최근 데이터에 가중치를 더 추가할 수 있습니다. recencyWeightingFactor는 최근 데이터가 가져야 하는 가중치 비율입니다. recencyWeightingPeriod는 최근 시간 길이입니다. 예를 들어, recencyWeightingFactor를 .30으로, recencyWeightingPeriod를 24로 구성하십시오. 이는 데이터의 이전 24시간이 고려되는 모든 데이터의 30%임을 의미합니다. 일주일치 데이터가 있으면, 처음 6일 간의 모든 평균 데이터가 데이터의 70%이고 나머지 날은 데이터의 30%입니다.

모든 세션은 학습 스테이징 테이블에 다음 데이터를 기록합니다.

- 오퍼 컨택
- 오퍼 수락
- 학습 속성

애그리게이터는 구성 가능한 간격으로 스테이징 테이블에서 데이터를 읽고 컴파일한 후 테이블에 기록합니다. 학습 모듈은 집계된 이 데이터를 읽고 계산에서 사용합니다.

학습 모듈 사용

모든 런타임 서버에는 기본 제공 학습 모듈이 있습니다. 기본적으로 이 학습 모듈은 비활성되어 있습니다. 구성 등록 정보를 변경하여 학습을 설정할 수 있습니다.

런타임 환경의 Marketing Platform에서 Interact > offerserving 범주의 다음 구성 등록 정보를 편집하십시오.

구성 등록 정보	설정
optimizationType	**BuiltInLearning**

학습 속성

학습 모듈은 방문자 속성 및 오퍼 수락 데이터를 사용하여 학습합니다. 모니터하는 방문자 속성을 선택할 수 있습니다. 이 방문자 속성은 대화식 플로차트에서 참조하는 차원 테이블에 저장된 속성 또는 실시간으로 수집하는 일부 이벤트 매개변수를 포함하여 고객 프로파일 내 어떤 것일 수 있습니다.

모니터할 속성을 얼마든지 구성할 수 있지만, IBM에서는 다음 지침을 따라 정적 및 동적 학습 속성 사이에 오직 10개의 학습 속성을 구성하도록 권장합니다.

- 독립 속성을 선택하십시오.

유사한 속성을 선택하지 마십시오. 예를 들어, HighValue라는 속성을 작성하고 해당 속성이 급여를 기준 계산에 의해 정의된 경우 HighValue와 Salary를 모두 선택하지는 마십시오. 유사한 속성은 학습 알고리즘에 도움이 되지 않습니다.

- 별개 값을 가진 속성을 선택하십시오.

  속성이 값 범위를 갖는 경우, 정확한 값을 선택해야 합니다. 예를 들어, 급여를 속성으로 사용하고 각 급여 범위에 특정 값을 지정해야 할 경우 20,000 - 30,000 범위는 A여야 하고 30,001 - 40,000은 B여야 합니다.
- 성능을 저해하지 않도록 추적할 속성 수를 제한하십시오.

  추적할 수 있는 속성 수는 성능 요구 사항 및 Interact 설치에 따라 다릅니다. 가능하다면, 다른 모델링 도구(예: PredictiveInsight)를 사용하여 상위 10개의 예측 가능한 속성을 판별하십시오. 예측 불가능하지만 성능 비용도 드는 속성을 자동으로 정리하도록 학습 모듈을 구성할 수 있습니다.

모니터하는 속성 수와 모니터하는 속성당 값 수를 모두 정의하여 성능을 관리할 수 있습니다. `maxAttributeNames` 등록 정보는 추적하는 최대 방문자 수 속성을 정의합니다. `maxAttributeValues` 등록 정보는 속성당 추적하는 최대 값 수를 정의합니다. `otherAttributeValue` 등록 정보 값이 정의한 범주에는 기타 모든 값이 할당됩니다. 그러나 학습 엔진은 발생하는 첫 번째 값만 추적합니다. 예를 들어, 방문자 속성 안구 색상을 추적 중입니다. 파란색, 갈색, 녹색에만 관심이 있으므로 `maxAttributeValues`를 3으로 설정합니다. 그러나 처음 세 방문자의 값은 파란색, 갈색, 담갈색입니다. 이는 녹색 안구를 가진 모든 방문자에게 `otherAttributeValue`가 지정됨을 의미합니다.

학습 조건을 보다 명확하게 지정할 수 있게 하는 동적 학습 속성도 사용할 수 있습니다. 동적 학습 속성을 사용하여 두 속성의 조합을 단일 항목으로 학습할 수 있습니다. 예를 들어, 다음 프로파일 정보를 고려하십시오.

방문자 ID	카드 유형	카드 잔액
1	골드 카드	$1,000
2	골드 카드	$9,000
3	브론즈 카드	$1,000
4	브론즈 카드	$9,000

표준 학습 속성을 사용하는 경우, 카드 유형 및 잔액만 개별적으로 학습할 수 있습니다. 방문자 1, 2는 동일한 카드 유형을 기준으로 함께 그룹화되고 방문자 2, 4는 카드 잔액을 기준으로 그룹화됩니다. 이는 오퍼 수락 동작의 정확한 작동의 예측 변수가 아닐 수 있습니다. 골드 카드 소지자의 잔액이 더 많으면 방문자 2의 동작은 방문자 4와 근본적으로 다르며, 이는 표준 학습 속성을 왜곡합니다. 그러나 동적 학습 속성을 사용하면 이러한 방문자를 각각 개별적으로 학습하며 예측이 보다 정확합니다.

동적 학습 속성을 사용하고 방문자가 속성에 대한 두 개의 유효한 값을 갖는 경우, 학습 모듈은 모듈이 찾는 첫 번째 값을 선택합니다.

enablePruning 등록 정보를 yes로 설정하면, 학습 모듈은 예측 불가능한 속성을 알고리즘적으로 판별하고 가중치 계산 시 해당 속성을 고려하지 않습니다. 예를 들어, 모발 색상을 나타내는 속성을 추적하고 학습 모듈이 방문자의 모발 색상을 기준으로 수락할 패턴이 없다고 판별하면 학습 모듈은 모발 색상 속성을 고려하지 않습니다. 속성은 학습 집계 프로세스가 실행될 때마다 다시 평가됩니다 (aggregateStatsIntervalInMinutes 등록 정보로 정의됨). 동적 학습 속성도 정리됩니다.

학습 속성 정의

최대 maxAttributeNames개의 방문자 수 속성을 구성할 수 있습니다.

디자인 환경의 Marketing Platform에서 Campaign > partitions > partition*n* > Interact > learning 범주의 다음 구성 등록 정보를 편집하십시오.

*(learningAttributes)*는 새 학습 속성을 작성할 템플릿입니다. 각 속성에 대해 새 이름을 입력해야 합니다. 이름이 동일한 두 개의 범주를 작성할 수 없습니다.

구성 등록 정보	설정
attributeName	attributeName은 프로파일 데이터의 이름-값 쌍 이름과 일치해야 합니다. 이 이름은 대소문자를 구분하지 않습니다.

동적 학습 속성 정의

동적 학습 속성을 정의하려면 학습 데이터 소스의 UACI_AttributeList 테이블을 채워야 합니다.

이 테이블의 모든 열은 varchar(64) 유형입니다.

열	설명
AttributeName	학습할 동적 속성 이름입니다. AttributeNameCol에서 가능한 실제 값이어야 합니다.
AttributeNameCol	AttributeName을 찾을 수 있는 완전한 열 이름입니다(프로파일 테이블에서 시작하는 계층 구조). 이 열 이름은 표준 학습 속성이 아니어도 됩니다.
AttributeValueCol	AttributeName에 대한 연관 값을 찾을 수 있는 완전한 열 이름입니다(프로파일 테이블에서 시작하는 계층 구조).

예를 들어, 다음 프로파일 테이블 및 연관된 차원 테이블을 고려하십시오.

표 5. *MyProfileTable*

VisitorID	KeyField
1	Key1
2	Key2
3	Key3
4	Key4

표 6. *MyDimensionTable*

KeyField	CardType	CardBalance
Key1	골드 카드	1000
Key2	골드 카드	9000
Key3	브론즈 카드	1000
Key4	브론즈 카드	9000

다음은 카드 유형과 잔액이 일치하는 `UACI_AttributeList` 테이블 샘플입니다.

표 7. *UACI_AttributeList*

AttributeName	AttributeNameCol	AttributeValueCol
골드 카드	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance
브론즈 카드	MyProfileTable.MyDimensionTable.CardType	MyProfileTable.MyDimensionTable.CardBalance

외부 학습 사용

학습 Java™ API를 사용하여 자체 학습 모듈을 작성할 수 있습니다. Marketing Platform에서 학습 유틸리티를 인식하도록 런타임 환경을 구성할 수 있습니다.

런타임 환경의 Marketing Platform에서 `Interact > offerserving` 범주의 다음 구성 등록 정보를 편집하십시오. 학습 최적화 프로그램 API에 대한 구성 등록 정보는 `Interact > offerserving > External Learning Config` 범주에 있습니다.

구성 등록 정보	설정
`optimizationType`	**ExternalLearning**
`externalLearningClass`	외부 학습에 대한 클래스 이름
`externalLearningClassPath`	런타임 서버의 외부 학습에 대한 클래스 또는 jar 파일 경로입니다. 서버 그룹을 사용 중이고 런타임 서버가 동일한 Marketing Platform 인스턴스를 참조하는 경우, 모든 서버에는 동일한 위치에 클래스 또는 jar 파일 복사본이 있어야 합니다.

이러한 변경 사항을 적용하려면 Interact 런타임 서버를 다시 시작해야 합니다.

제 5 장 Interact API 이해

Interact는 다양한 접점에 동적으로 오퍼를 제공합니다. 예를 들어, 특정 유형의 서비스 조회로 통화한 고객의 최상 상향 판매 또는 연결 판매 가능성을 알리는 메시지를 콜센터 직원에게 보내도록 런타임 환경과 접점을 구성할 수 있습니다. 또한 웹 사이트의 특정 영역으로 이동한 고객(방문자)에게 고객 맞춤 오퍼를 제공하도록 런타임 환경과 접점을 구성할 수도 있습니다.

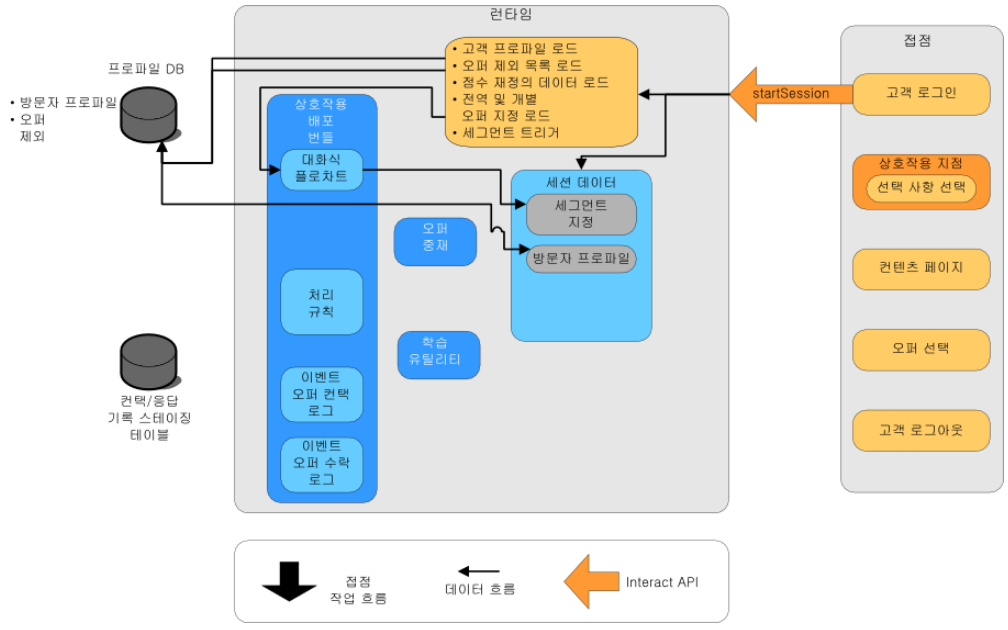
Interact API(Application Programming Interface)로 가능한 최상의 오퍼를 제공하도록 런타임 서버 및 접점을 구성할 수 있습니다. API를 사용하여 접점은 런타임 서버에서 방문자를 그룹(세그먼트)에 할당하고 이 세그먼트에 기초하여 오퍼를 제시할 정보를 요청할 수 있습니다. 오퍼 프리젠테이션 전략을 세분화하기 위해 나중에 분석하도록 데이터를 로그할 수도 있습니다.

Interact와 사용자의 환경을 가능한한 탄력적으로 통합하기 위해 IBM은 Interact API를 사용하여 액세스할 수 있는 웹 서비스를 제공합니다.

Interact API 데이터 플로

다음 그림은 Interact API의 단순 구현을 보여줍니다. 방문자는 웹 사이트에 로그인하고 오퍼가 표시된 페이지로 이동합니다. 방문자는 오퍼를 선택하고 로그아웃합니다. 상호작용이 단순한 반면 접점과 런타임 서버 모두에서 발생하는 여러 이벤트가 있습니다.

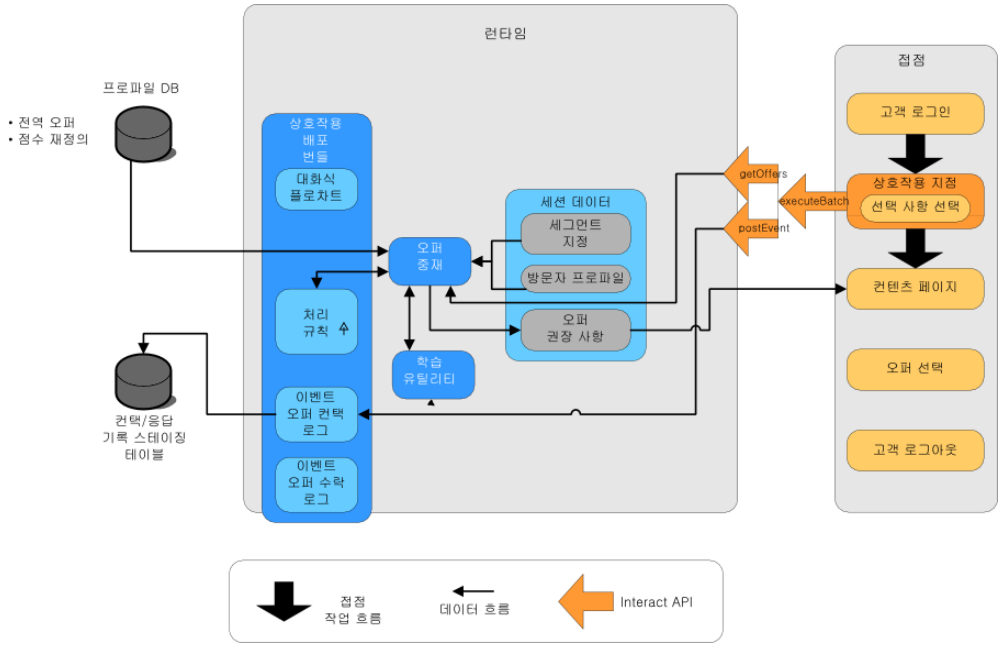
방문자가 로그인하면 `startSession`이 트리거됩니다.



© Copyright IBM Corp. 2001, 2012

이 예에서 startSession 메소드는 네 가지 일을 수행합니다. 첫 번째로 새 런타임 세션을 생성합니다. 두 번째로, 고객 프로파일 데이터를 세션으로 로드할 요청을 보냅니다. 세 번째는, 프로파일 데이터를 사용하고 고객을 세그먼트로 배치할 대화식 플로차트를 시작하는 요청을 보냅니다. 이 플로차트는 비동기식으로 실행됩니다. 네 번째는, 런타임이 오퍼 제외 및 전역과 개별 오퍼 처리 정보를 세션으로 로드합니다. 세션 기간 동안 세션 데이터가 메모리에 보관됩니다.

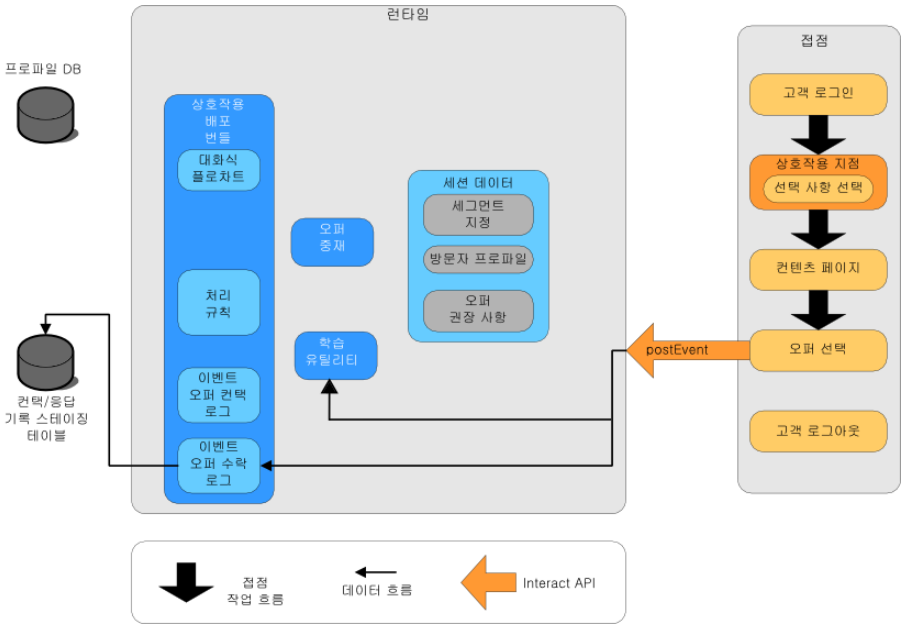
방문자는 미리 정의된 상호작용 지점에 도달할 때까지 사이트를 탐색합니다. 그림에서, 두 번째 상호작용 지점(선택 사항 선택)는 방문자가 오퍼 세트를 표시하는 링크를 클릭하는 곳입니다. 접점 관리자가 executeBatch 메소드를 트리거하도록 링크를 구성했습니다.



executeBatch 메소드로 런타임 서버에 대한 단일 호출에서 둘 이상의 메소드를 호출할 수 있습니다. 이 특정 executeBatch는 두 가지 다른 메소드인 getOffers 및 postEvent를 호출합니다. getOffers 메소드는 오퍼 목록을 요청합니다. 런타임에 세그먼트 데이터, 오퍼 제외 목록, 처리 규칙, 학습 모듈을 사용하여 오퍼 세트를 제안합니다. 런타임에 컨텐츠 페이지에 표시된 오퍼 세트가 리턴됩니다.

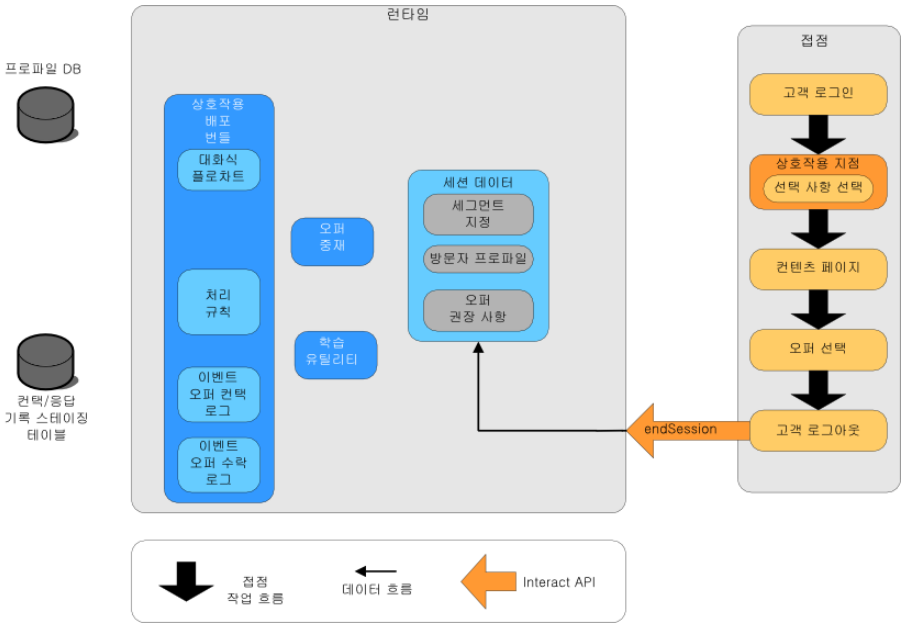
postEvent 메소드는 디자인 환경에 정의된 메소드 중 하나를 트리거합니다. 특별한 경우 이벤트는 컨택 기록에 표시된 오퍼를 로그할 요청을 보냅니다.

방문자가 오퍼 중 하나를 선택합니다(오퍼 선택).



오퍼 선택과 연관된 단추는 또 다른 postEvent 메소드를 보내도록 구성되어 있습니다. 이 이벤트는 응답 기록에 오퍼 수락을 로그할 요청을 보냅니다.

오퍼를 선택한 후 방문자는 웹 사이트를 마치고 로그아웃합니다. 로그아웃 명령은 endSession 메소드로 연결됩니다.



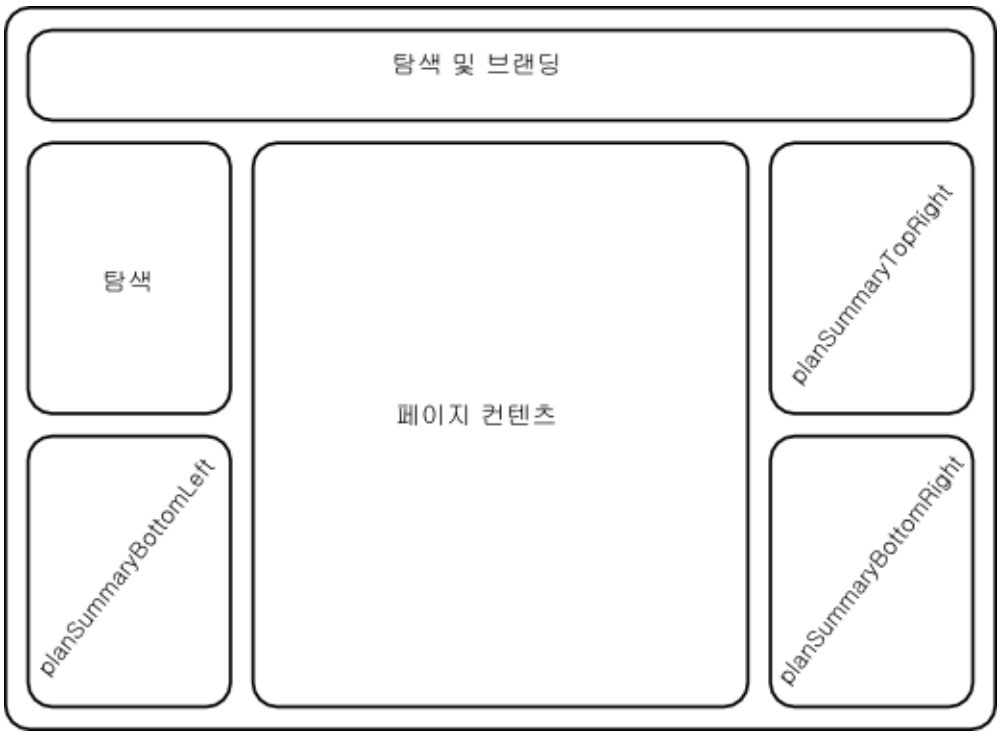
endSession 메소드가 세션을 닫습니다. 방문자가 로그아웃을 잊은 경우 모든 세션이 종료될 수 있도록 구성 가능한 세션 제한시간이 있습니다. startSession 또는 setAudience 메소드의 매개변수에 포함된 정보와 같이 세션에 전달된 데이터를 보관하려면 대화식 플로차트를 생성한 사람과 작업하십시오. 대화식 플로차트를 생성하는 사람은 스냅샷 프로세스를 사용하여 세션이 종료되고 이 데이터가 손실되기 전에 데이터

를 데이터베이스에 씁니다. 그러면 postEvent 메소드를 사용하여 스냅샷 프로세스를 포함한 대화식 플로차트를 호출할 수 있습니다.

이 예는 접점과 런타임 환경 간에 API가 작동하는 방식의 기본사항을 보여주는 매우 단순한 예입니다(방문자가 단순 상호작용인 네 가지 작업 즉, 로그인, 오퍼가 표시된 페이지로 이동, 오퍼 선택, 로그아웃만 수행함). 필요한 대로 보다 복잡하게 통합을 디자인할 수 있습니다(성능 요구 사항의 한계 내에서).

단순 상호작용 계획 예

휴대 전화 회사의 웹 사이트에 대한 상호작용을 디자인하고 있습니다. 다음 다이어그램은 휴대 전화 계획 요약 페이지의 레이아웃을 보여줍니다.



휴대 전화 계획 요약 페이지의 요구 사항에 맞게 다음 항목을 정의합니다.

업그레이드에 대한 오퍼 전용 영역에 표시할 오퍼 하나

- 업그레이드 오퍼를 표시하는 페이지의 영역을 정의해야 합니다. 또한 Interact가 표시할 오퍼를 선택하고 나면 정보를 로그해야 합니다.

  상호작용 지점: ip_planSummaryBottomRight

  이벤트: evt_logOffer

전화 업그레이드를 위한 두 개의 오퍼

- 전화 업그레이드 표시하는 페이지의 각 영역을 정의해야 합니다.

상호작용 지점: ip_planSummaryTopRight

상호작용 지점: ip_planSummaryBottomLeft

분석을 위해 수락되는 오퍼와 거부되는 오퍼를 로그해야 합니다.

이벤트: evt_offerAccept

이벤트: evt_offerReject

오퍼 컨택, 수락 또는 거부를 로그할 때마다 오퍼의 처리 코드를 전달해야 하는지도 알고 있어야 합니다. 필요에 따라 NameValuePair를 생성하여 다음 예에서처럼 처리 코드를 포함합니다.

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

이제 디자인 환경 사용자에게 접점과의 통합을 코딩하는 동안 상호작용 지점 및 이벤트를 생성하도록 요청할 수 있습니다.

오퍼를 표시할 각 상호작용 지점마다 먼저 오퍼를 가져온 후 오퍼를 표시해야 할 정보를 추출해야 합니다. 예를 들어, 웹 페이지의 하단 오른쪽 영역에 대한 오퍼를 요청하십시오(planSummaryBottomRight).

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

그러면 OfferList 응답을 포함한 응답 개체가 리턴됩니다. 하지만 웹 페이지에 OfferList 개체를 사용할 수 없습니다. 오퍼 속성 중 하나로 알고 있는 오퍼의 이미지 파일(offerImg)이 필요합니다. OfferList에서 필요한 오퍼 속성을 추출해야 합니다.

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

이제 오퍼를 표시해야 하므로 오퍼를 컨택으로 로그하려 합니다.

```
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)
```

각 메소드를 개별적으로 호출하지 않고 웹 페이지의 planSummaryBottomLeft 부분에 대한 다음 예에 표시된 대로 executeBatch 메소드를 사용할 수 있습니다.

```
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);
```

UACIOfferTrackingCode를 제공하지 않으면 Interact Interact 런타임 서버가 자동으로 권장하는 최신 처리 목록을 컨택으로 로그하기 때문에 이 예에서는 UACIOfferTrackingCode를 정의할 필요가 없습니다.

또한 전화 업그레이드 오퍼를 표시하는 페이지의 두 번째 영역에 대해 30초마다 표시되는 이미지를 변경하기 위해 무언가를 썼습니다. 세 가지 이미지를 회전시키도록 결정했으므로 다음을 이용하여 이미지를 회전시키기 위해 코드에 사용할 캐시하려는 오퍼 세트를 검색해야 합니다.

```
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
  }
 }
}
```

로컬 캐시에서 클라이언트 코드 페치를 쓰고 이미지가 표시된 후 각 오퍼마다 한 번만 컨택에 로그해야 합니다. 컨택에 로그하려면 UACITrackingCode 매개변수를 예전처럼 게시해야 합니다. 각 오퍼의 추적 코드는 서로 다릅니다.

```
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);
  }
 }
}
```

각 오퍼를 클릭하여 수락된 오퍼와 거부된 오퍼를 로그해야 합니다. (이 시나리오에서 명시적으로 선택되지 않은 오퍼는 거부된 것으로 간주됩니다.) 다음은 ip_planSummaryTopRight 오퍼가 선택된 경우의 예입니다.

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

특히, 세 가지 postEvent 호출을 executeBatch 메소드와 함께 보내는 것이 가장 좋습니다.

이는 기본 예이며 통합을 쓰는 최상의 방법을 표시하지는 않습니다. 예를 들어, 이 예의 어디에도 Response 클래스를 사용한 오류 검사는 포함되어 있지 않습니다.

Interact API 통합 디자인

접점과의 Interact API 통합을 빌드하려면 구현을 시작하기 전에 몇 가지 디자인 작업이 필요합니다. 마케팅 팀과 작업하여 접점에서 런타임 환경에 오퍼를 제공할 위치(상호작용 지점 정의) 및 사용하려는 추적이나 대화식 기능의 유형(이벤트 정의)을 결정해야 합니다. 디자인 단계에서 이는 그저 아웃라인일 수 있습니다. 예를 들어, 전자 통신 웹 사이트의 경우 고객의 계획 요약 페이지에는 계획 업그레이드에 관한 한 가지 오퍼와 전화 업그레이드에 대한 두 개의 오퍼가 표시되어야 합니다.

회사에서 고객과 상호작용할 위치 및 방식을 결정하고 나면 Interact를 사용하여 세부 정보를 정의해야 합니다. 플로차트를 만든 이는 재세그먼트 이벤트가 발생할 때 사용할 대화식 플로차트를 디자인해야 합니다. 적절한 세그먼트, 이벤트 게시, 오퍼 검색을 위해 전달해야 하는 데이터 뿐 아니라 상호작용 지점 및 이벤트의 수와 이름을 결정해야 합니다. 디자인 환경 사용자는 대화식 채널의 상호작용 지점와 이벤트를 정의합니다. 그러면 런타임 환경에서 접점과의 통합을 코딩할 때 이 이름을 사용합니다. 오퍼 컨택 및 응답을 로그해야 할 시기를 정의하기 위해 필요한 메트릭 정보도 정의해야 합니다.

고려할 사항

통합을 쓸 때 다음 팁을 기억하십시오.

- 접점을 디자인할 때 오퍼를 제시할 수 있는 모든 상호작용 지점에 대해 몇 가지 기본 필러 컨텐츠(일반적으로 무해한 유형의 메시지나 비어 있는 컨텐츠)를 생성하십시오. 이는 현재 상황에서 현재 방문자에게 적합한 오퍼가 제공되지 않을 경우입니다. 상호작용 지점의 기본 문자열로 이 기본 필러 컨텐츠를 할당해야 합니다.
- 접점을 디자인할 때 어떠한 뜻하지 않은 이유로 인해 접점이 런타임 서버 그룹에 도달할 수 없는 경우 컨텐츠를 표시할 몇 가지 방법을 포함하십시오.
- `postEvent` 및 `setAudience`를 포함하여 방문자를 다시 세그먼트하는 이벤트를 트리거할 때 실행 중인 플로차트에 약간의 시간이 소요됨에 유의하십시오. `getOffers` 메소드는 실행하기 전에 세그먼트가 완료될 때까지 대기합니다. 지나치게 자주 재세그먼트를 수행하는 경우 `getOffers` 호출 응답 성능이 저하될 수 있습니다.
- "오퍼 거부"의 의미를 결정해야 합니다. 채널 오퍼 성과 요약 보고서와 같은 여러 보고서에는 오퍼가 거부된 횟수가 표시됩니다. 이는 `postEvent`로 오퍼 거부 로그 작업이 트리거된 횟수입니다. 오퍼 거부 로그를 실제 거부(예를 들어, "아니요, 감사합니다."란 레이블의 링크를 클릭) 또는 무시되는 오퍼(예를 들어, 새 가지 배너 광고가 표시된 페이지에서 아무 것도 선택되지 않음)로 할지 여부를 판별해야 합니다.
- 학습, 오퍼 제외, 개별 오퍼 지정, 다른 오퍼 제공 요소를 포함하여 Interact 오퍼 선택을 개선할 수 있는 여러 가지 선택적 기능이 있습니다. 통합을 향상시키기 위해 사용할 선택적 기능의 수를 판별해야 합니다(사용하는 경우).

제 6 장 IBM Unica Interact API 관리

startSession 메소드를 사용할 때마다 런타임 서버에 Interact 런타임 세션을 생성합니다. 구성 등록 정보를 사용하여 런타임 서버의 세션을 관리할 수 있습니다. 접점과의 Interact 통합을 구현할 때 이 설정을 구성해야 할 수 있습니다.

구성 등록 정보는 sessionManagement 범주에 있습니다.

로케일 및 Interact API

영어 이외의 접점에 Interact를 사용할 수 있습니다. API의 모든 문자열과 접점에는 런타임 환경 사용자에 대해 정의된 로케일이 사용됩니다.

서비스 그룹별로 로케일을 하나만 선택할 수 있습니다.

예를 들어, 런타임 환경에서 사용자 로케일이 영어로 설정된 asm_admin_en 및 사용자 로케일이 프랑스어로 설정된 asm_admin_fr이라는 두 명의 사용자를 생성합니다. 접점이 프랑스어로 디자인된 경우 런타임 환경의 asmUserForDefaultLocale 등록 정보를 asm_admin_fr로 정의하십시오.

JMX 모니터링 정보

Interact는 JMX 응용 프로그램으로 액세스할 수 있는 JMX(Java Management Extensions) 모니터링 서비스를 제공합니다. JMX 모니터링으로 런타임 서버를 모니터하고 관리할 수 있습니다. JMX 속성은 런타임 서버에 대한 많은 자세한 정보를 제공합니다. 예를 들어, JMX 속성 ErrorCount는 마지막 재설정 또는 시스템 시작 이후에 로그된 오류 메시지 수를 제공합니다. 이 정보를 이용하여 시스템에 오류가 발생하는 빈도를 확인할 수 있습니다. 세션 종료를 호출하는 용도로만 웹 사이트를 코딩한 경우 누군가 트랜잭션을 완료하면 startSessionCount를 endSessionCount와 비교하여 미완료된 트랜잭션 수를 확인할 수 있습니다.

Interact는 JSR 160에 정의된 RMI 및 JMXMP를 지원합니다. JSR160 호환 JMX 클라이언트로 JMX 모니터링 서비스에 연결할 수 있습니다.

대화식 플로차트는 JMX 모니터링을 통해서만 모니터할 수 있습니다. 대화식 플로차트에 대한 정보는 Campaign 모니터링에 표시되지 않습니다.

참고: IBM WebSphere®를 노드 관리자로 사용 중인 경우 JMX 모니터링을 설정하려면 일반 JVM 인수를 정의해야 합니다.

© Copyright IBM Corp. 2001, 2012

RMI 프로토콜에 대한 JMX 모니터링을 사용하도록 Interact 구성

런타임 환경에 대한 Marketing Platform에서 Interact > 모니터링 범주의 다음 구성 등록 정보를 편집하십시오.

구성 등록 정보	설정
protocol	**RMI**
port	JMX 서비스의 포트 번호입니다.
enableSecurity	**False** RMI 프로토콜의 Interact 구현은 보안을 지원하지 않습니다.

RMI 프로토콜에 대한 모니터링의 기본 주소는 service:jmx:rmi:///jndi/rmi://*RuntimeServer*:*port*/interact입니다.

JMXMP 프로토콜에 대한 JMX 모니터링을 사용하도록 Interact 구성

JMXMP 프로토콜은 클래스 경로 InteractJMX.jar 및 jmxremote_optional.jar에 다음 순서로 두 개의 추가 라이브러리가 필요합니다. 두 파일 모두 런타임 환경 설치의 lib 디렉토리에 있습니다.

참고: 보안을 사용하면 사용자 이름과 암호가 런타임 환경에 대한 Marketing Platform의 사용자와 일치해야 합니다. 비어 있는 암호를 사용하면 안됩니다.

런타임 환경에 대한 Marketing Platform에서 Interact > 모니터링 범주의 다음 구성 등록 정보를 편집하십시오.

구성 등록 정보	설정
protocol	**JMXMP**
port	JMX 서비스의 포트 번호입니다.
enableSecurity	**False**로 설정하여 보안을 비활성화하거나 **True**로 설정하여 보안을 사용하십시오.

JMXMP 프로토콜에 대한 모니터링의 기본 주소는 service:jmx:jmxmp://*RuntimeServer*:*port*입니다.

jconsole 스크립트 사용

별도의 JMX 모니터링 응용 프로그램이 없는 경우 JVM과 함께 설치된 jconsole을 사용할 수 있습니다. Interact/tools 디렉토리에서 시작 스크립트를 사용하여 jconsole을 시작할 수 있습니다.

1. Interact\tools\jconsole.bat(Windows) 또는 Interact/tools/jconsole.sh(Unix)를 텍스트 편집기에서 여십시오.

2. INTERACT_LIB를 *InteractInstallationDirectory*/lib 디렉토리의 전체 경로로 설정하십시오.
3. HOST를 모니터하려는 런타임 서버의 호스트 이름으로 설정하십시오.
4. Interact > 모니터링 > port 등록 정보에서 청취할 JMX를 구성한 포트로 PORT를 설정하십시오.
5. RMI 프로토콜을 통해 모니터 중이면 JMXMP 연결 앞에 설명을 추가하고 RMI 연결 앞의 설명을 제거하십시오.

   스크립트가 기본적으로 JMXMP 프로토콜을 통해 모니터합니다.

   예를 들어, jconsole.bat의 기본 설정을 참조하십시오.

   JMXMP 연결

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

   RMI 연결

```
%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
```

JMX 속성

다음 테이블은 JMX 모니터링에 사용 가능한 속성을 설명합니다.

JMX 모니터링의 제공된 모든 데이터는 마지막 재설정 또는 시스템 시작 이후의 데이터입니다. 예를 들어, 개수는 설치 이후가 아닌 마지막 재설정이나 시스템 시작 이후의 항목 수입니다.

표 8. 컨택 응답 기록 ETL 모니터

속성	설명
AvgCHExecutionTime	컨택 및 응답 기록 모듈이 컨택 기록 테이블에 쓰는 데 걸리는 평균 시간(밀리초)입니다. 이 평균은 컨택 기록 테이블에 최소 하나의 레코드가 기록되었으며 성공한 작업에 대해서만 계산됩니다.
AvgETLExecutionTime	컨택 및 응답 기록 모듈이 런타임 환경에서 데이터를 읽는 데 걸리는 평균 시간(밀리초)입니다. 성공한 작업과 실패한 작업에 대한 시간이 평균에 포함됩니다.
AvgRHExecutionTime	컨택 및 응답 기록 모듈이 응답 기록 테이블에 쓰는 데 걸리는 평균 시간(밀리초)입니다. 이 평균은 응답 기록 테이블에 최소 하나의 레코드가 기록되었으며 성공한 작업에 대해서만 계산됩니다.

표 8. 컨택 응답 기록 ETL 모니터 (계속)

속성	설명
ErrorCount	마지막 재설정 또는 시스템 시작 이후에 로그된 오류 메시지 수입니다(있는 경우).
HighWaterMarkCHExecutionTime	컨택 및 응답 기록 모듈이 컨택 기록 테이블에 쓰는 데 걸린 최대 시간(밀리초)입니다. 이 값은 컨택 기록 테이블에 최소 하나의 레코드가 기록되었으며 성공한 작업에 대해서만 계산됩니다.
HighWaterMarkETLExecutionTime	컨택 및 응답 기록 모듈이 런타임 환경에서 데이터를 읽는 데 걸린 최대 시간(밀리초)입니다. 성공한 작업과 실패한 작업 모두 계산에 포함됩니다.
HighWaterMarkRHExecutionTime	컨택 및 응답 기록 모듈이 응답 기록 테이블에 쓰는 데 걸린 최대 시간(밀리초)입니다. 이 값은 응답 기록 테이블에 최소 하나의 레코드가 기록되었으며 성공한 작업에 대해서만 계산됩니다.
LastExecutionDuration	컨택 및 응답 기록 모듈이 마지막 복사를 수행하는 데 걸린 시간(밀리초)입니다.
NumberOfExecutions	초기화 이후 컨택 및 응답 기록 모듈이 실행된 횟수입니다.
LastExecutionStart	컨택 및 응답 기록 모듈의 마지막 실행이 시작된 시간입니다.
LastExecutionSuccessful	true이면 컨택 및 응답 기록 모듈의 마지막 실행이 성공했습니다. false인 경우 오류가 발생했습니다.
NumberOfContactHistoryRecordsMarked	컨택 및 응답 기록 모듈의 현재 실행 중 이동되고 있는 UACI_CHStaging 테이블의 컨택 기록 레코드 수입니다. 이 값은 컨택 및 응답 기록 모듈이 현재 실행 중인 경우에만 0보다 큽니다.
NumberOfResponseHistoryRecordsMarked	컨택 및 응답 기록 모듈의 현재 실행 중 이동되고 있는 UACI_RHStaging 테이블의 응답 기록 레코드 수입니다. 이 값은 컨택 및 응답 기록 모듈이 현재 실행 중인 경우에만 0보다 큽니다.

컨택 응답 기록 ETL 모니터 속성은 디자인 환경의 일부입니다. 다음의 모든 속성은 런타임 환경의 일부입니다.

표 9. 예외

속성	설명
errorCount	마지막 재설정 또는 시스템 시작 이후에 로그된 오류 메시지 수입니다.
warningCount	마지막 재설정 또는 시스템 시작 이후에 로그된 경고 메시지 수입니다.

표 10. 플로차트 엔진 통계

속성	설명
activeProcessBoxThreads	현재 실행 중인 플로차트 프로세스 스레드(모든 실행에서 공유)의 활성 개수입니다.
activeSchedulerThreads	현재 실행 중인 플로차트 스케줄러 스레드의 활성 개수입니다.
avgExecutionTimeMillis	평균 플로차트 실행 시간(밀리초)입니다.
CurrentJobsInProcessBoxQueue	플로차트 프로세스 스레드가 실행을 대기 중인 작업 수입니다.
CurrentJobsInSchedulerQueue	플로차트 스케줄러 스레드가 실행을 대기 중인 작업 수입니다.
maximumProcessBoxThreads	실행할 수 있는 플로차트 프로세스 스레드(모든 실행에서 공유)의 최대 수입니다.
maximumSchedulerThreads	실행할 수 있는 플로차트 스케줄러 스레드(실행별로 하나의 스레드)의 최대 수입니다.
numExecutionsCompleted	완료된 플로차트 실행의 총 수입니다.
numExecutionsStarted	시작된 플로차트 실행의 총 수입니다.

표 11. 대화식 채널별 특정 플로차트

속성	설명
AvgExecutionTimeMillis	이 대화식 채널에서 이 플로차트의 평균 실행 시간(밀리초)입니다.
HighWaterMarkForExecutionTime	이 대화식 채널에서 이 플로차트의 최대 실행 시간(밀리초)입니다.
LastCompletedExecutionTimeMillis	이 대화식 채널에서 이 플로차트의 마지막 완료 실행 시간(밀리초)입니다.
NumExecutionsCompleted	이 대화식 채널에서 이 플로차트의 완료된 총 실행 수입니다.
NumExecutionsStarted	이 대화식 채널에서 이 플로차트의 시작된 총 실행 수입니다.

표 12. 로케일

속성	설명
locale	JMX 클라이언트의 로케일 설정입니다.

표 13. 로거 구성

속성	설명
category	로그 수준을 조작할 수 있는 로그 범주를 변경합니다.

표 14. 서비스 스레드 풀 통계

속성	설명
activeContactHistThreads	컨택 기록과 응답 기록에 대한 작업을 활발하게 실행 중인 스레드의 대략적 수입니다.

표 14. 서비스 스레드 풀 통계 (계속)

속성	설명
activeFlushCacheToDBThreads	캐시된 통계를 데이터 저장소로 비우는 작업을 활발하게 실행 중인 스레드의 대략적 수입니다.
activeOtherStatsThreads	자격 통계, 이벤트 활동, 기본 통계에 대한 작업을 활발하게 실행 중인 스레드의 대략적 수입니다.
CurrentHighWaterMarkInContactHistQueue	컨택 및 응답 기록 데이터를 수집하는 서비스가 로그하도록 대기열에 있는 항목의 최대 수입니다.
CurrentHighWaterMark InFlushCachetoDBQueue	캐시의 데이터를 데이터베이스 테이블에 쓰는 서비스가 로그하도록 대기열에 있는 항목의 최대 수입니다.
CurrentHighWaterMarkInOtherStatsQueue	오퍼 자격 통계, 기본 문자열 사용 통계, 이벤트 활동 통계, 테이블 데이터에 대한 사용자 정의 로그를 수집하는 서비스가 로그하도록 대기열에 있는 항목의 최대 수입니다.
currentMsgsInContactHistQueue	컨택 기록 및 응답 기록에 사용되는 스레드 풀에 대한 대기열의 작업 수입니다.
currentMsgsInFlushCacheToDBQueue	캐시된 통계를 데이터 저장소로 비우는 데 사용되는 스레드 풀에 대한 대기열의 작업 수입니다.
currentMsgsInOtherStatsQueue	자격 통계, 이벤트 활동, 기본 통계에 사용되는 스레드 풀에 대한 대기열의 작업 수입니다.
maximumContactHistThreads	컨택 기록과 응답 기록에 사용되는 풀에 동시에 있었던 스레드의 최대 수입니다.
maximumFlushCacheToDBThreads	캐시된 통계를 데이터 저장소로 비우는 데 사용되는 풀에 동시에 있었던 스레드의 최대 수입니다.
maximumOtherStatsThreads	적합 통계, 이벤트 활동, 기본 통계에 사용되는 풀에 동시에 있었던 스레드의 최대 수입니다.

서비스 통계는 각 서비스에 대한 속성 세트로 이루어집니다.

- ContactHistoryMemoryCacheStatistics - 컨택 기록 스테이징 테이블에 대한 데이터를 수집하는 서비스입니다.
- CustomLoggerStatistics - 테이블에 쓸 사용자 정의 데이터를 수집하는 서비스입니다(`UACICustomLoggerTableName` 이벤트 매개변수를 사용하는 이벤트).
- 기본 통계 - 상호작용 지점에 대한 기본 문자열이 사용된 횟수에 관한 통계를 수집하는 서비스입니다.
- 자격 통계 - 적합한 오퍼에 대한 통계를 쓰는 서비스입니다.
- 이벤트 활동 통계 - `getOffer` 또는 `startSession`과 같은 시스템 이벤트 및 `postEvent`로 트리거되는 사용자 이벤트 모두의 이벤트 통계를 수집하는 서비스입니다.

- 응답 기록 메모리 캐시 통계 - 응답 기록 스테이징 테이블에 쓰는 서비스입니다.
- 교차 세션 응답 통계 - 교차 세션 응답 추적 데이터를 수집하는 서비스입니다.

표 15. 서비스 통계

속성	설명
Count	처리된 메시지 수입니다.
ExecTimeInsideMutex	다른 스레드를 대기하는 시간을 제외하고, 이 서비스에 대한 메시지를 처리하는 데 걸린 시간(밀리초)입니다. ExecTimeInsidMutex와 ExecTimeMillis의 차이가 크면 서비스의 스레드 풀 크기를 변경해야 할 수 있습니다.
ExecTimeMillis	다른 스레드를 대기하는 시간을 포함하여, 이 서비스에 대한 메시지를 처리하는 데 걸린 시간(밀리초)입니다.
ExecTimeOfDBInsertOnly	일괄처리 삽입 부분만 처리하는 데 걸린 시간(밀리초)입니다.
HighWaterMark	이 서비스의 처리된 최대 메시지 수입니다.
NumberOfDBInserts	실행된 일괄처리 삽입의 총 수입니다.
TotalRowsInserted	데이터베이스에 삽입된 총 행 수입니다.

표 16. 서비스 통계 - 데이터베이스 로드 유틸리티

속성	설명
ExecTimeOfWriteToCache	필요에 따라 데이터베이스에서 기본 키를 가져오고 파일에 쓰는 시간을 포함하여, 파일 캐시에 쓰는 데 걸린 시간(밀리초)입니다.
ExecTimeOfLoaderDBAccessOnly	데이터베이스 로더 부분만 실행하는 데 걸린 시간(밀리초)입니다.
ExecTimeOfLoaderThreads	데이터베이스 로더 스레드에 소모된 시간(밀리초)입니다.
ExecTimeOfFlushCacheFiles	캐시를 비우고 새로 재생성하는 데 걸린 시간(밀리초)입니다.
ExecTimeOfRetrievePKDBAccess	기본 키 데이터베이스 액세스를 검색하는 데 걸린 시간(밀리초)입니다.
NumberOfDBLoaderRuns	데이터베이스 로더 실행의 총 수입니다.
NumberOfLoaderStagingDirCreated	생성된 스테이징 디렉토리의 총 수입니다.
NumberOfLoaderStagingDirRemoved	제거된 스테이징 디렉토리의 총 수입니다.
NumberOfLoaderStagingDirMovedToAttention	주의로 이름이 변경된 스테이징 디렉토리의 총 수입니다.
NumberOfLoaderStagingDirMovedToError	오류로 이름이 변경된 스테이징 디렉토리의 총 수입니다.
NumberOfLoaderStagingDirRecovered	백그라운드 스레드의 재실행 및 시작 시간을 포함하여, 복구된 스테이징 디렉토리의 총 수입니다.
NumberOfTimesRetrievePKFromDB	데이터베이스에서 기본 키를 검색한 총 횟수입니다.

표 16. 서비스 통계 - 데이터베이스 로드 유틸리티 (계속)

속성	설명
NumberOfLoaderThreadsRuns	데이터베이스 로더 스레드 실행의 총 수입니다.
NumberOfFlushCacheFiles	파일 캐시를 비운 총 횟수입니다.

표 17. API 통계

속성	설명
endSessionCount	마지막 재설정 또는 시스템 시작 이후의 endSession API 호출 수입니다.
endSessionDuration	마지막 endSession API 호출 이후의 경과 시간입니다.
executeBatchCount	마지막 재설정 또는 시스템 시작 이후의 executeBatch API 호출 수입니다.
executeBatchDuration	마지막 executeBatch API 호출 이후의 경과 시간입니다.
getOffersCount	마지막 재설정 또는 시스템 시작 이후의 getOffers API 호출 수입니다.
getOffersDuration	마지막 getOffer API 호출 이후의 경과 시간입니다.
getProfileCount	마지막 재설정 또는 시스템 시작 이후의 getProfile API 호출 수입니다.
getProfileDuration	마지막 getProfileDuration API 호출 이후의 경과 시간입니다.
getVersionCount	마지막 재설정 또는 시스템 시작 이후의 getVersion API 호출 수입니다.
getVersionDuration	마지막 getVersion API 호출 이후의 경과 시간입니다.
loadOfferSuppressionDuration	마지막 loadOfferSuppression API 호출 이후의 경과 시간입니다.
LoadOffersBySQLCount	마지막 재설정 또는 시스템 시작 이후의 LoadOffersBySQL API 호출 수입니다.
LoadOffersBySQLDuration	마지막 LoadOffersBySQL API 호출 이후의 경과 시간입니다.
loadProfileDuration	마지막 loadProfile API 호출 이후의 경과 시간입니다.
loadScoreOverrideDuration	마지막 loadScoreOverride API 호출 이후의 경과 시간입니다.
postEventCount	마지막 재설정 또는 시스템 시작 이후의 postEvent API 호출 수입니다.
postEventDuration	마지막 postEvent API 호출 이후의 경과 시간입니다.
runSegmentationDuration	마지막 runSegmentation API 호출 이후의 경과 시간입니다.
setAudienceCount	마지막 재설정 또는 시스템 시작 이후의 setAudience API 호출 수입니다.

표 17. API 통계 (계속)

속성	설명
setAudienceDuration	마지막 setAudience API 호출 이후의 경과 시간입니다.
setDebugCount	마지막 재설정 또는 시스템 시작 이후의 setDebug API 호출 수입니다.
setDebugDuration	마지막 setDebug API 호출 이후의 경과 시간입니다.
startSessionCount	마지막 재설정 또는 시스템 시작 이후의 startSession API 호출 수입니다.
startSessionDuration	마지막 startSession API 호출 이후의 경과 시간입니다.

표 18. 학습 최적화 프로그램 통계

속성	설명
LearningOptimizerAcceptCalls	학습 모듈에 전달된 수락 이벤트 수입니다.
LearningOptimizerAcceptTrackingDuration	학습 모듈에서 수락 이벤트를 로그하는 데 걸린 총 시간(밀리초)입니다.
LearningOptimizerContactCalls	학습 모듈에 전달된 컨택 이벤트 수입니다.
LearningOptimizerContactTrackingDuration	학습 모듈에서 컨택 이벤트를 로그하는 데 걸린 총 시간(밀리초)입니다.
LearningOptimizerLogOtherCalls	학습 모듈에 전달된 비컨택 및 비수락 이벤트 수입니다.
LearningOptimizerLogOtherTrackingDuration	학습 모듈에서 기타 이벤트(비컨택 및 비수락)를 로그하는 데 걸린 기간(밀리초)입니다.
LearningOptimizerNonRandomCalls	구성된 학습 구현이 적용된 횟수입니다.
LearningOptimizerRandomCalls	구성된 학습 구현을 무시하고 무작위 선택이 적용된 횟수입니다.
LearningOptimizerRecommendCalls	학습 모듈에 전달된 권장 요청 수입니다.
LearningOptimizerRecommendDuration	학습 권장 논리에 소모된 총 시간(밀리초)입니다.

표 19. 기본 오퍼 통계

속성	설명
LoadDefaultOffersDuration	기본 오퍼 로딩의 경과된 시간입니다.
DefaultOffersCalls	기본 오퍼 로드 횟수입니다.

JMX 작업

다음 테이블은 JMX 모니터링에 사용 가능한 작업을 설명합니다.

그룹	속성	설명
로거 구성	activateDebug	Interact/conf/interact_log4j.properties에 정의된 로그 파일의 로그 수준을 디버그로 설정합니다.
로거 구성	activateError	Interact/conf/interact_log4j.properties에 정의된 로그 파일의 로그 수준을 오류로 설정합니다.
로거 구성	activateFatal	Interact/conf/interact_log4j.properties에 정의된 로그 파일의 로그 수준을 심각으로 설정합니다.
로거 구성	activateInfo	Interact/conf/interact_log4j.properties에 정의된 로그 파일의 로그 수준을 정보로 설정합니다.
로거 구성	activateTrace	Interact/conf/interact_log4j.properties에 정의된 로그 파일의 로그 수준을 추적으로 설정합니다.
로거 구성	activateWarn	Interact/conf/interact_log4j.properties에 정의된 로그 파일의 로그 수준을 경고로 설정합니다.
로케일	changeLocale	JMX 클라이언트의 로케일을 변경합니다. Interact 지원 로케일은 de, en, es, fr입니다.
ContactResponseHistory ETLMonitor	reset	모든 카운터를 재설정합니다.
기본 오퍼 통계	updatePollPeriod	defaultOfferUpdatePollPeriod를 업데이트합니다. 이 값(초)은 캐시의 기본 오퍼를 업데이트하기 전에 대기할 기간을 시스템에 알립니다. -1로 설정하면 시스템이 시작 시 기본 오퍼 수만 읽습니다.

제 7 장 IBM Unica Interact API의 클래스 및 메소드

다음 섹션에는 Interact API에 대해 작업하기 전에 알고 있어야 할 요구 사항과 기타 상세 정보가 나열됩니다.

참고: 이 섹션은 사용자가 접점, Java 프로그래밍 언어, Java 기반 API에 대한 작업에 익숙하다고 가정합니다.

Interact API에는 HTTP의 Java 직렬화를 사용하는 Java 클라이언트 어댑터가 있습니다. Interact는 SOAP 클라이언트를 지원할 WSDL도 제공합니다. WSDL은 Java 클라이언트 어댑터와 동일한 기능 세트를 표시하므로 예를 제외한 다음 섹션이 여전히 적용됩니다.

Interact API 클래스

Interact API는 `InteractAPI` 클래스에 기반을 두고 있습니다. 6개의 지원되는 인터페이스가 있습니다.

- `AdvisoryMessage`
- `BatchResponse`
- `NameValuePair`
- 오퍼
- `OfferList`
- 응답

이 인터페이스에는 3개의 지원되는 구체적 클래스가 있습니다. 다음의 두 가지 구체적 클래스를 인스턴스화한 후 Interact API 메소드에 인수로 전달해야 합니다.

- NameValuePairImpl
- CommandImpl

`AdvisoryMessageCode`라는 세 번째 구체적 클래스는 적용 가능한 경우 서버에서 리턴된 메시지 코드를 구별하는 데 사용되는 상수를 제공할 수 있습니다.

이 섹션의 나머지는 Interact API를 이루는 메소드를 설명합니다.

HTTP의 Java 직렬화 필수 구성 요소

1. Java 직렬화 어댑터에 대해 작업하려면 `CLASSPATH`에 다음 파일을 추가해야 합니다.

   ```
   Interact_Runtime_Environment_Installation_Directory/lib/interact_client.jar
   ```

© Copyright IBM Corp. 2001, 2012

2. 클라이언트와 서버 사이에 전달된 모든 개체는 com.unicacorp.interact.api 패키지에 있습니다. 자세한 내용은 Interact API JavaDoc을 참조하십시오.
3. InteractAPI 클래스의 인스턴스를 얻으려면 정적 메소드 getInstance를 Interact 런타임 서버의 URL과 함께 호출하십시오.

SOAP 필수 구성 요소

중요사항: 성능 테스트 결과 Java 직렬화 어댑터가 생성된 SOAP 클라이언트보다 수행 성능 수준이 훨씬 더 높았습니다. 최상의 성능을 위해서는 가능할 때마다 Java 직렬화 어댑터를 사용하십시오.

SOAP를 사용하여 런타임 서버에 액세스하려면 다음을 수행해야 합니다.

1. 선택한 SOAP 툴킷을 사용하여 Interact API WSDL을 변환하십시오.

   Interact API WSDL은 Interact와 함께 Interact/conf 디렉토리에 설치됩니다.

   WSDL 텍스트는 이 가이드의 끝 부분에 있습니다.
2. 런타임 서버를 설치하고 구성하십시오.

   통합을 완전히 테스트하려면 런타임 서버가 실행되고 있어야 합니다.

SOAP 버전

Interact는 axis2 1.3을 Interact 런타임 서버의 SOAP 인프라로 사용합니다. axis2 1.3이 지원하는 SOAP 버전에 대한 세부 정보는 다음 웹 사이트를 참조하십시오.

Apache Axis2

Interact는 axis2, XFire, JAX-WS-Ri, DotNet, SOAPUI, IBM RAD SOAP 클라이언트로 테스트되었습니다.

API JavaDoc

이 가이드 외에도 Interact API에 대한 JavaDoc이 런타임 서버와 함께 설치됩니다. JavaDoc은 Interact/docs/apiJavaDoc 디렉토리에 참조용으로 설치됩니다.

API 예 정보

이 가이드의 모든 예는 HTTP 어댑터의 Java 직렬화를 사용하여 생성되었습니다. SOAP를 사용 중인 경우 WSDL에서 생성된 클래스가 선택한 SOAP 툴킷 및 옵션에 따라 다를 수 있으므로 이 예가 사용자의 환경에서 정확히 동일하게 작용하지 않을 수도 있습니다.

세션 데이터에 대한 작업

startSession 메소드로 세션을 시작하면 세션 데이터가 메모리로 로드됩니다. 세션 기간 내내 세션 데이터(정적 프로파일 데이터의 수퍼세트)를 읽고 쓸 수 있습니다. 세션에는 다음 데이터가 포함됩니다.

- 정적 프로파일 데이터
- 세그먼트 지정
- 실시간 데이터
- 오퍼 권장사항

모든 세션 데이터는 endSession 메소드를 호출하거나 sessionTimeout 시간이 경과할 때까지 사용 가능합니다. 세션이 종료된 후에는 모든 데이터가 명시적으로 컨택 또는 응답 기록에 저장되거나 일부 다른 데이터베이스 테이블은 손실됩니다.

데이터는 이름-값 쌍 세트로 저장됩니다. 데이터베이스 테이블에서 데이터를 읽는 경우 이름은 테이블의 열입니다.

Interact API에 대해 작업할 때 이 이름-값 쌍을 생성할 수 있습니다. 전역 영역에서 모든 이름-값 쌍을 선언할 필요는 없습니다. 새 이벤트 매개변수를 이름-값 쌍으로 설정하면 런타임 환경이 이름-값 쌍을 세션 데이터에 추가합니다. 예를 들어, postEvent 메소드와 함께 이벤트 매개변수를 사용하면 프로파일 데이터에서 이벤트 매개변수가 사용 불가능한 경우에도 런타임 환경이 세션 데이터에 이벤트 매개변수를 추가합니다. 이 데이터는 세션 데이터에만 존재합니다.

세션 데이터를 언제든지 덮어쓸 수 있습니다. 예를 들어, 고객 프로파일의 일부에 creditScore가 포함된 경우 사용자 정의 유형 NameValuePair를 사용하여 이벤트 매개변수를 전달할 수 있습니다. NameValuePair 클래스에 setName 및 setValueAsNumeric 메소드를 사용하여 값을 변경할 수 있습니다. 이름을 일치시켜야 합니다. 세션 데이터 내에서 이름은 대소문자를 구분하지 않습니다. 따라서 creditscore 또는 CrEdItScOrE는 모두 creditScore를 덮어씁니다.

세션 데이터에 쓴 마지막 데이터만 보관됩니다. 예를 들어, startSession은 lastOffer의 값에 대한 프로파일 데이터를 로드합니다. postEvent 메소드는 lastOffer를 덮어씁니다. 그런 다음 두 번째 postEvent 메소드는 lastOffer를 덮어씁니다. 런타임 환경은 세션 데이터의 두 번째 postEvent 메소드로 쓴 데이터만 보관합니다.

세션이 종료될 때 대화식 플로차트에서 데이터베이스 테이블에 데이터를 쓸 스냅샷 프로세스를 사용하는 것과 같이 특별히 고려하지 않으면 데이터가 손실됩니다. 스냅샷 프로세스를 사용할 계획이면 데이터베이스의 제한사항에 맞게 이름을 지정해야 함을 기억하십시오. 예를 들어, 열 이름에 256자만 허용된 경우에는 이름-값 쌍의 이름이256자를 초과하면 안됩니다.

InteractAPI 클래스 정보

InteractAPI 클래스는 런타임 서버와 접점을 통합하기 위해 사용하는 메소드를 포함합니다. Interact API의 다른 모든 클래스와 메소드는 이 클래스의 메소드를 지원합니다.

Interact 런타임 환경 설치의 lib 디렉토리에 있는 interact_client.jar에 대해 구현을 컴파일해야 합니다.

endSession

endSession(String *sessionID*)

endSession 메소드는 런타임 세션 종료를 표시합니다. 런타임 서버는 이 메소드를 수신하면 기록에 로그하고 메모리 등을 지웁니다.

- **sessionID** - 세션을 식별하는 고유 문자열입니다.

endSession 메소드가 호출되면, 런타임 세션이 시간 초과됩니다. sessionTimeout 등록 정보를 사용하여 제한 시간 기간을 구성할 수 있습니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 endSession에 응답합니다.

- SessionID
- ApiVersion
- StatusCode
- AdvisoryMessages

예제

다음 예제는 endSession 메소드 및 응답 구문 분석 방법을 표시합니다. sessionId는 이 세션을 시작한 startSession 호출에서 사용되는 세션을 식별할 동일한 문자열입니다.

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

executeBatch

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

executeBatch 메소드를 사용하여 런타임 서버에 대한 단일 요청으로 몇 개의 메소드를 실행할 수 있습니다.

- **sessionID** - 세션 ID를 식별하는 문자열입니다. 세션 ID는 이 메소드 호출에 의해 실행되는 모든 명령에 사용됩니다.
- **commandImpl[]** - 수행할 각 명령에 대해 하나씩 CommandImpl 개체의 배열입니다.

이 메소드 호출의 결과는 Command 배열의 각 메소드를 명시적으로 호출하는 것과 같습니다. 이 메소드는 런타임 서버에 대한 실제 요청 수를 최소화합니다. 런타임 서버는 각 메소드를 순차로 실행합니다. 각 호출에 대해 오류 또는 경고가 해당 메소드 호출에 대응하는 Response 개체에 캡처됩니다. 오류가 발생하면, executeBatch가 일괄처리로 나머지 호출을 계속합니다. 메소드 실행 결과 오류가 발생하면, BatchResponse 개체의 최상위 상태가 해당 오류를 반영합니다. 오류가 발생하지 않은 경우, 최상위 상태는 발생했을 수도 있는 경고를 반영합니다. 경고가 발생하지 않은 경우, 최상위 상태는 성공한 일괄처리 실행을 반영합니다.

리턴값

런타임 서버는 BatchResponse 개체로 executeBatch에 응답합니다.

예제

다음 예제는 단일 executeBatch 호출로 getOffer 및 postEvent 메소드를 모두 호출하는 방법과 응답 처리 방법에 대한 제안을 표시합니다.

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

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

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

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

```
  postEventCommand,
};

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

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

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

getInstance

```
getInstance(String URL)
```

getInstance 메소드는 지정된 런타임 서버와 통신하는 Interact API 인스턴스를 작성합니다.

중요사항: Interact API를 사용하여 작성하는 모든 응용 프로그램은 getInstance를 호출하여 URL 매개변수가 지정한 런타임 서버에 맵핑되는 InteractAPI 개체를 인스턴스화해야 합니다.

서버 그룹의 경우, 로드 밸런서를 사용 중인 경우 로드 밸런서와 함께 구성하는 호스트 이름 및 포트를 사용하십시오. 로드 밸런서가 없으면, 사용 가능한 런타임 서버 간에 순환할 논리를 포함시켜야 합니다.

이 메소드는 HTTP 어댑터를 통한 Java 직렬화에만 적용할 수 있습니다. SOAP WSDL에는 해당 메소드가 정의되어 있지 않습니다. 각 SOAP 클라이언트 구현마다 엔드포인트 URL을 설정하는 자체 방법이 있습니다.

- **URL** - 런타임 서버의 URL을 식별하는 문자열입니다(예: http://localhost:7001/Interact/servlet/InteractJSService).

리턴값

런타임 서버는 InteractAPI를 리턴합니다.

예제

다음 예제는 접점과 동일한 시스템에서 실행 중인 런타임 서버 인스턴스를 가리키는 InteractAPI 개체 인스턴스화 방법을 표시합니다.

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

getOffers

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

getOffers 메소드를 사용하여 런타임 서버로부터 오퍼를 요청할 수 있습니다.

- **sessionID** - 현재 세션을 식별하는 문자열입니다.
- **interactionPoint** - 이 메소드가 참조하는 상호작용 지점의 이름을 식별하는 문자열입니다.

  참고: 이 이름은 대화식 채널에 정의된 상호작용 지점의 이름과 정확히 일치해야 합니다.
- **numberOfOffers** - 요청한 오퍼 수를 정의하는 정수입니다.

getOffers 메소드는 실행 전에 segmentationMaxWaitTimeInMS 등록 정보에 정의된 시간(밀리초) 동안 모든 재세그먼트가 완료되기를 기다립니다. 따라서 getOffers 호출 직전에 재세그먼트 또는 setAudience 메소드를 트리거하는 postEvent 메소드를 호출하면 지연이 있습니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 getOffers에 응답합니다.

- AdvisoryMessages
- ApiVersion
- OfferList
- SessionID
- StatusCode

예제

다음 예제는 개요 페이지 배너 1 상호작용 지점에 대한 단일 오퍼 요청 및 응답 처리 방법을 표시합니다.

sessionId는 이 세션을 시작한 startSession 호출에서 사용되는 런타임 세션을 식별할 동일한 문자열입니다.

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

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

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

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

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

getOffersForMultipleInteractionPoints

`getOffersForMultipleInteractionPoints(String sessionID, String requestStr)`

`getOffersForMultipleInteractionPoints` 메소드를 사용하여 중복 데이터가 삭제된 다중 IP에 대해 런타임 서버로부터 오퍼를 요청할 수 있습니다.

- **sessionID** - 현재 세션을 식별하는 문자열입니다.
- **requestStr** - `GetOfferRequest` 개체의 배열을 제공하는 문자열입니다.

  각 `GetOfferRequest` 개체는 다음을 지정합니다.

  – **ipName** - 개체가 오퍼를 요청하는 상호작용 지점(IP) 이름

  – **numberRequested** - 지정된 IP에 대해 필요한 고유 오퍼 수

  – **offerAttributes** - `OfferAttributeRequirements` 인스턴스를 사용하여 전달된 오퍼의 속성에 대한 요구 사항

  – **duplicationPolicy** - 전달될 오퍼에 대한 중복 제거 정책 ID

중복 정책은 단일 메소드 호출의 여러 상호작용 지점에서 중복된 오퍼가 반환될 지 여부를 판별합니다(개별 상호작용 지점 *내에서* 중복된 오퍼는 반환되지 않음). 현재 두 개의 중복 정책이 지원됩니다.

- NO_DUPLICATION(ID 값 = 1). 이전 `GetOfferRequest` 인스턴스에 포함된 오퍼가 이 `GetOfferRequest` 인스턴스에 포함되지 않습니다(즉, Interact가 중복 데이터 삭제를 적용함).
- ALLOW_DUPLICATION(ID 값 = 2). 이 `GetOfferRequest` 인스턴스에 지정된 요구 사항을 충족시키는 오퍼가 포함됩니다. 이전 `GetOfferRequest` 인스턴스에 포함된 오퍼는 조정되지 않습니다.

배열 매개변수의 요청 순서 역시 오퍼가 배달될 때의 우선 순위순입니다.

예를 들어, 요청의 IP가 IP1, IP2이고 중복된 오퍼가 허용되지 않으며(중복 정책 ID = 1), 각각 두 개의 오퍼를 요청한다고 가정하십시오. Interact가 IP1에 대해서는 오퍼 A, B, C를, IP2에 대해서는 오퍼 A, D를 찾을 경우, 응답은 IP1에 대해서는 오퍼 A, B를, IP2에 대해서는 오퍼 D만 포함합니다.

또한 중복 정책 ID가 1이면, IP를 통해 배달된 우선 순위가 높은 오퍼는 이 IP를 통해 배달되지 않음을 참고하십시오.

`getOffersForMultipleInteractionPoints` 메소드는 실행 전에 `segmentationMaxWaitTimeInMS` 등록 정보에 정의된 시간(밀리초) 동안 모든 재세그먼트가 완료되기를 기다립니다. 따라서 `getOffers` 호출 직전에 재세그먼트 또는 `setAudience` 메소드를 트리거하는 `postEvent` 메소드를 호출하면 지연이 있습니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 `getOffersForMultipleInteractionPoints`에 응답합니다.

- AdvisoryMessages
- ApiVersion
- OfferList 배열
- SessionID
- StatusCode

예제

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

```
Response response = api.getOffersForMultipleInteractionPoints(sessionId,
         requestStr);

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

requestStr 구문은 다음과 같습니다.

```
requests_for_IP[<requests_for_IP]
```

여기서,

```
<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
```

위에 표시된 예제에서 requestForIP1 ({IP1,5,1,(5,attr1=1|numeric; attr2=value2|string, (3,attr3=value3|string)(3,attr4=4|numeric))})은 IP1이라는 상호작용 지점의 경우 이 동일한 메소드 호출 동안 다른 상호작용 지점에 대해서도 반환할 수 없는 최대 5개의 고유 오퍼를 전달함을 의미합니다. 해당 5개의 오퍼 모두 attr1이라는 numeric 속성(값이 1이어야 함)을 가져야 하며 attr2라는 string 속성(값이 *value2*여야 함)을 가져야 합니다. 해당 5개의 오퍼 중에서 최대 3개는 attr3이라는 string 속성(값이 *value3*이어야 함)을 가져야 하며 최대 3개는 attr4라는 numeric 속성(값이 *4*여야 함)을 가져야 합니다.

허용되는 속성 유형은 numeric, string, datetime이며 datetime 속성 값 형식은 MM/dd/yyyy HH:mm:ss여야 합니다. 반환된 오퍼를 검색하려면 Response.getAllOfferLists() 메소드를 사용하십시오. 구문 이해를 돕기 위해 setGetOfferRequests의 예제는 Java 개체 사용 중에 선호하는 동일한 GetOfferRequests 인스턴스를 빌드합니다.

getProfile

```
getProfile(String sessionID)
```

getProfile 메소드를 사용하여 접점을 방문하는 방문자에 대한 프로파일 및 임시 정보를 검색할 수 있습니다.

- **sessionID** - 세션 ID를 식별하는 문자열입니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 getProfile에 응답합니다.

- AdvisoryMessages
- ApiVersion
- ProfileRecord
- SessionID
- StatusCode

예제

다음은 getProfile 사용 예제 및 응답 처리 방법입니다.

sessionId는 이 세션을 시작한 startSession 호출에서 사용되는 세션을 식별할 동일한 문자열입니다.

```
response = api.getProfile(sessionId);
/**  Process the response appropriately */
    // check if response is successful or not
    if(response.getStatusCode() == Response.STATUS_SUCCESS)
    {
        System.out.println("getProfile call processed with no warnings or errors");
        // Print the profile - it's just an array of NameValuePair objects
        for(NameValuePair nvp : response.getProfileRecord())
        {
            System.out.println("Name:"+nvp.getName());
            if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
            {
                System.out.println("Value:"+nvp.getValueAsDate());
            }
            else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
            {
                System.out.println("Value:"+nvp.getValueAsNumeric());
            }
            else
            {
                System.out.println("Value:"+nvp.getValueAsString());
            }
        }
    }
    else if(response.getStatusCode() == Response.STATUS_WARNING)
    {
        System.out.println("getProfile call processed with a warning");
    }
    else
    {
        System.out.println("getProfile call processed with an error");
    }
    // For any non-successes, there should be advisory messages explaining why
    if(response.getStatusCode() != Response.STATUS_SUCCESS)
        printDetailMessageOfWarningOrError("getProfile",
   response.getAdvisoryMessages());
```

getVersion

```
getVersion()
```

getVersion 메소드는 현재 구현된 Interact 런타임 서버의 버전을 리턴합니다.

모범 사례는 Interact API를 사용하여 접점을 초기화할 때 이 메소드를 사용하는 것입니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 getVersion에 응답합니다.

- AdvisoryMessages
- ApiVersion
- StatusCode

예제

다음 예제는 getVersion을 호출하고 결과를 처리하는 간단한 방법을 표시합니다.

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

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

postEvent

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

postEvent 메소드를 사용하여 대화식 채널에 정의된 이벤트를 실행할 수 있습니다.

- **sessionID** - 세션 ID를 식별하는 문자열입니다.
- **eventName** - 이벤트 이름을 식별하는 문자열입니다.

  참고: 이벤트 이름은 대화식 채널에 정의된 이벤트 이름과 일치해야 합니다. 이 이름은 대소문자를 구분하지 않습니다.

- **eventParameters** - 이벤트와 함께 전달해야 하는 매개변수를 식별하는 NameValuePairImpl 개체입니다. 이 값은 세션 데이터에 저장됩니다.

  이 이벤트가 재세그먼트를 트리거하는 경우, 대화식 플로차트에 필요한 모든 데이터를 세션 데이터에서 사용할 수 있는지 확인해야 합니다. 이 값이 이전 작업(예: `startSession`이나 `setAudience` 또는 프로파일 테이블 로드)으로 채워지지 않은 경우, 누락된 모든 값에 대해 eventParameter를 포함시켜야 합니다. 예를 들어, 모든 프로파일 테이블을 메모리로 로드하도록 구성한 경우 대화식 플로차트에 필요한 임시 데이터에 대해 NameValuePair를 포함시켜야 합니다.

  두 개 이상의 대상 수준을 사용 중인 경우, 각 대상 수준마다 eventParameters 집합이 다를 가능성이 높습니다. 대상 수준에 대한 올바른 매개변수 집합을 선택하도록 일부 논리를 포함시켜야 합니다.

  중요사항: 이 이벤트가 응답 기록에 로그되면, 오퍼에 대한 처리 코드를 전달해야 합니다. NameValuePair 이름을 `"UACIOfferTrackingCode"`로 정의해야 합니다.

  이벤트당 하나의 처리 코드만 전달할 수 있습니다. 오퍼 컨택에 대한 처리 코드를 전달하지 못하면, Interact가 마지막 오퍼 권장 목록의 모든 오퍼에 대한 오퍼 컨택을 로그합니다. 응답에 대한 처리 코드를 전달하지 못하면, Interact가 오류를 리턴합니다.
- `postEvent` 및 기타 메소드에서 사용되는 기타 몇 개의 예약 매개변수가 있으며, 이 섹션에서 나중에 설명합니다.

컨택 또는 응답 기록에 쓰기 또는 재세그먼트 요청은 응답을 기다리지 않습니다.

`UACIExecuteFlowchartByName` 매개변수를 사용하여 지정하지 않으면 재세그먼트는 현재 대상 수준에 대해 이 대화식 채널과 연관된 모든 대화식 플로차트를 실행합니다. `getOffers` 메소드는 실행 전에 재세그먼트가 완료되기를 기다립니다. 따라서 `getOffers` 호출 직전에 재세그먼트를 트리거하는 `postEvent` 메소드를 호출하면 지연이 있습니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 `postEvent`에 응답합니다.

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

예제

다음 `postEvent` 예제는 재세그먼트를 트리거하는 이벤트에 대한 새 매개변수 보내기 및 응답 처리 방법을 표시합니다.

sessionId는 이 세션을 시작한 startSession 호출에서 사용되는 세션을 식별할 동일한 문자열입니다.

```
String eventName = "SearchExecution";

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

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

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

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

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

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

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

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

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

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

setAudience

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

setAudience 메소드를 사용하여 방문자에 대해 대상 ID 및 수준을 설정할 수 있습니다.

- **sessionID** - 세션 ID를 식별하는 문자열입니다.
- **audienceID** - 대상 ID를 정의하는 NameValuePairImpl 개체의 배열입니다.
- **audienceLevel** - 대상 수준을 정의하는 문자열입니다.
- **parameters** - setAudience와 함께 전달해야 하는 매개변수를 식별하는 NameValuePairImpl 개체입니다. 이 값은 세션 데이터에 저장되며 세그먼트에 사용할 수 있습니다.

  프로파일의 모든 열에 대해 값이 있어야 합니다. 이는 대화식 채널 및 실시간 데이터에 대해 정의된 모든 테이블에 있는 모든 열의 상위 집합입니다. 모든 세션 데이터를 startSession 또는 postEvent로 이미 채운 경우, 새 매개변수를 보내지 않아도 됩니다.

setAudience 메소드는 재세그먼트를 트리거합니다. getOffers 메소드는 실행 전에 재세그먼트가 완료되기를 기다립니다. 따라서 getOffers 호출 직전에 setAudience 메소드를 호출하면 지연이 있습니다.

setAudience 메소드는 대상 ID에 대한 프로파일 데이터도 로드합니다. setAudience 메소드를 사용하여 startSession 메소드가 로드한 동일한 프로파일 데이터를 강제로 다시 로드할 수 있습니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 setAudience에 응답합니다.

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

예제

이 예제의 경우, 대상 수준은 동일하지만 마치 익명의 사용자가 로그인하여 알려지는 것처럼 ID가 변경됩니다.

sessionId 및 audienceLevel은 이 세션을 시작한 startSession 호출에서 사용되는 세션 및 대상 수준을 식별할 동일한 문자열입니다.

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

NameValuePair[] newAudienceId = { custId2 };

/** Parameters can be passed in as well.  For this example, there are no parameters,
* therefore pass in null */
NameValuePair[] noParameters=null;

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

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

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

setDebug

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

setDebug 메소드를 사용하여 세션의 모든 코드 경로에 대해 로깅 상세 수준을 설정할 수 있습니다.

- **sessionID** - 세션 ID를 식별하는 문자열입니다.
- **debug** - 디버그 정보를 활성화하거나 비활성화하는 부울입니다. 유효한 값은 true 또는 false입니다. true이면, Interact가 런타임 서버 로그에 디버그 정보를 로그합니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 setDebug에 응답합니다.

- AdvisoryMessages
- ApiVersion
- SessionID
- StatusCode

예제

다음 예제는 세션 디버그 수준 변경을 표시합니다.

sessionId는 이 세션을 시작한 startSession 호출에서 사용되는 세션을 식별할 동일한 문자열입니다.

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

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

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

startSession

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

startSession 메소드는 런타임 세션을 작성하고 정의합니다. startSession은 다음 작업을 다섯 개까지 트리거할 수 있습니다.

- 런타임 세션 작성
- 대화식 채널에 대해 정의된 테이블 맵핑에 로드하도록 표시된 차원 테이블을 포함하여 현재 대상 수준에 대한 방문자 프로파일 데이터를 런타임 세션으로 로드
- 현재 대상 수준에 대한 대화식 플로차트를 모두 실행하여 세그먼트 트리거
- 오퍼 제외 데이터를 세션으로 로드(enableOfferSuppressionLookup 등록 정보가 true로 설정된 경우)
- 점수 재정의 데이터를 세션으로 로드(enableScoreOverrideLookup 등록 정보가 true로 설정된 경우)

startSession 메소드에 필요한 매개변수는 다음과 같습니다.

- **sessionID** - 세션 ID를 식별하는 문자열입니다. 세션 ID를 정의해야 합니다. 예를 들어, 고객 ID와 타임스탬프를 조합하여 사용할 수 있습니다.

런타임 세션 구성 요소를 정의하려면 세션 ID를 지정해야 합니다. 이 값은 클라이언트가 관리합니다. 클라이언트는 동일한 세션 ID에 대한 모든 메소드 호출을 동기화해야 합니다. 동일한 세션 ID를 사용한 동시 API 호출에 대한 동작은 정의되지 않습니다.

- **relyOnExistingSession** - 이 세션이 새 세션을 사용하는지 또는 기존 세션을 사용하는지 여부를 정의하는 부울입니다. 유효한 값은 `true` 또는 `false`입니다. `true`이면, `startSession` 메소드와 함께 기존 세션 ID를 제공해야 합니다. `false`이면, 새 세션 ID를 제공해야 합니다.

  `relyOnExistingSession`을 `true`로 설정하고 세션이 있는 경우, 런타임 환경은 기존 세션 데이터를 사용하며 데이터를 다시 로드하거나 세그먼트를 트리거하지 않습니다. 세션이 없는 경우, 런타임 환경은 데이터 로드 및 세그먼트 트리거를 포함하여 새 세션을 작성합니다. 접점에 런타임 세션보다 길이가 긴 세션이 있으면 `relyOnExistingSession`을 true로 설정하고 모든 `startSession` 호출에 사용하는 것이 유용합니다. 예를 들어, 웹 사이트 세션은 2시간 동안 지속되지만 런타임 세션은 20분 동안만 지속됩니다.

  동일한 세션 ID를 사용하여 `startSession`을 두 번 호출하는 경우, `relyOnExistingSession`이 false이면 첫 번째 `startSession` 호출의 모든 세션 데이터가 손실됩니다.
- **debug** - 디버그 정보를 활성화하거나 비활성화하는 부울입니다. 유효한 값은 `true` 또는 `false`입니다. true이면, Interact가 런타임 서버 로그에 디버그 정보를 로그합니다. 각 세션마다 개별적으로 디버그 플래그가 설정됩니다. 따라서 개별 세션에 대한 디버그 데이터를 추적할 수 있습니다.
- **interactiveChannel** - 이 세션이 참조하는 대화식 채널 이름을 정의하는 문자열입니다. 이 이름은 Campaign에 정의된 대화식 채널 이름과 정확히 일치해야 합니다.
- **audienceID** - 이름이 대상 ID를 포함하는 테이블의 실제 열 이름과 일치해야 하는 NameValuePairImpl 개체의 배열입니다.
- **audienceLevel** - 대상 수준을 정의하는 문자열입니다.
- **parameters** - `startSession`과 함께 전달해야 하는 매개변수를 식별하는 NameValuePairImpl 개체입니다. 이 값은 세션 데이터에 저장되며 세그먼트에 사용할 수 있습니다.

  동일한 대상 수준에 대해 몇 개의 대화식 플로차트가 있으면, 모든 테이블에 있는 모든 열의 상위 집합을 포함시켜야 합니다. 프로파일 테이블을 로드하도록 런타임을 구성하고 프로파일 테이블에 사용자가 필요로 하는 모든 열이 포함된 경우, 프로파일 테이블의 데이터를 덮어쓰지 않으려면 매개변수를 전달하지 않아도 됩니다. 프로파일 테이블에 필수 열 하위 집합이 포함된 경우, 누락된 열을 매개변수로 포함시켜야 합니다.

audienceID 또는 audienceLevel이 유효하지 않고 relyOnExistingSession이 false이면, startSession 호출에 실패합니다. interactiveChannel이 유효하지 않으면, relyOnExistingSession이 true이건 false이건 startSession이 실패합니다.

relyOnExistingSession이 true이고 동일한 sessionID를 사용하여 두 번째 startSession 호출을 수행하지만 첫 번째 세션이 만료된 경우, Interact는 새 세션을 작성합니다.

relyOnExistingSession이 true이고 동일한 sessionID를 사용하지만 다른 audienceID 또는 audienceLevel을 사용하여 두 번째 startSession 호출을 수행하는 경우, 런타임 서버는 기존 세션에 대해 대상을 변경합니다.

relyOnExistingSession이 true이고 동일한 sessionID를 사용하지만 다른 interactiveChannel을 사용하여 두 번째 startSession 호출을 수행하는 경우, 런타임 서버는 새 세션을 작성합니다.

리턴값

런타임 서버는 다음 속성이 채워진 Response 개체로 startSession에 응답합니다.

- AdvisoryMessages(StatusCode가 0과 같지 않음)
- ApiVersion
- SessionID
- StatusCode

예제

다음 예제는 startSession을 호출하는 한 가지 방법을 표시합니다.

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

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

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

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

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

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

/** Specifying the parameters (optional) */
NameValuePair[] initialParameters = { parm1,
 parm2,
 parm3,
 parm4,
 parm5,
 parm6
 };

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

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

processStartSessionResponse는 startSession이 리턴한 Response 개체를 처리하는 메소드입니다.

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

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

예약된 매개변수

Interact API에 사용되는 몇 개의 예약된 매개변수가 있습니다. 일부는 런타임 서버에 필요하고 나머지는 추가 기능에 사용할 수 있습니다.

postEvent 기능

기능	매개변수	설명
사용자 정의 테이블에 로그	UACICustomLoggerTableName	런타임 테이블 데이터 소스의 테이블 이름입니다. 이 매개변수에 유효한 테이블 이름을 제공하면, 런타임 환경은 선택한 테이블에 모든 세션 데이터를 기록합니다. 세션 데이터 NameValuePair와 일치하는 테이블의 모든 열 이름이 채워집니다. 런타임 환경은 세션 이름-값 쌍과 일치하지 않는 모든 열을 Null로 채웁니다. customLogger 구성 등록 정보를 사용하여 데이터베이스에 기록하는 프로세스를 관리할 수 있습니다.
다중 응답 유형	UACILogToLearning	값이 1 또는 0인 정수입니다. 1은 런타임 환경이 학습에 대한 수락으로 이벤트를 로그해야 함을 표시합니다. 0은 런타임 환경이 학습에 대한 이벤트를 로그하지 않아야 함을 표시합니다. 이 매개변수를 사용하여 학습에 영향을 주지 않고 여러 응답 유형을 로깅하는 몇 개의 postEvent 메소드를 작성할 수 있습니다. 컨택, 수락 또는 거부를 로그하도록 설정된 이벤트에 대해서는 이 매개변수를 정의하지 않아도 됩니다. 이 매개변수를 UACIResponseTypeCode와 함께 사용해야 합니다. UACILOGTOLEARNING을 정의하지 않은 경우, 런타임 환경은 이벤트가 컨택, 수락 또는 거부 로그를 트리거하지 않으면 기본값 0을 가정합니다.
	UACIResponseTypeCode	응답 유형 코드를 나타내는 값입니다. 이 값은 UA_UsrResponseType 테이블의 올바른 항목이어야 합니다.
응답 추적	UACIOfferTrackingCode	오퍼에 대한 처리 코드입니다. 이벤트가 컨택 또는 기록에 로그하는 경우 이 매개변수를 정의해야 합니다. 이벤트당 하나의 처리 코드만 전달할 수 있습니다. 오퍼 컨택에 대한 처리 코드를 전달하지 못할 경우, 런타임 환경은 마지막 오퍼 권장 목록의 모든 오퍼에 대한 오퍼 컨택을 로그합니다. 응답에 대한 처리 코드를 전달하지 못할 경우, 런타임 환경은 오류를 리턴합니다. 교차 세션 응답 추적을 구성하면, UACIOfferTrackingcodeType 매개변수를 사용하여 처리 코드가 아닌 사용할 다른 추적 코드 유형을 정의할 수 있습니다.
교체 세션 응답 추적	UACIOfferTrackingCodeType	추적 코드 유형을 정의하는 숫자입니다. 1은 기본 처리 코드이고 2는 오퍼 코드입니다. 모든 코드는 UACI_TrackingType 테이블의 올바른 항목이어야 합니다. 이 테이블에 기타 사용자 정의 코드를 추가할 수 있습니다.

기능	매개변수	설명
특정 플로차트 실행	UACIExecuteFlowchartByName	현재 대상 수준에 대해 모든 플로차트를 실행하지 않고 세그먼트를 트리거하는 메소드(startSession, setAudience 또는 재세그먼트를 트리거하는 postEvent)에 대해 이 매개변수를 정의하면, Interact가 이름 지정된 플로차트만 실행합니다. 파이프(|) 문자로 구분된 플로차트 목록을 제공할 수 있습니다.

런타임 환경 예약 매개변수

런타임 환경에서 사용하는 예약 매개변수는 다음과 같습니다. 이러한 이름을 이벤트 매개변수에 사용하지 마십시오.

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

AdvisoryMessage 클래스 정보

advisoryMessage 클래스는 권고 메시지 개체를 정의하는 메소드를 포함합니다. 권고 메시지 개체는 Response 개체에 포함되어 있습니다. InteractAPI의 모든 메소드는 Response 개체를 리턴합니다(batchResponse 개체를 리턴하는 executeBatch 메소드는 예외임). 오류나 경고가 있는 경우 Interact 서버는 권고 메시지 개체를 채웁니다. 권고 메시지 개체는 다음 속성을 포함합니다.

- **DetailMessage** - 권고 메시지에 대한 자세한 설명입니다. 모든 권고 메시지에 이 속성이 사용 가능하지는 않습니다. 사용 가능한 경우 DetailMessage가 현지화되어 있지 않을 수 있습니다.
- **Message** - 권고 메시지에 대한 간단한 설명입니다.
- **MessageCode** - 권고 메시지의 코드 번호입니다.
- **StatusLevel** - 권고 메시지 심각도의 코드 번호입니다.

getAdvisoryMessages 메소드를 사용하여 advisoryMessage 개체를 검색합니다.

getDetailMessage

```
getDetailMessage()
```

getDetailMessage 메소드는 Advisory Message 개체에 대한 자세하고 상세한 설명을 리턴합니다.

모든 메시지에 자세한 메시지가 있는 것은 아닙니다.

리턴값

Advisory Message 개체는 문자열을 리턴합니다.

예제

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

getMessage

```
getMessage()
```

getMessage 메소드는 Advisory Message 개체에 대한 간단한 설명을 리턴합니다.

리턴값

Advisory Message 개체는 문자열을 리턴합니다.

예제

다음 메소드는 AdvisoryMessage 개체의 메시지 및 자세한 메시지를 인쇄합니다.

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

getMessageCode

```
getMessageCode()
```

getMessageCode 메소드는 상태 수준이 2이면(STATUS_LEVEL_ERROR) Advisory Message 개체와 연관된 내부 오류 코드를 리턴합니다.

리턴값

AdvisoryMessage 개체는 정수를 리턴합니다.

예제

다음 메소드는 AdvisoryMessage 개체의 메시지 코드를 인쇄합니다.

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

getStatusLevel

```
getStatusLevel()
```

getStatusLevel 메소드는 Advisory Message 개체의 상태 수준을 리턴합니다.

리턴값

Advisory Message 개체는 정수를 리턴합니다.

- 0 - STATUS_LEVEL_SUCCESS - 호출된 메소드가 오류 없이 완료되었습니다.
- 1 - STATUS_LEVEL_WARNING - 호출된 메소드가 완료되고 한 개 이상의 경고가 표시됩니다(오류 없음).
- 2 - STATUS_LEVEL_ERROR - 호출된 메소드가 완료되지 않았으며 한 개 이상의 오류가 있습니다.

예제

다음 메소드는 AdvisoryMessage 개체의 상태 수준을 인쇄합니다.

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

AdvisoryMessageCode 클래스 정보

advisoryMessageCode 클래스는 권고 메시지 코드를 정의하는 메소드를 포함합니다. getMessageCode 메소드로 권고 메시지 코드를 검색합니다.

권고 메시지 코드

코드	설명
1	INVALID_SESSION_ID - 세션 ID가 유효한 세션을 참조하지 않습니다.

코드	설명
2	ERROR_TRYING_TO_ABORT_SEGMENTATION - endSession 중에 세그먼트 중단을 시도하는 동안 오류가 발생했습니다.
3	INVALID_INTERACTIVE_CHANNEL - 대화식 채널에 대해 전달된 인수가 유효한 대화식 채널을 참조하지 않습니다.
4	INVALID_EVENT_NAME - 이벤트에 대해 전달된 인수가 현재 대화식 채널에 유효한 이벤트를 참조하지 않습니다.
5	INVALID_INTERACTION_POINT - 상호작용 지점에 대해 전달된 인수가 현재 대화식 채널에 유효한 상호작용 지점을 참조하지 않습니다.
6	ERROR_WHILE_MAKING_INITIAL_SEGMENTATION_REQUEST - 세그먼트 요청을 제출할 때 오류가 발생했습니다.
7	SEGMENTATION_RUN_FAILED - 세그먼트가 부분적으로 실행되었지만 오류가 발생했습니다.
8	PROFILE_LOAD_FAILED - 프로파일 또는 차원 테이블 로드에 실패했습니다.
9	OFFER_SUPPRESSION_LOAD_FAILED - 오퍼 제외 테이블 로드에 실패했습니다.
10	COMMAND_METHOD_UNRECOGNIZED - executeBatch 내에서 명령에 대해 지정된 명령 메소드가 유효하지 않습니다.
11	ERROR_TRYING_TO_POST_EVENT_PARAMETERS - 이벤트 매개변수를 게시하는 동안 오류가 발생했습니다.
12	LOG_SYSTEM_EVENT_EXCEPTION - 로깅을 위해 시스템 이벤트(세션 종료, 오퍼 가져오기, 프로파일 가져오기, 대상 설정, 디버그 설정 또는 세션 시작)를 제출할 때 예외가 발생했습니다.
13	LOG_USER_EVENT_EXCEPTION - 로깅을 위해 사용자 이벤트를 제출할 때 예외가 발생했습니다.
14	ERROR_TRYING_TO_LOOK_UP_EVENT - 이벤트 이름을 검색할 때 오류가 발생했습니다.
15	ERROR_TRYING_TO_LOOK_UP_INTERACTIVE_CHANNEL - 대화식 채널 이름을 검색할 때 오류가 발생했습니다.
16	ERROR_TRYING_TO_LOOK_UP_INTERACTION_POINT - 상호작용 지점 이름을 검색할 때 오류가 발생했습니다.
17	RUNTIME_EXCEPTION_ENCOUNTERED - 예상치 않은 런타임 예외가 발생했습니다.
18	ERROR_TRYING_TO_EXECUTE_ASSOCIATED_ACTION - 연관된 작업(재세그먼트 트리거, 오퍼 컨택 로그, 오퍼 수락 로그 또는 오퍼 거부 로그)을 실행하는 중에 오류가 발생했습니다.
19	ERROR_TRYING_RUN_FLOWCHART - 플로차트를 실행하는 중에 오류가 발생했습니다.
20	FLOWCHART_FAILED - 플로차트가 실패했습니다.
21	FLOWCHART_ABORTED - 플로차트가 중단되었습니다.
22	FLOWCHART_NEVER_RUN - 플로차트가 실행되지 않습니다.
23	FLOWCHART_STILL_RUNNING - 플로차트가 여전히 실행 중입니다.
24	ERROR_WHILE_READING_PARAMETERS - 매개변수를 읽는 동안 오류가 발생했습니다.
25	ERROR_WHILE_LOADING_RECOMMENDED_OFFERS - 권장되는 오퍼를 로드하는 동안 오류가 발생했습니다.

코드	설명
26	ERROR_WHILE_LOGGING_DEFAULT_TEXT_STATISTICS - 기본 텍스트 통계(표시된 상호작용 지점에 대한 기본 문자열 횟수)를 로깅하는 동안 오류가 발생했습니다.
27	SCORE_OVERRIDE_LOAD_FAILED - 점수 재정의 테이블 로드에 실패했습니다.
28	NULL_AUDIENCE_ID - 대상 ID가 비어 있습니다.
29	UNRECOGNIZED_AUDIENCE_LEVEL - 알 수 없는 대상 수준입니다.
30	MISSING_AUDIENCE_FIELD - 대상 필드가 누락되었습니다.
31	INVALID_AUDIENCE_FIELD_TYPE - 유효하지 않은 대상 필드 유형입니다.
32	UNSUPPORTED_AUDIENCE_FIELD_TYPE - 지원되지 않는 대상 필드 유형입니다.
33	TIMEOUT_REACHED_ON_GET_OFFERS_CALL - getOffers 호출이 오퍼를 반환하지 않고 시간 초과되었습니다.
34	INTERACT_INITIALIZATION_NOT_COMPLETED_SUCCESSFULLY - 런타임 초기화가 완료되지 않았습니다.
35	SESSION_ID_UNDEFINED - 세션 ID가 정의되지 않았습니다.
36	INVALID_NUMBER_OF_OFFERS_REQUESTED - 요청한 오퍼 수가 유효하지 않습니다.
37	NO_SESSION_EXIST_BUT_WILL_CREATE_NEW_ONE
38	AUDIENCE_ID_NOT_FOUND_IN_PROFILE_TABLE
39	LOG_CUSTOM_LOGGER_EVENT_EXCEPTION - 사용자 정의 로깅 데이터 이벤트를 제출할 때 예외가 발생했습니다.
40	SPECIFIED_FLOWCHART_FOR_EXECUTION_DOES_NOT_EXIST
41	AUDIENCE_NOT_DEFINED_IN_CONFIGURATION

BatchResponse 클래스 정보

BatchResponse 클래스는 executeBatch 메소드의 결과를 정의하는 메소드를 포함합니다. Batch Response 개체는 다음 속성을 포함합니다.

- **BatchStatusCode** - executeBatch 메소드로 요청된 모든 응답의 최고 상태 코드 값입니다.
- **Responses** - executeBatch 메소드로 요청된 Response 개체의 배열입니다.

getBatchStatusCode

```
getBatchStatusCode()
```

getBatchStatusCode 메소드는 executeBatch 메소드에 의해 실행된 명령 배열에서 최상위 상태 코드를 리턴합니다.

리턴값

getBatchStatusCode 메소드는 정수를 리턴합니다.

- 0 - STATUS_SUCCESS - 호출된 메소드가 오류 없이 완료되었습니다.
- 1 - STATUS_WARNING - 호출된 메소드가 완료되고 한 개 이상의 경고가 표시됩니다(오류 없음).

- 2 - STATUS_ERROR - 호출된 메소드가 완료되지 않았으며 한 개 이상의 오류가 있습니다.

예제

다음 코드 샘플은 BatchStatusCode 검색 방법 예제를 제공합니다.

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

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

getResponses

```
getResponses()
```

getResponses 메소드는 executeBatch 메소드에 의해 실행된 명령 배열에 대응하는 Response 개체의 배열을 리턴합니다.

리턴값

getResponses 메소드는 Response 개체의 배열을 리턴합니다.

예제

다음 예제는 응답을 모두 선택하고 명령이 실패한 경우 권고 메시지를 인쇄합니다.

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

명령 인터페이스 정보

executeBatch 메소드는 명령 인터페이스를 구현하는 개체 배열로 전달해야 합니다. 기본 구현, CommandImpl을 사용하여 명령 개체를 전달해야 합니다.

다음 테이블에는 명령, 명령이 표시하는 InteractAPI 클래스의 메소드, 각 명령에 사용해야 하는 명령 인터페이스 메소드가 나열됩니다. executeBatch 메소드에 세션 ID가 이미 포함되어 있으므로 세션 ID를 포함할 필요는 없습니다.

명령	상호작용 API 메소드	명령 인터페이스 메소드
COMMAND_ENDSESSION	endSession	없음
COMMAND_GETOFFERS	getOffers	• setInteractionPoint • setNumberRequested
COMMAND_GETPROFILE	getProfile	없음
COMMAND_GETVERSION	getVersion	없음
COMMAND_POSTEVENT	postEvent	• setEvent • setEventParameters
COMMAND_SETAUDIENCE	setAudience	• setAudienceID • setAudienceLevel • setEventParameters
COMMAND_SETDEBUG	setDebug	setDebug
COMMAND_STARTSESSION	startSession	• setAudienceID • setAudienceLevel • setDebug • setEventParameters • setInteractiveChannel • setRelyOnExistingSession

setAudienceID

setAudienceID(*audienceID*)

setAudienceID 메소드는 setAudience 및 startSession 명령에 대해 AudienceID를 정의합니다.

- **audienceID** - AudienceID를 정의하는 NameValuePair 개체의 배열입니다.

리턴값

없음.

예제

다음 예제는 startSession 및 setAudience를 호출하는 executeBatch 메소드에서 발췌한 것입니다.

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

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

setAudienceLevel

setAudienceLevel(*audienceLevel*)

setAudienceLevel 메소드는 setAudience 및 startSession 명령에 대해 대상 수준을 정의합니다.

- *audienceLevel* - 대상 수준을 포함하는 문자열입니다.

  중요사항: *audienceLevel* 이름은 Campaign에 정의된 대상 수준 이름과 정확히 일치해야 합니다. 대소문자를 구분합니다.

리턴값

없음.

예제

다음 예제는 startSession 및 setAudience를 호출하는 executeBatch 메소드에서 발췌한 것입니다.

```
String audienceLevel="Customer";
. . .
Command startSessionCommand = new CommandImpl();
startSessionCommand.setAudienceID(initialAudienceId);
```

```
. . .
Command setAudienceCommand = new CommandImpl();
setAudienceCommand.setAudienceLevel(audienceLevel);
. . .
/** Build command array */
Command[] commands =
    {
        startSessionCommand,
        setAudienceCommand,
    };
/** Make the call */
    BatchResponse batchResponse = api.executeBatch(sessionId, commands);

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

setDebug

setDebug(*debug*)

setDebug 메소드는 startSession 명령의 디버그 수준을 정의합니다. true이면, 런타임 서버가 런타임 서버 로그에 디버그 정보를 로그합니다. false이면, 런타임 서버가 디버그 정보를 로그하지 않습니다. 각 세션마다 개별적으로 디버그 플래그가 설정됩니다. 따라서 개별 런타임 세션에 대한 디버그 데이터를 추적할 수 있습니다.

- **debug** - 부울(true 또는 false).

리턴값

없음.

예제

다음 예제는 startSession 및 setDebug를 호출하는 executeBatch 메소드에서 발췌한 것입니다.

```
boolean initialDebugFlag=true;
boolean newDebugFlag=false;
. . .
/* build the startSession command */
Command startSessionCommand = new CommandImpl();
startSessionCommand.setDebug(initialDebugFlag);
. . .

/* build the setDebug command */
Command setDebugCommand = new CommandImpl();
setDebugCommand.setMethodIdentifier(Command.COMMAND_SETDEBUG);
setDebugCommand.setDebug(newDebugFlag);

/** Build command array */
Command[] commands =
    {
        startSessionCommand,
        setDebugCommand,
    };
```

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

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

setEvent

```
setEvent(event)
```

setEvent 메소드는 postEvent 명령에서 사용되는 이벤트 이름을 정의합니다.

- **event** - 이벤트 이름을 포함하는 문자열입니다.

중요사항: *event* 이름은 대화식 채널에 정의된 이벤트 이름과 정확히 일치해야 합니다. 대소문자를 구분합니다.

리턴값

없음.

예제

다음 예제는 postEvent를 호출하는 executeBatch 메소드에서 발췌한 것입니다.

```
String eventName = "SearchExecution";

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

setEventParameters

```
setEventParameters(eventParameters)
```

setEventParameters 메소드는 postEvent 명령에서 사용되는 이벤트 매개변수를 정의합니다. 이 값은 세션 데이터에 저장됩니다.

- **eventParameters** - 이벤트 매개변수를 정의하는 NameValuePair 개체의 배열입니다.

예를 들어, 이벤트가 컨택 기록에 오퍼를 로깅하는 경우 오퍼 처리 코드를 포함시켜야 합니다.

리턴값

없음.

예제

다음 예제는 postEvent를 호출하는 executeBatch 메소드에서 발췌한 것입니다.

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

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

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

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

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

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

NameValuePair[] postEventParameters = { parmB1,
 parmB2,
 parmB3,
 parmB4,
 parmB5,
 parmB6
  };
. . .
Command postEventCommand = new CommandImpl();
postEventCommand.setMethodIdentifier(Command.COMMAND_POSTEVENT);
postEventCommand.setEventParameters(postEventParameters);
postEventCommand.setEvent(eventName);
```

setGetOfferRequests

setGetOfferRequests(*numberRequested*)

setGetOfferRequests 메소드는 getOffersForMultipleInteractionPoints 명령에서 사용되는 오퍼 수신을 위한 매개변수를 설정합니다.

- **numberRequested** - 오퍼 수신을 위한 매개변수를 정의하는 GetOfferRequest 개체의 배열입니다.

리턴값

없음.

예제

다음 예제는 setGetOfferRequests를 호출하는 GetOfferRequest 메소드에서 발췌한 것입니다.

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

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

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

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

setInteractiveChannel

setInteractiveChannel(*interactiveChannel*)

setInteractiveChannel 메소드는 startSession 명령에서 사용되는 대화식 채널 이름을 정의합니다.

- **interactiveChannel** - 대화식 채널 이름을 포함하는 문자열입니다.

중요사항: *interactiveChannel* 이름은 Campaign에 정의된 대화식 채널 이름과 정확히 일치해야 합니다. 대소문자를 구분합니다.

리턴값

없음.

예제

다음 예제는 startSession을 호출하는 executeBatch 메소드에서 발췌한 것입니다.

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

setInteractionPoint

setInteractionPoint(*interactionPoint*)

setInteractionPoint 메소드는 getOffers 및 postEvent 명령에서 사용되는 상호작용 지점의 이름을 정의합니다.

- **interactionPoint** - 상호작용 지점 이름을 포함하는 문자열입니다.

중요사항: *interactionPoint*는 대화식 채널에 정의된 상호작용 지점의 이름과 정확히 일치해야 합니다. 대소문자를 구분합니다.

리턴값

없음.

예제

다음 예제는 getOffers를 호출하는 executeBatch 메소드에서 발췌한 것입니다.

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

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

setMethodIdentifier

setMethodIdentifier(*methodIdentifier*)

setMethodIdentifier 메소드는 Command 개체에 포함된 명령 유형을 정의합니다.

- **methodIdentifier** - 명령 유형을 포함하는 문자열입니다.

  유효한 값은 다음과 같습니다.

  - **COMMAND_ENDSESSION** - endSession 메소드를 나타냅니다.
  - **COMMAND_GETOFFERS** - getOffers 메소드를 나타냅니다.

- **COMMAND_GETPROFILE** - getProfile 메소드를 나타냅니다.
- **COMMAND_GETVERSION** - getVersion 메소드를 나타냅니다.
- **COMMAND_POSTEVENT** - postEvent 메소드를 나타냅니다.
- **COMMAND_SETAUDIENCE** - setAudience 메소드를 나타냅니다.
- **COMMAND_SETDEBUG** - setDebug 메소드를 나타냅니다.
- **COMMAND_STARTSESSION** - startSession 메소드를 나타냅니다.

리턴값

없음.

예제

다음 예제는 getVersion 및 endSession을 호출하는 executeBatch 메소드에서 발췌한 것입니다.

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

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

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

setNumberRequested

setNumberRequested(*numberRequested*)

setNumberRequested 메소드는 getOffers 명령이 요청한 오퍼 수를 정의합니다.

- **numberRequested** - getOffers 명령이 요청한 오퍼 수를 정의하는 정수입니다.

리턴값

없음.

예제

다음 예제는 getOffers를 호출하는 executeBatch 메소드에서 발췌한 것입니다.

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

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

setRelyOnExistingSession

setRelyOnExistingSession(*relyOnExistingSession*)

setRelyOnExistingSession 메소드는 startSession 명령이 기존 세션을 사용하는지 여부를 정의하는 부울을 정의합니다.

true이면, executeBatch에 대한 세션 ID가 기존 세션 ID와 일치해야 합니다. false이면, executeBatch 메소드와 함께 새 세션 ID를 제공해야 합니다.

- **relyOnExistingSession** - 부울(true 또는 false).

리턴값

없음.

예제

다음 예제는 startSession을 호출하는 executeBatch 메소드에서 발췌한 것입니다.

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

NameValuePair 인터페이스 정보

Interact API의 많은 메소드는 NameValuePair 개체를 리턴하거나 사용자가 NameValuePair 개체를 인수로 전달해야 합니다. 메소드에 인수를 전달할 때에는 기본 구현 NameValuePairImpl을 사용해야 합니다.

getName

getName()

getName 메소드는 NameValuePair 개체의 이름 구성 요소를 리턴합니다.

리턴값

getName 메소드는 문자열을 리턴합니다.

예제

다음 예제는 getProfile에 대한 Response 개체를 처리하는 메소드에서 발췌한 것입니다.

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

getValueAsDate

```
getValueAsDate()
```

getValueAsDate 메소드는 NameValuePair 개체 값을 리턴합니다.

getValueAsDate를 사용하기 전에 getValueDataType을 사용하여 올바른 데이터 유형을 참조하고 있는지 확인해야 합니다.

리턴값

getValueAsDate 메소드는 날짜를 리턴합니다.

예제

다음 예제는 NameValuePair를 처리하는 메소드에서 발췌한 것이며 값이 날짜이면 해당 값을 인쇄합니다.

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

getValueAsNumeric

```
getValueAsNumeric()
```

getValueAsNumeric 메소드는 NameValuePair 개체 값을 리턴합니다.

getValueAsNumeric을 사용하기 전에 getValueDataType을 사용하여 올바른 데이터 유형을 참조하고 있는지 확인해야 합니다.

리턴값

getValueAsNumeric 메소드는 double을 리턴합니다. 예를 들어, 프로파일 테이블에 원래 정수로 저장된 값을 검색하는 경우 getValueAsNumeric은 double을 리턴합니다.

예제

다음 예제는 NameValuePair를 처리하는 메소드에서 발췌한 것이며 값이 숫자이면 해당 값을 인쇄합니다.

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

getValueAsString

```
getValueAsString()
```

getValueAsString 메소드는 NameValuePair 개체 값을 리턴합니다.

getValueAsString을 사용하기 전에 getValueDataType을 사용하여 올바른 데이터 유형을 참조하고 있는지 확인해야 합니다.

리턴값

getValueAsString 메소드는 문자열을 리턴합니다. 예를 들어, 프로파일 테이블에 원래 char, varchar 또는 char[10]으로 저장된 값을 검색하는 경우 getValueAsString은 문자열을 리턴합니다.

예제

다음 예제는 NameValuePair를 처리하는 메소드에서 발췌한 것이며 값이 문자열이면 해당 값을 인쇄합니다.

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

getValueDataType

```
getValueDataType()
```

getValueDataType 메소드는 NameValuePair 개체의 데이터 유형을 리턴합니다.

getValueAsDate, getValueAsNumeric 또는 getValueAsString을 사용하기 전에 getValueDataType을 사용하여 올바른 데이터 유형을 참조하고 있는지 확인해야 합니다.

리턴값

getValueDataType 메소드는 NameValuePair에 데이터, 숫자 또는 문자열이 포함되어 있는지 여부를 표시하는 문자열을 리턴합니다.

유효한 값은 다음과 같습니다.

- **DATA_TYPE_DATETIME** - 날짜 및 시간 값을 포함하는 날짜입니다.
- **DATA_TYPE_NUMERIC** - 숫자 값을 포함하는 double입니다.
- **DATA_TYPE_STRING** - 텍스트 값을 포함하는 문자열입니다.

예제

다음 예제는 getProfile 메소드에서 Response 개체를 처리하는 메소드에서 발췌한 것입니다.

```
for(NameValuePair nvp : response.getProfileRecord())
{
    System.out.println("Name:"+nvp.getName());
    if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
    {
        System.out.println("Value:"+nvp.getValueAsDate());
    }
    else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
    {
        System.out.println("Value:"+nvp.getValueAsNumeric());
    }
    else
    {
        System.out.println("Value:"+nvp.getValueAsString());
    }
}
```

setName

```
setName(name)
```

setName 메소드는 NameValuePair 개체의 이름 구성 요소를 정의합니다.

- **name** - NameValuePair 개체의 이름 구성 요소를 포함하는 문자열입니다.

리턴값

없음.

예제

다음 예제는 NameValuePair의 이름 구성 요소 정의 방법을 표시합니다.

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

setValueAsDate

```
setValueAsDate(valueAsDate)
```

setValueAsDate 메소드는 NameValuePair 개체 값을 정의합니다.

- **valueAsDate** - NameValuePair 개체의 날짜 및 시간 값을 포함하는 날짜입니다.

리턴값

없음.

예제

다음 예제는 값이 날짜인 경우 NameValuePair의 값 구성 요소 정의 방법을 표시합니다.

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

setValueAsNumeric

`setValueAsNumeric(`*`valueAsNumeric`*`)`

`setValueAsNumeric` 메소드는 NameValuePair 개체 값을 정의합니다.

- **valueAsNumeric** - NameValuePair 개체의 숫자 값을 포함하는 double입니다.

리턴값

없음.

예제

다음 예제는 값이 숫자인 경우 NameValuePair의 값 구성 요소 정의 방법을 표시합니다.

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

setValueAsString

`setValueAsString(`*`valueAsString`*`)`

`setValueAsString` 메소드는 NameValuePair 개체 값을 정의합니다.

- **valueAsString** - NameValuePair 개체 값을 포함하는 문자열입니다.

리턴값

없음.

예제

다음 예제는 값이 숫자인 경우 NameValuePair의 값 구성 요소 정의 방법을 표시합니다.

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

setValueDataType

`getValueDataType(valueDataType)`

`setValueDataType` 메소드는 NameValuePair 개체의 데이터 유형을 정의합니다.

유효한 값은 다음과 같습니다.

- **DATA_TYPE_DATETIME** - 날짜 및 시간 값을 포함하는 날짜입니다.
- **DATA_TYPE_NUMERIC** - 숫자 값을 포함하는 double입니다.
- **DATA_TYPE_STRING** - 텍스트 값을 포함하는 문자열입니다.

리턴값

없음.

예제

다음 예제는 NameValuePair 값의 데이터 유형 설정 방법을 표시합니다.

```
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);
```

Offer 클래스 정보

`Offer` 클래스는 Offer 개체를 정의하는 메소드를 포함합니다. 이 오퍼 개체는 Campaign의 오퍼의 많은 동일한 등록 정보를 포함합니다. 오퍼 개체는 다음 속성을 포함합니다.

- **AdditionalAttributes** - Campaign에 정의한 사용자 정의 오퍼 속성을 포함하는 NameValuePairs입니다.
- **Description** - 오퍼에 대한 설명입니다.
- **EffectiveDate** - 오퍼의 유효 날짜입니다.
- **ExpirationDate** - 오퍼의 만료 날짜입니다.
- **OfferCode** - 오퍼의 오퍼 코드입니다.
- **OfferName** - 오퍼의 이름입니다.
- **TreatmentCode** - 오퍼의 처리 코드입니다.

- **Score** - enableScoreOverrideLookup 등록 정보가 true인 경우 ScoreOverrideTable에 정의된 점수 또는 오퍼의 마케팅 점수입니다.

getAdditionalAttributes

```
getAdditionalAttributes()
```

getAdditionalAttributes 메소드는 Campaign에 정의된 사용자 정의 오퍼 속성을 리턴합니다.

리턴값

getAdditionalAttributes 메소드는 NameValuePair 개체의 배열을 리턴합니다.

예제

다음 예제는 모든 추가 속성을 자세히 살펴 유효 날짜 및 만료 날짜를 확인하고 기타 속성을 인쇄합니다.

```
for(NameValuePair offerAttribute : offer.getAdditionalAttributes())
{
    // check to see if the effective date exists
    if(offerAttribute.getName().equalsIgnoreCase("effectiveDate"))
    {
        System.out.println("Found effective date");
    }
    // check to see if the expiration date exists
    else if(offerAttribute.getName().equalsIgnoreCase("expirationDate"))
    {
        System.out.println("Found expiration date");
    }
    printNameValuePair(offerAttribute);
    }
}
public static void printNameValuePair(NameValuePair nvp)
{
    // print out the name:
    System.out.println("Name:"+nvp.getName());

    // based on the datatype, call the appropriate method to get the value
    if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_DATETIME)
        System.out.println("DateValue:"+nvp.getValueAsDate());
    else if(nvp.getValueDataType()==NameValuePair.DATA_TYPE_NUMERIC)
        System.out.println("NumericValue:"+nvp.getValueAsNumeric());
    else
        System.out.println("StringValue:"+nvp.getValueAsString());
}
```

getDescription

```
getDescription()
```

getDescription 메소드는 Campaign에 정의된 오퍼 설명을 리턴합니다.

리턴값

getDescription 메소드는 문자열을 리턴합니다.

예제

다음 예제는 오퍼 설명을 인쇄합니다.

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

getOfferCode

```
getOfferCode()
```

getOfferCode 메소드는 Campaign에 정의된 오퍼의 오퍼 코드를 리턴합니다.

리턴값

getOfferCode 메소드는 오퍼의 오퍼 코드를 포함하는 문자열 배열을 리턴합니다.

예제

다음 예제는 오퍼의 오퍼 코드를 인쇄합니다.

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

getOfferName

```
getOfferName()
```

getOfferName 메소드는 Campaign에 정의된 오퍼 이름을 리턴합니다.

리턴값

getOfferName 메소드는 문자열을 리턴합니다.

예제

다음 예제는 오퍼 이름을 인쇄합니다.

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

getScore

```
getScore()
```

getScore 메소드는 다음 중 하나를 리턴합니다.

- 기본 오퍼 테이블, 점수 재정의 테이블 또는 기본 제공 학습을 활성화하지 않은 경우, 이 메소드는 상호작용 전략 탭에 정의된 대로 오퍼의 마케팅 점수를 리턴합니다.
- 기본 오퍼 또는 점수 재정의 테이블을 활성화하고 기본 제공 학습은 활성화하지 않은 경우, 이 메소드는 기본 오퍼 테이블, 마케팅 담당자의 점수, 점수 재정의 테이블 간의 우선 순위 순서가 정의한 대로 오퍼 점수를 리턴합니다.
- 기본 제공 학습을 활성화한 경우, 이 메소드는 기본 제공 학습이 오퍼를 정렬하는 데 사용한 최종 점수를 리턴합니다.

리턴값

getScore 메소드는 오퍼 점수를 나타내는 정수를 리턴합니다.

예제

다음 예제는 오퍼 점수를 인쇄합니다.

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

getTreatmentCode

```
getTreatmentCode()
```

getTreatmentCode 메소드는 Campaign에 정의된 오퍼의 처리 코드를 리턴합니다.

Campaign은 처리 코드를 사용하여 제공된 오퍼의 인스턴스를 식별하기 때문에 postEvent 메소드를 사용하여 오퍼 컨택, 수락 또는 거부 이벤트를 로그할 때 이 코드를 이벤트 매개변수로 리턴해야 합니다. 오퍼 수락 또는 거부를 로깅하는 경우, 처리 코드를 나타내는 NameValuePair의 이름 값을 UACIOfferTrackingCode로 설정해야 합니다.

리턴값

getTreatmentCode 메소드는 문자열을 리턴합니다.

예제

다음 예제는 오퍼의 처리 코드를 인쇄합니다.

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

OfferList 클래스 정보

OfferList 클래스는 getOffers 메소드의 결과를 정의하는 메소드를 포함합니다. OfferList 개체는 다음 속성을 포함합니다.

- **DefaultString** - 대화식 채널의 상호작용 지점에 대해 정의된 기본 문자열입니다.
- **RecommendedOffers** - getOffers 메소드로 요청된 Offer 개체의 배열입니다.

OfferList 클래스는 오퍼 목록에 대해 작업합니다. 이 클래스는 Campaign 오퍼 목록과는 관련이 없습니다.

getDefaultString

```
getDefaultString()
```

getDefaultString 메소드는 Campaign에 정의된 상호작용 지점의 기본 문자열을 리턴합니다.

RecommendedOffers 개체가 비어 있으면, 일부 컨텐츠가 제공되도록 이 문자열을 제공할 접점을 구성해야 합니다. Interact는 RecommendedOffers 개체가 비어 있는 경우에만 DefaultString 개체를 채웁니다.

리턴값

getDefaultString 메소드는 문자열을 리턴합니다.

예제

다음 예제는 offerList 개체가 오퍼를 포함하지 않은 경우 기본 문자열을 가져옵니다.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
  System.out.println("Offer Name:"+offer.getOfferName());
    }
}
else // count on the default Offer String
    System.out.println("Default offer:"+offerList.getDefaultString());
```

getRecommendedOffers

```
getRecommendedOffers()
```

getRecommendedOffers 메소드는 getOffers 메소드가 요청한 Offer 개체의 배열을 리턴합니다.

getRecommendedOffer에 대한 응답이 비어 있으면, 접점이 getDefaultString의 결과를 제공해야 합니다.

리턴값

getRecommendedOffers 메소드는 Offer 개체를 리턴합니다.

예제

다음 예제는 OfferList 개체를 처리하고 권장되는 모든 오퍼에 대한 오퍼 이름을 인쇄합니다.

```
OfferList offerList=response.getOfferList();
if(offerList.getRecommendedOffers() != null)
{
    for(Offer offer : offerList.getRecommendedOffers())
    {
        // print offer
        System.out.println("Offer Name:"+offer.getOfferName());
    }
}
else // count on the default Offer String
System.out.println("Default offer:"+offerList.getDefaultString());
```

Response 클래스 정보

Response 클래스는 InteractAPI 클래스 메소드의 결과를 정의하는 메소드를 포함합니다. Response 개체는 다음 속성을 포함합니다.

- **AdvisoryMessages** - 권고 메시지의 배열입니다. 이 속성은 메소드가 실행될 때 경고나 오류가 발생한 경우에만 채워집니다.
- **ApiVersion** - API 버전을 포함한 문자열입니다. 이 속성은 getVersion 메소드로 채워집니다.
- **OfferList** - getOffers 메소드로 요청된 오퍼를 포함한 OfferList 개체입니다.
- **ProfileRecord** - 프로파일 데이터를 포함한 NameValuePairs 배열입니다. 이 속성은 getProfile 메소드로 채워집니다.
- **SessionID** - 세션 ID를 정의하는 문자열입니다. 모든 InteractAPI 클래스 메소드가 이 속성을 리턴합니다.
- **StatusCode** - 메소드가 오류 없이 또는 경고나 오류가 발생한 채로 실행되었는지 여부를 나타내는 숫자입니다. 모든 InteractAPI 클래스 메소드가 이 속성을 리턴합니다.

getAdvisoryMessages

```
getAdvisoryMessages()
```

getAdvisoryMessages 메소드는 Response 개체의 Advisory Message 배열을 리턴합니다.

리턴값

getAdvisoryMessages 메소드는 Advisory Message 개체의 배열을 리턴합니다.

예제

다음 예제는 Response 개체에서 AdvisoryMessage 개체를 가져오고 이를 반복하며 메시지를 인쇄합니다.

```
AdvisoryMessage[] messages = response.getAdvisoryMessages();
    for(AdvisoryMessage msg : messages)
    {
        System.out.println(msg.getMessage());
        // Some advisory messages may have additional detail:
        System.out.println(msg.getDetailMessage());
    }
```

getApiVersion

```
getApiVersion()
```

getApiVersion 메소드는 Response 개체의 API 버전을 리턴합니다.

getVersion 메소드는 Response 개체의 ApiVersion 속성을 채웁니다.

리턴값

Response 개체는 문자열을 리턴합니다.

예제

다음 예제는 getVersion에 대한 Response 개체를 처리하는 메소드에서 발췌한 것입니다.

```
if(response.getStatusCode() == Response.STATUS_SUCCESS)
{
    System.out.println("getVersion call processed with no warnings or errors");
    System.out.println("API Version:" + response.getApiVersion());
}
```

getOfferList

```
getOfferList()
```

getOfferList 메소드는 Response 개체의 OfferList 개체를 리턴합니다.

getOffers 메소드는 Response 개체의 OfferList 개체를 채웁니다.

리턴값

Response 개체는 OfferList 개체를 리턴합니다.

예제

다음 예제는 getOffers에 대한 Response 개체를 처리하는 메소드에서 발췌한 것입니다.

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

getAllOfferLists

```
getAllOfferLists()
```

getAllOfferLists 메소드는 Response 개체의 모든 OfferList 배열을 리턴합니다.

getOffersForMultipleInteractionPoints 메소드는 이를 사용하여 Response 개체의 OfferList 배열 개체를 채웁니다.

리턴값

Response 개체는 OfferList 배열을 리턴합니다.

예제

다음 예제는 getOffers에 대한 Response 개체를 처리하는 메소드에서 발췌한 것입니다.

```
OfferList[] allOfferLists = response.getAllOfferLists();
 if (allOfferLists != null) {
  for (OfferList ol : allOfferLists) {
   System.out.println("The following offers are delivered for interaction point "
     + ol.getInteractionPointName() + ":");
   for (Offer o : ol.getRecommendedOffers()) {
    System.out.println(o.getOfferName());
   }
  }
 }
```

getProfileRecord

```
getProfileRecord()
```

getProfileRecord 메소드는 현재 세션에 대한 프로파일 레코드를 NameValuePair 개체의 배열로 리턴합니다. 이 프로파일 레코드는 런타임 세션에서 앞서 추가된 eventParameters도 포함합니다.

getProfile 메소드는 Response 개체의 프로파일 레코드 NameValuePair 개체를 채웁니다.

리턴값

Response 개체는 NameValuePair 개체의 배열을 리턴합니다.

예제

다음 예제는 getOffers에 대한 Response 개체를 처리하는 메소드에서 발췌한 것입니다.

```
for(NameValuePair nvp : response.getProfileRecord())
{
    System.out.println("Name:"+nvp.getName());
    if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_DATETIME))
    {
        System.out.println("Value:"+nvp.getValueAsDate());
    }
    else if(nvp.getValueDataType().equals(NameValuePair.DATA_TYPE_NUMERIC))
    {
         System.out.println("Value:"+nvp.getValueAsNumeric());
    }
    else
    {
        System.out.println("Value:"+nvp.getValueAsString());
    }
}
```

getSessionID

```
getSessionID()
```

getSessionID 메소드는 세션 ID를 리턴합니다.

리턴값

getSessionID 메소드는 문자열을 리턴합니다.

예제

다음 예제는 오류가 관계된 세션을 표시하기 위해 오류 처리 시작 또는 끝에 표시할 수 있는 메시지를 표시합니다.

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

getStatusCode

```
getStatusCode()
```

getStatusCode 메소드는 Response 개체의 상태 코드를 리턴합니다.

리턴값

Response 개체는 정수를 리턴합니다.

- 0 - STATUS_SUCCESS - 호출된 메소드가 오류 없이 완료되었습니다. Advisory Message가 있을 수도 있고 없을 수도 있습니다.
- 1 - STATUS_WARNING - 호출된 메소드가 완료되고 한 개 이상의 경고 메지가 표시됩니다(오류 없음). 자세한 내용은 Advisory Message를 쿼리하십시오.
- 2 - STATUS_ERROR - 호출된 메소드가 완료되지 않았으며 한 개 이상의 오류 메시지가 있습니다. 자세한 내용은 Advisory Message를 쿼리하십시오.

예제

다음은 오류 처리 시 getStatusCode 사용 방법 예제입니다.

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

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

제 8 장 ExternalCallout API 정보

Interact는 대화식 플로차트에 사용할 수 있는 확장 가능한 매크로, EXTERNALCALLOUT을 제공합니다. 이 매크로로 플로차트 실행 중 외부 시스템과 통신할 사용자 정의 논리를 수행할 수 있습니다. 예를 들어, 플로차트 실행 중 고객의 신용 점수를 계산하려는 경우 Java 클래스(콜아웃)를 생성하여 점수를 계산한 후 대화식 플로차트의 선택 프로세스에서 EXTERNALCALLOUT 매크로를 사용해서 콜아웃으로부터 신용 점수를 가져올 수 있습니다.

EXTERNALCALLOUT 구성은 두 가지 주요 단계가 있습니다. 먼저, ExternalCallout API를 구현하는 Java 클래스를 생성해야 합니다. 두 번째는, Interact | 플로차트 | ExternalCallouts 범주에서 런타임 서버에 필요한 Marketing Platform 구성 등록 정보를 구성해야 합니다.

이 섹션의 정보 외에, Interact 런타임 서버의 Interact/docs/externalCalloutJavaDoc 디렉토리에서 ExternalCallout API에 대한 JavaDoc이 사용 가능합니다.

IAffiniumExternalCallout 인터페이스

ExternalCallout API는 IAffiniumExternalCallout 인터페이스에 포함되어 있습니다. EXTERNALCALLOUT 매크로를 사용하려면 IAffiniumExternalCallout 인터페이스를 구현해야 합니다.

IAffiniumExternalCallout를 구현하는 클래스에는 런타임 서버가 초기화할 수 있는 구성자가 있어야 합니다.

- 클래스에 구성자가 없으면 Java 컴파일러가 기본 구성자를 생성하며 이 구성자로 충분합니다.
- 인수가 있는 구성자가 있으면 런타임 서버에 사용될 인수가 없는 공용 구성자를 제공해야 합니다.

외부 콜아웃을 배포할 때 다음에 유의하십시오.

- 외부 콜아웃의 각 표현식 평가는 클래스의 새 인스턴스를 생성합니다. 클래스의 정적 멤버에 대한 스레드 안전 문제를 관리해야 합니다.
- 외부 콜아웃에 파일이나 데이터베이스 연결과 같은 시스템 자원이 사용되는 경우 연결을 관리해야 합니다. 런타임 서버에는 연결을 자동으로 정리하는 기능이 없습니다.

IBM Unica Interact 런타임 환경 설치의 lib 디렉토리에 있는 interact_externalcallout.jar에 대해 구현을 컴파일해야 합니다.

© Copyright IBM Corp. 2001, 2012

IAffiniumExternalCallout은 Java 클래스의 데이터를 요청하기 위해 런타임 서버를 사용합니다. 인터페이스는 다음 네 개의 메소드로 이루어집니다.

- getNumberOfArguments
- getValue
- initialize
- shutdown

EXTERNALCALLOUT에 사용할 웹 서비스 추가

EXTERNALCALLOUT 매크로는 해당 구성 등록 정보를 정의한 경우에만 콜아웃을 인식합니다.

런타임 환경에 대한 Marketing Platform에서 Interact > 플로차트 > externalCallouts 범주의 다음 구성 등록 정보를 추가 또는 정의하십시오.

구성 등록 정보	설정
externalCallouts 범주	외부 콜아웃에 대한 새 범주 생성
class	외부 콜아웃의 클래스 이름
classpath	외부 콜아웃 클래스 파일에 대한 클래스 경로
매개변수 데이터 범주	외부 콜아웃에 매개변수가 필요한 경우 새 매개변수 구성 등록 정보를 생성하고 각각에 값을 할당하십시오.

getNumberOfArguments

```
getNumberOfArguments()
```

getNumberOfArguments 메소드는 통합 중인 Java 클래스에서 예상하는 인수 수를 리턴합니다.

리턴값

getNumberOfArguments 메소드는 정수를 리턴합니다.

예제

다음 예제는 인수 수 인쇄를 표시합니다.

```
public int getNumberOfArguments()
{
    return 0;
}
```

getValue

```
getValue(audienceID, configData, arguments)
```

getValue 메소드는 콜아웃의 핵심 기능을 수행하고 결과를 리턴합니다.

getValue 메소드에 필요한 매개변수는 다음과 같습니다.

- **audienceID** - 대상 ID를 식별하는 값입니다.
- **configData** - 콜아웃에 필요한 구성 데이터의 키-값 쌍이 있는 맵입니다.
- **arguments** - 콜아웃에 필요한 인수입니다. 각 인수는 String, Double, Date 또는 이 중 하나의 List입니다. List 인수는 null 값을 포함할 수 있지만 List는 예를 들어 String 및 Double을 포함할 수 없습니다.

  구현 내에서 인수 유형 검사를 수행해야 합니다.

getValue 메소드가 어떤 이유로 실패하면 CalloutException을 리턴합니다.

리턴값

getValue 메소드는 문자열 목록을 리턴합니다.

예제

```
public List<String> getValue(AudienceId audienceId, Map<String,
  String> configurationData, Object... arguments) throws CalloutException
{
 Long customerId = (Long) audienceId.getComponentValue("Customer");
 // now query scoreQueryUtility for the credit score of customerId
 Double score = scoreQueryUtility.query(customerId);
 String str = Double.toString(score);
 List<String> list = new LinkedList<String>();
 list.add(str);
 return list;
}
```

initialize

```
initialize(configData)
```

initialize 메소드는 런타임 서버가 시작되면 한 번 호출됩니다. 런타임 중에 성능을 저해할 수 있는 작업(예: 데이터베이스 로드)이 있으면, 이 메소드를 통해 수행해야 합니다.

initialize 메소드에 필요한 매개변수는 다음과 같습니다.

- **configData** - 콜아웃에 필요한 구성 데이터의 키-값 쌍이 있는 맵입니다.

  Interact는 Interact > Flowchart > External Callouts > [External Callout] > Parameter Data 범주에 정의된 External Callout 매개변수에서 이 값을 읽습니다.

initialize 메소드가 어떤 이유로 실패하면 CalloutException을 리턴합니다.

리턴값

없음.

예제

```
public void initialize(Map<String, String> configurationData) throws CalloutException
{
    // configurationData has the key-value pairs specific to the environment
    // the server is running in
    // initialize scoreQueryUtility here
}
```

shutdown

shutdown(configData)

shutdown 메소드는 런타임 서버가 종료되면 한 번 호출됩니다. 콜아웃에 필요한 정리 작업이 있으면 이때에 실행해야 합니다.

shutdown 메소드에 필요한 매개변수는 다음과 같습니다.

- **configData** - 콜아웃에 필요한 구성 데이터의 키-값 쌍이 있는 맵입니다.

shutdown 메소드가 어떤 이유로 실패하면 CalloutException을 리턴합니다.

리턴값

없음.

예제

```
public void shutdown(Map<String, String> configurationData) throws CalloutException
{
    // shutdown scoreQueryUtility here
}
```

ExternalCallout API 예

1. 다음 컨텐츠로 GetCreditScore.java라는 파일을 생성하십시오. 파일은 모델링 응용 프로그램에서 점수를 페치하는 ScoreQueryUtility라는 클래스가 있다고 가정합니다.

```
import java.util.Map;
import com.unicacorp.interact.session.AudienceId;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.IAffiniumExternalCallout;
import com.unicacorp.interact.flowchart.macrolang.storedobjs.CalloutException;
import java.util.Random;

public class GetCreditScore implements IAffiniumExternalCallout
{
 // the class that has the logic to query an external system for a customer's credit score
 private static ScoreQueryUtility scoreQueryUtility;
 public void initialize(Map<String, String> configurationData) throws CalloutException
 {
  // configurationData has the key- value pairs specific to the environment the server is running in
  // initialize scoreQueryUtility here
 }

 public void shutdown(Map<String, String> configurationData) throws CalloutException
 {
 // shutdown scoreQueryUtility here
 }

 public int getNumberOfArguments()
```

```
 {
  // do not expect any additional arguments other than the customer's id
  return 0;
 }

 public List<String> getValue(AudienceId audienceId, Map<String, String> configurationData,
  Object... arguments) throws CalloutException
 {
  Long customerId = (Long) audienceId.getComponentValue("Customer");
  // now query scoreQueryUtility for the credit score of customerId
  Double score = scoreQueryUtility.query(customerId);
  String str = Double.toString(score);
  List<String> list = new LinkedList<String>();
  list.add(str);
  return list;
 }
}
```

2. GetCreditScore.java를 GetCreditScore.class로 컴파일하십시오.
3. GetCreditScore.class 및 사용하는 다른 클래스 파일을 포함한 creditscore.jar라는 jar 파일을 생성하십시오.
4. 런타임 서버의 일부 위치(예를 들어, /data/interact/creditscore.jar)로 jar 파일을 복사하십시오.
5. GetCreditScore 이름 및 /data/interact/creditscore.jar 클래스 경로의 외부 콜아웃을 구성 관리 페이지의 externalCallouts 범주에 생성하십시오.
6. 대화식 플로차트에서 콜아웃을 EXTERNALCALLOUT('GetCreditScore')로 사용할 수 있습니다.

IInteractProfileDataService 인터페이스

프로파일 데이터 서비스 API는 iInteractProfileDataService 인터페이스에 포함되어 있습니다. 이 인터페이스를 사용하면 Interact 세션이 시작할 때나 Interact 세션의 대상 ID가 변경될 때, (플랫 파일, 웹 서비스 등과 같은) 하나 이상의 외부 데이터 소스를 통해 Interact 세션으로 계층 구조 데이터를 가져올 수 있습니다.

프로파일 데이터 서비스 API를 사용하여 계층 구조 데이터 가져오기를 개발하려면 임의의 데이터 소스에서 정보를 끌어와서 ISessionDataRootNode 개체로 맵핑하는 Java 클래스를 작성한 후, EXTERNALCALLOUT 매크로를 사용하여 맵핑된 데이터를 참조해야 합니다.

IBM Unica Interact 런타임 환경 설치의 lib 디렉토리에 있는 interact_externalcallout.jar에 대해 구현을 컴파일해야 합니다.

이 인터페이스의 사용에 대한 전체 Javadoc 설명서는 웹 브라우저로 *Interact_home*/docs/externalCalloutJavaDoc에 있는 파일을 참조하십시오.

예제가 구현된 방법에 대해 주석으로 처리된 설명을 포함하여, 프로파일 데이터 서비스를 사용하는 방법에 대한 샘플 구현은 *Interact_home*/samples/externalcallout/XMLProfileDataService.java를 참조하십시오.

프로파일 데이터 서비스에서 사용할 데이터 소스 추가

EXTERNALCALLOUT 매크로는 해당 구성 등록 정보를 정의한 경우에만 프로파일 데이터 서비스 계층 구조 데이터 가져오기를 위한 데이터 소스를 인식합니다.

런타임 환경에 대한 Marketing Platform의 Interact > 프로파일 > 대상 수준 > [AudienceLevelName] > 프로파일 데이터 서비스 범주에서 다음 구성 등록 정보를 추가 또는 정의하십시오.

구성 등록 정보	설정
새 범주 이름 범주	정의하고 있는 데이터 소스의 이름입니다. 여기에 입력하는 이름은 같은 대상 수준에 대한 데이터 소스 중에서 고유해야 합니다.
활성화됨	정의되는 대상 수준에 대해 데이터 소스가 활성화되어 있는지 표시합니다.
className	IInteractProfileDataService를 구현하는 데이터 소스 클래스의 완전한 이름
classPath	프로파일 데이터 서비스 클래스 파일에 대한 클래스 경로입니다. 이것을 생략하면 포함하는 응용 프로그램 서버의 클래스 경로가 기본적으로 사용됩니다.
우선순위 범주	이 대상 수준 내부에서 이 데이터 소스의 우선순위입니다. 각 대상 수준에 대한 모든 데이터 소스 사이에서 고유한 값이어야 합니다. (즉, 어떤 데이터 소스에 대해 우선순위가 100으로 설정된 경우 해당 대상 수준 내에 있는 다른 데이터 소스는 그 어떤 것도 100의 우선순위를 가질 수 없습니다.)

제 9 장 IBM Unica Interact 유틸리티

이 섹션에서는 Interact에서 사용 가능한 관리 유틸리티를 설명합니다.

배포 유틸리티 실행(runDeployment.sh/.bat)

runDeployment 명령행 도구를 사용하는 경우, 가능한 모든 매개변수를 전체적으로 제시하고 runDeployment 도구 자체와 같은 위치에서 사용 가능한 deployment.properties 파일에서 제공하는 설정을 사용하여 명령행에서 특정 서버 그룹에 대한 대화식 채널을 배포할 수 있습니다. 명령행에서 대화식 채널 배포를 실행하는 기능은 OffersBySQL 기능을 사용하고 있을 때 특히 유용합니다. 예를 들어, Campaign 일괄처리 플로차트를 주기적으로 실행하도록 구성할 수도 있습니다. 플로차트 실행이 완료되면 이 명령행 도구를 사용하여 OffersBySQL 테이블에서 오퍼 배포를 초기화하는 트리거가 호출될 수 있습니다.

설명

Interact Design Time 서버의 다음 위치에 자동으로 설치된 runDeployment 명령행 도구를 찾을 수 있습니다.

Interact_home/interactDT/tools/deployment/runDeployment.sh(또는 Windows 서버의 runDeployment.bat)

이 명령으로 전달되는 유일한 인수는 대화식 채널/런타임 서버 그룹 조합을 배포하는 데 필요한 모든 가능한 매개변수를 설명하는 deployment.properties라는 파일의 위치입니다. 샘플 파일이 참조용으로 제공됩니다.

참고: runDeployment 유틸리티를 사용하기 전, 우선 서버에서 Java 런타임 환경의 위치를 제공하도록 텍스트 편집기로 편집해야 합니다. 예를 들어, *Interact_home*/jre 또는 *Platform_home*/jre 디렉토리 중 하나에 유틸리티에서 사용하도록 할 Java 런타임이 있는 경우, 두 디렉토리 중 하나를 경로로 지정할 수 있습니다. 또는 이 릴리스의 IBM Unica 제품에서 사용할 수 있도록 지원되는 Java 런타임 환경에 경로를 제공할 수도 있습니다.

보안(SSL) 환경에서 runDeployment 유틸리티 사용

Interact 서버에서 보안이 활성화되어 SSL 포트를 통해 연결할 때 runDeployment 유틸리티를 사용하려면 다음과 같이 신뢰 저장소 Java 등록 정보를 추가해야 합니다.

© Copyright IBM Corp. 2001, 2012

1. 대화식 채널 배포를 위해 deployment.properties 파일을 편집하고 있을 때는 이 예제에서처럼 보안 SSL URL을 사용하도록 deploymentURL 등록 정보를 수정하십시오.

   deploymentURL=https://<*HOST*>.<*DOMAIN*>:<*PORT*>/Campaign/interact/InvokeDeploymentServlet

2. ${JAVA_HOME}으로 시작하는 행에 다음 인수를 추가하도록 텍스트 편집기를 사용하여 runDeployment.sh 또는 runDeployment.bat 스크립트를 편집하십시오.

   -Djavax.net.ssl.trustStore=<*TrustStorePath*>

   예를 들어, 신뢰 저장소 인수를 추가한 후 이 행은 다음과 같이 보일 수 있습니다.

   ```
   ${JAVA_HOME}/bin/java -Djavax.net.ssl.trustStore=<TrustStorePath>
   -cp ${CLASSPATH}com.unicacorp.Campaign.interact.deployment.tools.
   InvokeDeploymentClient $1
   ```

   <*TrustStorePath*>를 실제 SSL 신뢰 저장소의 경로로 바꾸십시오.

유틸리티 실행

Java 런타임 환경을 제공하도록 유틸리티를 편집하고 환경에 일치하도록 deployment.properties 파일의 복사본을 사용자 정의한 후, 다음 명령으로 유틸리티를 실행할 수 있습니다.

Interact_home/interactDT/tools/deployment/runDeployment.sh *deployment.properties*

*Interact_home*을 Interact 디자인 시간 설치의 실제 값으로 바꾸고, *deployment.properties*를 이 배포를 위해 사용자 정의한 등록 정보 파일의 실제 경로와 이름으로 바꾸십시오.

샘플 *deployment.properties* 파일

샘플 deployment.properties 파일에는 자체 환경과 일치시키기 위해 사용자 정의해야 하는 모든 매개변수와 그에 대한 설명이 추가된 목록이 있습니다. 샘플 파일에는 각 매개변수가 무엇인지, 왜 특정 값을 사용자 정의해야 하는지에 대한 설명도 포함됩니다.

```
################################################################################
#
# The following properties feed into the InvokeDeploymentClient program.
# The program will look for a deploymentURL setting.  The program will post a
# request against that url; all other settings are posted as parameters in
# that request. The program then checks the status of the deployment and
# returns back when the deployment is at a terminal state (or if the
# specified waitTime has been reached).
#
# the output of the program will be of this format:
# <STATE> : <Misc Detail>
#
# where state can be one of the following:
# ERROR
```

```
# RUNNING
# SUCCESS
#
# Misc Detail is data that would normally populate the status message area
#  in the deployment gui of the IC summary page. NOTE: HTML tags may exist
# in the Misc Detail
#
################################################################################

################################################################################
# deploymentURL: url to the InvokeDeployment servlet that resides in Interact
# Design time.  should be in the following format:
# http://dt_host:port/Campaign/interact/InvokeDeploymentServlet
################################################################################
deploymentURL=http://localhost:7001/Campaign/interact/InvokeDeploymentServlet

################################################################################
# dtLogin:  this is the login that you would use to login to the Design Time if
# you had wanted to deploy the IC via the deployment gui inside the IC summary
# page.
################################################################################
dtLogin=asm_admin

################################################################################
# dtPW:  this is the PW that goes along with the dtLogin
################################################################################
dtPW=

################################################################################
# icName:  this is the name of the Interactive Channel that you want to deploy
################################################################################
icName=ic1

################################################################################
# partition:  this is the name of the partition
################################################################################
partition=partition1

################################################################################
# request:  this is the type of request that you want this tool to execute
# currently, there two behaviors.  If the value is "deploy", then the deployment
# will be executed.  All other values would cause the tool to simply return the
# status of the last deployment of the specified IC.
################################################################################
request=deploy

################################################################################
# serverGroup:  this is the name of the server group that you would like to
# deploy the IC.
################################################################################
serverGroup=defaultServerGroup

################################################################################
# serverGroupType:  this will indicate whether or not this deployment is going
# against production server group or a test server group. 1 denotes production
# 2 denotes test.
################################################################################
serverGroupType=1

################################################################################
# rtLogin:  this is the account used to authenticate against the server group
# that you are deploying to.
```

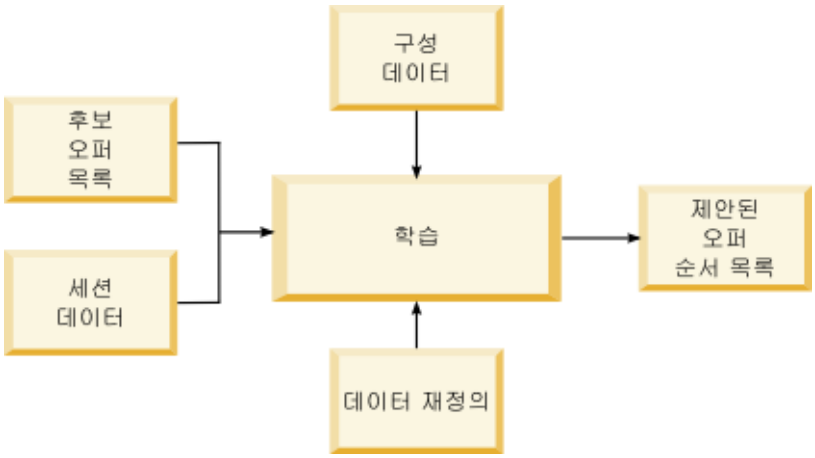
```
################################################################################
rtLogin=asm_admin

################################################################################
# rtPW:  this is the password associated to the rtLogin
################################################################################
rtPW=

################################################################################
# waitTime:  Once the tool submits the deployment request, the tool will check
# the status of the deployment.  If the deployment has not completed (or
# failed), then the tool will continue to poll the system for the status until
# a completed state has been reached, OR until the specified waitTime (in
# seconds) has been reached.
################################################################################
waitTime=5

################################################################################
# pollTime: If the status of a deployment is still in running state, then the
# tool will continue to check the status.  It will sleep in between status
# checks a number of seconds based on the pollTime setting .
################################################################################
pollTime=3

################################################################################
# global: Setting to false will make the tool NOT deploy the global settings.
# Non-availability of the property will still deploy the global settings.
################################################################################
global=true
```

제 10 장 학습 API 정보

Interact는 방문자 작업을 모니터하고 최적 오퍼를 제안하는(수락에 대해) 원시 베이지안 알고리즘을 사용하는 학습 모듈을 제공합니다. 학습 API를 사용하여 자신의 알고리즘으로 동일한 Java 인터페이스를 구현하여 학습 모듈을 직접 생성할 수 있습니다.

참고: 외부 학습을 사용하는 경우 학습에 관한 예 보고서(대화식 오퍼, 학습 세부 정보, 대화식 세그먼트 리프트 분석 보고서)는 올바른 데이터를 리턴하지 않습니다.

학습 API는 가장 단순한 수준으로 런타임 환경에서 데이터를 수집하고 권장되는 오퍼의 정렬된 목록을 리턴하는 메소드를 제공합니다.



Interact에서 다음 데이터를 수집할 수 있습니다.

- 오퍼 컨택 데이터
- 오퍼 수락 데이터
- 모든 세션 데이터
- Campaign 특정 오퍼 데이터
- 디자인 환경의 학습 범주 및 런타임 환경의 `offerserving`에 정의된 구성 등록 정보

알고리즘의 이 데이터를 사용하여 처리된 오퍼 목록을 생성할 수 있습니다. 그런 다음 가장 높은 권장에서 낮은 순으로 권장되는 오퍼 목록을 리턴합니다.

다이어그램에는 표시되지 않지만 학습 API를 사용하여 학습 구현에 대한 데이터를 수집할 수도 있습니다. 이 데이터를 메모리에 보관하거나 추가 분석을 위해 파일 또는 데이터베이스에 로그할 수 있습니다.

© Copyright IBM Corp. 2001, 2012

Java 클래스를 생성한 후 jar 파일로 변환할 수 있습니다. 또한 jar 파일을 생성한 후 구성 등록 정보를 편집하여 런타임 환경에 외부 학습 모듈이 인식되도록 구성해야 합니다. Java 클래스나 jar 파일을 외부 학습 모듈을 사용하는 모든 런타임 서버에 복사해야 합니다.

이 가이드의 정보 외에도, 런타임 서버의 Interact/docs/learningOptimizerJavaDoc 디렉토리에서 학습 최적화 프로그램 API에 대한 JavaDoc이 사용 가능합니다.

Interact 런타임 환경 설치의 lib 디렉토리에 있는 interact_learning.jar에 대해 구현을 컴파일해야 합니다.

사용자 정의 학습 구현을 쓸 때에는 다음 지침을 유념해야 합니다.

- 성능이 중요합니다.
- 멀티스레딩 및 스레드 안전 작업 방식이어야 합니다.
- 실패 모드 및 성능에 유의하여 모든 외부 자원을 관리해야 합니다.
- 예외, 로깅(log4j), 메모리를 적절하게 사용하십시오.

외부 학습 사용

학습 Java API를 사용하여 자체 학습 모듈을 작성할 수 있습니다. Marketing Platform에서 학습 유틸리티를 인식하도록 런타임 환경을 구성할 수 있습니다.

런타임 환경의 Marketing Platform에서 Interact > offerserving 범주의 다음 구성 등록 정보를 편집하십시오. 학습 최적화 프로그램 API에 대한 구성 등록 정보는 Interact > offerserving > External Learning Config 범주에 있습니다.

구성 등록 정보	설정
optimizationType	**ExternalLearning**
externalLearningClass	외부 학습에 대한 클래스 이름
externalLearningClassPath	런타임 서버의 외부 학습에 대한 클래스 또는 jar 파일 경로입니다. 서버 그룹을 사용 중이고 런타임 서버가 동일한 Marketing Platform 인스턴스를 참조하는 경우, 모든 서버에는 동일한 위치에 클래스 또는 jar 파일 복사본이 있어야 합니다.

이러한 변경 사항을 적용하려면 Interact 런타임 서버를 다시 시작해야 합니다.

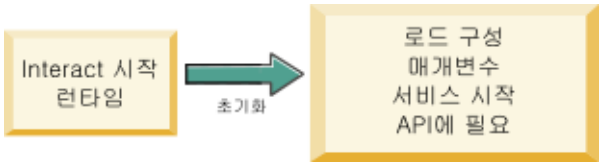
ILearning 인터페이스

학습 API는 ILearning 인터페이스를 중심으로 빌드됩니다. 학습 모듈의 사용자 정의 논리를 지원하려면 ILearning 인터페이스를 구현해야 합니다.

무엇보다 ILearning 인터페이스를 사용하여 런타임 환경에서 Java 클래스에 대한 데이터를 수집하고 권장되는 오퍼 목록을 런타임 서버로 다시 보낼 수 있습니다.

initialize

```
initialize(ILearningConfig config, boolean debug)
```



initialize 메소드는 런타임 서버가 시작되면 한 번 호출됩니다. 데이터베이스 테이블에서 정적 데이터를 로드하는 것과 같이 반복할 필요가 없지만 런타임 중 성능에 방해가 될 수 있는 작업이 있는 경우 이 메소드로 작업을 수행해야 합니다.

- **config** - ILearningConfig 개체는 학습에 관련된 모든 구성 등록 정보를 정의합니다.
- **debug** - 부울. true인 경우 런타임 환경 시스템의 로그 수준 상세도가 디버그로 설정되어 있음을 나타냅니다. 최상의 결과를 위해서는 로그에 쓰기 전에 이 값을 선택하십시오.

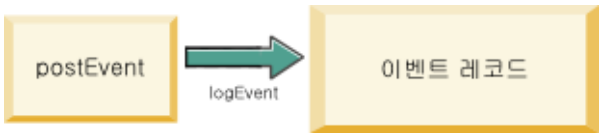
initialize 메소드가 어떠한 이유로 실패하는 경우 LearningException이 처리됩니다.

리턴값

없음

logEvent

```
logEvent(ILearningContext context,
              IOffer offer,
              IClientArgs clientArgs,
              IInteractSession session,
              boolean debug)
```



logEvent 메소드는 Interact API가 컨택이나 응답으로 로그하도록 구성된 이벤트를 게시할 때마다 런타임 서버에서 호출됩니다. 컨택 및 응답 데이터를 보고와 학습 용도로

데이터베이스나 파일에 로그하려면 이 메소드를 사용하십시오. 예를 들어, 기준에 따라 고객이 오퍼를 수락할 가능성을 알고리즘 방식으로 판별하려는 경우 이 메소드를 사용하여 데이터를 로그하십시오.

- **context** - 컨택, 수락 또는 거부와 같은 이벤트의 학습 컨텍스트를 정의하는 ILearningContext 개체입니다.
- **offer** - 이 이벤트가 로그되는 사항에 대한 오퍼를 정의하는 IOffer 개체입니다.
- **clientArgs** - 매개변수를 정의하는 IClientArgs 개체입니다. 현재 logEvent에는 clientArgs가 필요하지 않으므로 이 매개변수는 비어 있을 수 있습니다.
- **session** - 모든 세션 데이터를 정의하는 IInteractSession 개체입니다.
- **debug** - 부울. true인 경우 런타임 환경 시스템의 로그 수준 상세도가 디버그로 설정되어 있음을 나타냅니다. 최상의 결과를 위해서는 로그에 쓰기 전에 이 값을 선택하십시오.

logEvent 메소드가 실패하면 LearningException 예외가 처리됩니다.

리턴값

없음

optimizeRecommendList

```
optimizeRecommendList(list(ITreatment) recList,
  IClientArgs clientArg, IInteractSession session,
  boolean debug)
```

optimizeRecommendList 메소드는 권장되는 오퍼 목록과 세션 데이터를 사용하고 요청된 수의 오퍼를 포함한 목록을 리턴해야 합니다. optimizeRecommendList 메소드는 사용자의 학습 알고리즘을 통해 일정한 방식으로 오퍼를 정렬해야 합니다. 처음에 제공하려는 오퍼가 목록의 맨 처음에 오도록 오퍼 목록을 정렬해야 합니다. 예를 들어, 학습 알고리즘이 최상의 오퍼에 낮은 점수를 주는 경우 오퍼가 1, 2, 3으로 정렬되어야 하고 학습 알고리즘이 최상의 오퍼에 높은 점수를 주는 경우에는 오퍼가 100, 99, 98로 정렬되어야 합니다.

optimizeRecommendList 메소드에는 다음 매개변수가 필요합니다.

- **recList** - 런타임 환경에 권장되는 처리 개체(오퍼) 목록입니다.
- **clientArg** - 런타임 환경에 요청된 최소 수의 오퍼를 포함한 IClientArgs 개체입니다.

- **session** - 모든 세션 데이터를 포함한 IInteractSession 개체입니다.
- **debug** - 부울. true인 경우 런타임 환경 시스템의 로그 수준 상세도가 디버그로 설정되어 있음을 나타냅니다. 최상의 결과를 위해서는 로그에 쓰기 전에 이 값을 선택하십시오.

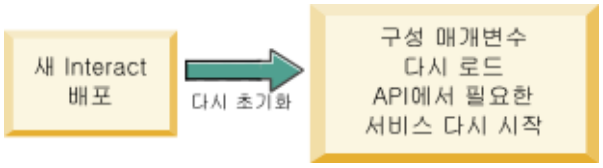
optimizeRecommendList 메소드가 실패하면 LearningException 예외가 처리됩니다.

리턴값

optimizeRecommendList 메소드는 ITreatment 개체의 목록을 리턴합니다.

reinitialize

```
reinitialize(ILearningConfig config,
                    boolean debug)
```



런타임 환경은 새 배포가 있을 때마다 reinitialize 메소드를 호출합니다. 이 메소드는 모든 학습 구성 데이터를 전달합니다. 학습 API에 필요한 구성 등록 정보를 읽는 서비스가 있는 경우 이 인터페이스가 해당 서비스를 다시 시작해야 합니다.

- **config** - 모든 구성 등록 정보를 포함한 ILearningConfig 개체입니다.
- **debug** - 부울. true인 경우 런타임 환경 시스템의 로그 수준 상세도가 디버그로 설정되어 있음을 나타냅니다. 최상의 결과를 위해서는 로그에 쓰기 전에 이 값을 선택하십시오.

logEvent 메소드가 실패하면 LearningException 예외가 처리됩니다.

리턴값

없음

shutdown

```
shutdown(ILearningConfig config, boolean debug)
```

런타임 환경은 런타임 서버가 시스템 종료될 때 shutdown 메소드를 호출합니다. 학습 모듈에 필요한 정리 작업이 있는 경우 이 때 작업을 실행해야 합니다.

shutdown 메소드에는 다음 매개변수가 필요합니다.

- **config** - 모든 구성 등록 정보를 정의하는 ILearningConfig 개체입니다.
- **debug** - 부울. true인 경우 런타임 환경 시스템의 로그 수준 상세도가 디버그로 설정되어 있음을 나타냅니다. 최상의 결과를 위해서는 로그에 쓰기 전에 이 값을 선택하십시오.

shutdown 메소드가 어떠한 이유로 실패하는 경우 LearningException이 처리됩니다.

리턴값

없음

IAudienceID 인터페이스

IAudienceID 인터페이스는 IInteractSession 인터페이스를 지원합니다. 대상 ID에 대한 인터페이스입니다. 대상 ID가 여러 파트로 이루어져 있을 수 있으므로 이 인터페이스를 사용하여 대상 수준 이름 및 대상 ID의 모든 요소에 액세스할 수 있습니다.

getAudienceLevel

```
getAudienceLevel()
```

getAudienceLevel 메소드는 대상 수준을 리턴합니다.

리턴값

getAudienceLevel 메소드는 대상 수준을 정의하는 문자열을 리턴합니다.

getComponentNames

```
getComponentNames()
```

getComponentNames 메소드는 대상 ID를 구성하는 구성 요소 이름 집합을 가져옵니다. 예를 들어, 대상 ID가 customerName 및 accountID 값으로 구성된 경우 getComponentNames는 customerName 및 accountID 문자열을 포함하는 집합을 리턴합니다.

리턴값

대상 ID의 구성 요소 이름을 포함하는 문자열 집합.

getComponentValue

getComponentValue(String *componentName*)

getComponentValue 메소드는 이름 지정된 구성 요소 값을 리턴합니다.

- **componentName** - 값을 검색할 구성 요소 이름을 정의하는 문자열입니다. 이 문자열은 대소문자를 구분하지 않습니다.

리턴값

getComponentValue 메소드는 구성 요소 값을 정의하는 개체를 리턴합니다.

IClientArgs

IClientArgs 인터페이스는 ILearning 인터페이스를 지원합니다. 이 인터페이스는 세션 데이터가 아직 적용되지 않은 접점에서 서버로 전달된 데이터에 적용할 추상입니다. 예를 들어, Interact API getOffers 메소드로 요청된 오퍼 수가 있습니다. 이 데이터는 맵에 저장됩니다.

getValue

getValue(int *clientArgKey*)

getValue 메소드는 요청된 맵 요소 값을 리턴합니다.

맵에서 필요한 요소는 다음과 같습니다.

- **1** - NUMBER_OF_OFFERS_REQUESTED. Interact API의 getOffers 메소드가 요청한 오퍼 수입니다. 이 상수는 정수를 리턴합니다.

리턴값

getValue 메소드는 요청된 맵 상수 값을 정의하는 개체를 리턴합니다.

IInteractSession

IInteractSession 인터페이스는 ILearning 인터페이스를 지원합니다. 런타임 환경의 현재 세션에 대한 인터페이스입니다.

getAudienceId

getAudienceId()

getAudienceId 메소드는 AudienceID 개체를 리턴합니다. 값을 추출하려면 IAudienceID 인터페이스를 사용하십시오.

리턴값

getAudienceId 메소드는 AudienceID 개체를 리턴합니다.

getSessionData

getSessionData()

getSessionData 메소드는 세션 변수 이름이 키인 수정 불가능한 세션 맵을 리턴합니다. 세션 변수 이름은 항상 대문자입니다. 값을 추출하려면 IInteractSessionData 인터페이스를 사용하십시오.

리턴값

getSessionData 메소드는 IInteractSessionData 개체를 리턴합니다.

IInteractSessionData 인터페이스

IInteractSessionData 인터페이스는 ILearning 인터페이스를 지원합니다. 현재 방문자의 런타임 세션 데이터에 대한 인터페이스입니다. 세션 데이터는 이름-값 쌍 목록으로 저장됩니다. 이 인터페이스를 사용하여 런타임 세션의 데이터 값을 변경할 수도 있습니다.

getDataType

getDataType(string *parameterName*)

getDataType 메소드는 지정된 매개변수 이름의 데이터 유형을 리턴합니다.

리턴값

getDataType 메소드는 InteractDataType 개체를 리턴합니다. IntearctDataType은 Unknown, String, Double, Date 또는 List로 표시된 Java 열거입니다.

getParameterNames

getParameterNames()

getParameterNames 메소드는 현재 세션의 모든 데이터 이름 집합을 리턴합니다.

리턴값

getParameterNames 메소드는 값이 설정된 이름 집합을 리턴합니다. 집합의 각 이름을 getValue(String)에 전달하여 값을 리턴할 수 있습니다.

getValue

getValue(*parameterName*)

getValue 메소드는 지정된 parameterName에 대응하는 개체 값을 리턴합니다. 개체는 String, Double 또는 Date입니다.

getValue 메소드에 필요한 매개변수는 다음과 같습니다.

- **parameterName** - 세션 데이터 이름-값 쌍의 이름을 정의하는 문자열입니다.

리턴값

getValue 메소드는 이름 지정된 매개변수 값을 포함하는 개체를 리턴합니다.

setValue

setValue(string *parameterName*, object *value*)

setValue 메소드를 사용하여 지정된 parameterName이 값을 설정할 수 있습니다. 값은 String, Double 또는 Date입니다.

setValue 메소드에 필요한 매개변수는 다음과 같습니다.

- **parameterName** - 세션 데이터 이름-값 쌍의 이름을 정의하는 문자열입니다.
- **value** - 지정된 매개변수 값을 정의하는 개체입니다.

리턴값

없음.

ILearningAttribute

ILearningAttribute 인터페이스는 ILearningConfig 인터페이스를 지원합니다. learningAttributes 범주의 구성 등록 정보에 정의된 학습 속성에 대한 인터페이스입니다.

getName

getName()

getName 메소드는 학습 속성 이름을 리턴합니다.

리턴값

getName 메소드는 학습 속성 이름을 정의하는 문자열을 리턴합니다.

ILearningConfig

ILearningConfig 인터페이스는 ILearning 인터페이스를 지원합니다. 이는 학습 구성 등록 정보에 대한 인터페이스입니다. 다음 메소드는 모두 등록 정보 값을 리턴합니다.

이 인터페이스는 다음 15개의 메소드로 구성됩니다.

- **getAdditionalParameters** - External Learning Config 범주에 정의된 추가 등록 정보 맵을 리턴합니다.
- **getAggregateStatsIntervalInMinutes** - 정수를 리턴합니다.
- **getConfidenceLevel** - 정수를 리턴합니다.
- **getDataSourceName** - 문자열을 리턴합니다.
- **getDataSourceType** - 문자열을 리턴합니다.
- **getInsertRawStatsIntervalInMinutes** - 정수를 리턴합니다.
- **getLearningAttributes** - ILearningAttribute 개체 목록을 리턴합니다.
- **getMaxAttributeNames** - 정수를 리턴합니다.
- **getMaxAttributeValues** - 정수를 리턴합니다.
- **getMinPresentCountThreshold** - 정수를 리턴합니다.
- **getOtherAttributeValue** - 문자열을 리턴합니다.
- **getPercentRandomSelection** - 정수를 리턴합니다.
- **getRecencyWeightingFactor** - 실수를 리턴합니다.
- **getRecencyWeightingPeriod** - 정수를 리턴합니다.
- **isPruningEnabled** - 부울을 리턴합니다.

ILearningContext

ILearningContext 인터페이스는 ILearning 인터페이스를 지원합니다.

getLearningContext

```
getLearningContext()
```

getLearningContext 메소드는 컨택, 수락 또는 거부 시나리오 여부를 알려주는 상수를 리턴합니다.

- **1** - LOG_AS_CONTACT
- **2** - LOG_AS_ACCEPT
- **3** - LOG_AS_REJECT

4, 5는 향후 사용을 위해 예약되어 있습니다.

리턴값

getLearningContext 메소드는 정수를 리턴합니다.

getResponseCode

getResponseCode()

getResponseCode 메소드는 이 오퍼에 할당된 응답 코드를 리턴합니다. 이 값은 Campaign 시스템 테이블의 UA_UsrResponseType 테이블에 있어야 합니다.

리턴값

getResponseCode 메소드는 응답 코드를 정의하는 문자열을 리턴합니다.

IOffer

IOffer 인터페이스는 ITreatment 인터페이스를 지원합니다. 이는 디자인 환경에 정의된 오퍼 개체에 대한 인터페이스입니다. 런타임 환경에서 오퍼 세부 정보를 수집하려면 IOffer 인터페이스를 사용하십시오.

getCreateDate

getCreateDate()

getCreateDate 메소드는 오퍼가 작성된 날짜를 리턴합니다.

리턴값

getCreateDate 메소드는 오퍼가 작성된 날짜를 정의하는 날짜를 리턴합니다.

getEffectiveDateFlag

getEffectiveDateFlag()

getEffectiveDateFlag 메소드는 오퍼 유효 날짜를 정의하는 숫자를 리턴합니다.

- **0** - 유효 날짜가 절대 날짜입니다(예: 2010년 3월 15일).
- **1** - 유효 날짜가 권장 날짜입니다.

리턴값

getEffectiveDateFlag 메소드는 오퍼 유효 날짜를 정의하는 정수를 리턴합니다.

getExpirationDateFlag

getExpirationDateFlag()

getExpirationDateFlag 메소드는 오퍼 만료 날짜를 설명하는 정수 값을 리턴합니다.

- **0** - 절대 날짜입니다(예: 2010년 3월 15일).
- **1** - 권장 날짜 이후의 일부 일 수 입니다(예: 14).
- **2** - 권장 날짜 이후의 월말입니다. 오퍼가 3월 31일에 제공된 경우, 오퍼는 해당 날짜에 만료됩니다.

리턴값

getExpirationDateFlag 메소드는 오퍼 만료 날짜를 설명하는 정수를 리턴합니다.

getOfferAttributes

```
getOfferAttributes()
```

getOfferAttributes 메소드는 오퍼에 대해 정의된 오퍼 속성을 IOfferAttributes 개체로 리턴합니다.

리턴값

getOfferAttributes 메소드는 IOfferAttributes 개체를 리턴합니다.

getOfferCode

```
getOfferCode()
```

getOfferCode 메소드는 Campaign에 정의된 오퍼의 오퍼 코드를 리턴합니다.

리턴값

getOfferCode 메소드는 IOfferCode 개체를 리턴합니다.

getOfferDescription

```
getOfferDescription()
```

getOfferDescription 메소드는 Campaign에 정의된 오퍼 설명을 리턴합니다.

리턴값

getOfferDescription 메소드는 문자열을 리턴합니다.

getOfferID

```
getOfferID()
```

getOfferID 메소드는 Campaign에 정의된 오퍼 ID를 리턴합니다.

리턴값

getOfferID 메소드는 오퍼 ID를 정의하는 long을 리턴합니다.

getOfferName

```
getOfferName()
```

getOfferName 메소드는 Campaign에 정의된 오퍼 이름을 리턴합니다.

리턴값

getOfferName 메소드는 문자열을 리턴합니다.

getUpdateDate

```
getUpdateDate()
```

getUpdateDate 메소드는 오퍼가 마지막으로 업데이트된 날짜를 리턴합니다.

리턴값

getUpdateDate 메소드는 오퍼가 마지막으로 업데이트된 날짜를 정의하는 날짜를 리턴합니다.

IOfferAttributes

IOfferAttributes 인터페이스는 IOffer 인터페이스를 지원합니다. 이는 디자인 환경에 정의된 오퍼에 대해 정의된 오퍼 속성에 대한 인터페이스입니다. 런타임 환경에서 오퍼 속성을 수집하려면 IOfferAttributes 인터페이스를 사용하십시오.

getParameterNames

```
getParameterNames()
```

getParameterNames 메소드는 오퍼 매개변수 이름 목록을 리턴합니다.

리턴값

getParameterNames 메소드는 오퍼 매개변수 이름 목록을 정의하는 집합을 리턴합니다.

getValue

```
getValue(String parameterName)
```

getValue 메소드는 지정된 오퍼 속성 값을 리턴합니다.

리턴값

getValue 메소드는 오퍼 속성 값을 정의하는 개체를 리턴합니다.

IOfferCode 인터페이스

IOfferCode 인터페이스는 ILearning 인터페이스를 지원합니다. 이는 디자인 환경에 정의된 오퍼에 대해 정의된 오퍼 코드에 대한 인터페이스입니다. 오퍼 코드는 1 - 다수의 문자열로 구성될 수 있습니다. 런타임 환경에서 오퍼 코드를 수집하려면 IOfferCode 인터페이스를 사용하십시오.

getPartCount

```
getPartCount()
```

getPartCount 메소드는 오퍼 코드를 구성하는 파트 수를 리턴합니다.

리턴값

getPartCount 메소드는 오퍼 코드의 파트 수를 정의하는 정수를 리턴합니다.

getParts

```
getParts()
```

getParts 메소드는 수정 불가능한 오퍼 코드 파트 목록을 가져옵니다.

리턴값

getParts 메소드는 수정 불가능한 오퍼 코드 파트 목록을 리턴합니다.

LearningException

LearningException 클래스는 ILearning 인터페이스를 지원합니다. 이 인터페이스 내 일부 메소드는 java.lang.Exception의 단순 하위 클래스인 LearningException 처리(throw)를 구현해야 합니다. 루트 예외가 있는 경우 루트 예외를 사용하여 LearningException을 생성할 것을 디버깅 목적으로 강력히 권장합니다.

IScoreOverride

IScoreOverride 인터페이스는 ITreatment 인터페이스를 지원합니다. 이 인터페이스를 사용하여 점수 재정의 또는 기본 오퍼 테이블에 정의된 데이터를 읽을 수 있습니다.

getOfferCode

```
getOfferCode()
```

getOfferCode 메소드는 이 대상 구성원에 대한 점수 재정의 테이블의 오퍼 코드 열 값을 리턴합니다.

리턴값

getOfferCode 메소드는 점수 재정의 테이블의 오퍼 코드 열 값을 정의하는 IOfferCode 개체를 리턴합니다.

getParameterNames

```
getParameterNames()
```

getParameterNames 메소드는 매개변수 목록을 리턴합니다.

리턴값

getParameterNames 메소드는 매개변수 목록을 정의하는 집합을 리턴합니다.

IScoreOverride 메소드에 포함된 매개변수는 다음과 같습니다. 별도로 지정하지 않으면 이 매개변수는 점수 재정의 테이블과 동일합니다.

- ADJ_EXPLORE_SCORE_COLUMN
- CELL_CODE_COLUMN
- ENABLE_STATE_ID_COLUMN
- ESTIMATED_PRESENT_COUNT - 예상 현재 수 재정의의 경우(오퍼 가중치 계산 중)
- FINAL_SCORE_COLUMN
- LIKELIHOOD_SCORE_COLUMN
- MARKETER_SCORE
- OVERRIDE_TYPE_ID_COLUMN
- PREDICATE_COLUMN - 오퍼 자격 판별을 위한 부울 표현식 작성의 경우
- PREDICATE_SCORE - 숫자 점수를 초래하는 표현식 작성의 경우
- SCORE_COLUMN
- ZONE_COLUMN

또한 열과 동일한 이름을 사용하여 점수 재정의 또는 기본 오퍼 테이블에 추가하는 열을 참조할 수 있습니다.

getValue

```
getValue(String parameterName)
```

getValue 메소드는 이 대상 구성원에 대한 점수 재정의 테이블의 영역 열 값을 리턴합니다.

- **parameterName** - 값을 검색할 매개변수 이름을 정의하는 문자열입니다.

리턴값

getValue 메소드는 요청된 매개변수 값을 정의하는 개체를 리턴합니다.

ISelectionMethod

ISelection 인터페이스는 권장 목록을 제시하는 데 사용되는 메소드를 표시합니다. Treatment 개체의 기본값은 EXTERNAL_LEARNING이므로 이 값을 설정하지 않아도 됩니다. 이 값은 궁극적으로 보고 목적으로 세부 컨택 기록에 저장됩니다.

나중에 분석을 위해 데이터를 저장하려면 이 인터페이스를 기존 상수를 벗어나 확장할 수 있습니다. 예를 들어, 두 개의 서로 다른 학습 모듈을 작성하고 별도의 서버 그룹에 이를 구현할 수 있습니다. SERVER_GROUP_1 및 SERVER_GROUP_2를 포함하도록 ISelection 인터페이스를 확장할 수 있습니다. 그런 다음 두 학습 모듈의 결과를 비교할 수 있습니다.

ITreatment 인터페이스

ITreatment 인터페이스는 처리 정보에 대한 인터페이스로 ILearning 인터페이스를 지원합니다. 처리는 디자인 환경에 정의된 대로 특정 셀에 할당된 오퍼를 나타냅니다. 이 인터페이스에서 할당된 마케팅 점수는 물론 셀 및 오퍼 정보도 얻을 수 있습니다.

getCellCode

getCellCode()

getCellCode 메소드는 Campaign에 정의된 셀 코드를 리턴합니다. 이 셀은 이 오퍼와 연관된 스마트 세그먼트에 할당된 셀입니다.

리턴값

getCellCode 메소드는 셀 코드를 정의하는 문자열을 리턴합니다.

getCellId

getOfferName()

getCellId 메소드는 Campaign에 정의된 셀의 내부 ID를 리턴합니다. 이 셀은 이 오퍼와 연관된 스마트 세그먼트에 할당된 셀입니다.

리턴값

getCellId 메소드는 셀 ID를 정의하는 long을 리턴합니다.

getCellName

getCellName()

getCellName 메소드는 Campaign에 정의된 셀 이름을 리턴합니다. 이 셀은 이 오퍼와 연관된 스마트 세그먼트에 할당된 셀입니다.

리턴값

getCellName 메소드는 셀 이름을 정의하는 문자열을 리턴합니다.

getLearningScore

```
getLearningScore()
```

getLearningScore 메소드는 이 처리에 대한 점수를 리턴합니다. 우선 순위는 다음과 같습니다.

1. 재정의 값(IScoreoveride.PREDICATE_SCORE_COLUMN별 키순 재정의 값 맵에 있는 경우)을 리턴
2. 값이 Null 아니면 예측 점수를 리턴
3. 마케팅 담당자의 점수(IScoreoveride.SCORE별 키순 재정의 값 맵에 있는 경우)를 리턴
4. 마케팅 담당자의 점수를 리턴

리턴값

getLearningScore 메소드는 학습 알고리즘이 판별한 점수를 정의하는 정수를 리턴합니다.

getMarketerScore

```
getMarketerScore()
```

getMarketerScore 메소드는 오퍼에 대한 상호작용 전략 탭에서 슬라이더로 정의된 마케팅 담당자의 점수를 리턴합니다.

상호작용 전략 탭 고급 옵션으로 정의한 마케팅 담당자의 점수를 검색하려면 getPredicateScore를 사용하십시오.

처리에서 실제로 사용되는 마케팅 담당자의 점수를 검색하려면 getLearningScore를 사용하십시오.

리턴값

getMarketerScore 메소드는 마케팅 담당자의 점수를 정의하는 정수를 리턴합니다.

getOffer

```
getOffer()
```

getOffer 메소드는 처리에 대한 오퍼를 리턴합니다.

리턴값

getOffer 메소드는 이 처리에 대한 오퍼를 정의하는 IOffer 개체를 리턴합니다.

getOverrideValues

```
getOverrideValues()
```

getOverrideValues 메소드는 기본 오퍼 또는 점수 재정의 테이블에 정의된 재정의를 리턴합니다.

리턴값

getOverrideValues 메소드는 IScoreOverride 개체를 리턴합니다.

getPredicate

```
getPredicate()
```

getPredicate 메소드는 기본 오퍼 테이블, 점수 재정의 테이블 또는 처리 규칙 고급 옵션의 예측 열이 정의한 예측을 리턴합니다.

리턴값

getPredicate 메소드는 기본 오퍼 테이블, 점수 재정의 테이블 또는 처리 규칙 고급 옵션의 예측 열이 정의한 예측을 정의하는 문자열을 리턴합니다.

getPredicateScore

```
getPredicateScore()
```

getPredicateScore 메소드는 기본 오퍼 테이블, 점수 재정의 테이블 또는 처리 규칙 고급 옵션의 예측 열에서 설정한 점수를 리턴합니다.

리턴값

getPredicateScore 메소드는 기본 오퍼 테이블, 점수 재정의 테이블 또는 처리 규칙 고급 옵션의 예측 열에서 설정한 점수를 정의하는 double을 리턴합니다.

getScore

```
getScore()
```

getScore 메소드는 다음 중 하나를 리턴합니다.

- enableScoreOverrideLookup 등록 정보가 false로 설정된 경우 Campaign의 상호작용 전략 탭에 정의된 오퍼의 마케팅 점수
- enableScoreOverrideLookup 등록 정보가 true로 설정된 경우 scoreOverrideTable이 정의한 오퍼 점수

리턴값

getScore 메소드는 오퍼 점수를 나타내는 정수를 리턴합니다.

getTreatmentCode

```
getTreatmentCode()
```

getTreatmentCode 메소드는 처리 코드를 리턴합니다.

리턴값

getTreatmentCode 메소드는 처리 코드를 정의하는 문자열을 리턴합니다.

setActualValueUsed

```
setActualValueUsed(string parmName, object value)
```

학습 알고리즘 실행의 여러 스테이지에서 사용되는 값을 정의하려면 setActualValueUsed 메소드를 사용하십시오.

예를 들어, 이 메소드를 사용하여 컨택 및 응답 기록 테이블에 기록하고 기존 샘플 보고서를 수정하는 경우 보고에 학습 알고리즘의 데이터를 포함시킬 수 있습니다.

- **parmName** - 설정 중인 매개변수 이름을 정의하는 문자열입니다.
- **value** - 설정 중인 매개변수 값을 정의하는 문자열입니다.

리턴값

없음.

학습 API 예

이 섹션은 LearningInterface의 샘플 구현을 포함합니다. 이 구현은 샘플일 뿐, 운용 환경에 사용하도록 디자인된 것이 아님에 유의하십시오.

이 예는 수락 및 컨택 개수를 추적하고 특정 오퍼의 수락 대 컨택 비율을 오퍼의 수락 가능성 비율로 사용합니다. 제시되지 않은 오퍼는 권장 사항에서 더 높은 우선순위를 차지합니다. 최소 하나의 컨택이 있는 오퍼가 내림차순 수락 가능성 비율에 따라 정렬됩니다.

이 예에서는 모든 개수가 메모리에 보관됩니다. 런타임 서버는 메모리가 부족하게 되므로 이는 실제 시나리오가 아닙니다. 실제 운용 시나리오에서는 데이터베이스로 개수를 지속시켜야 합니다.

```
package com.unicacorp.interact.samples.learning.v2;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.unicacorp.interact.samples.learning.SampleOptimizer.MyOfferSorter;
import com.unicacorp.interact.treatment.optimization.IClientArgs;
import com.unicacorp.interact.treatment.optimization.IInteractSession;
import com.unicacorp.interact.treatment.optimization.ILearningConfig;
import com.unicacorp.interact.treatment.optimization.ILearningContext;
import com.unicacorp.interact.treatment.optimization.IOffer;
import com.unicacorp.interact.treatment.optimization.LearningException;
import com.unicacorp.interact.treatment.optimization.v2.ILearning;
import com.unicacorp.interact.treatment.optimization.v2.ITreatment;

/**
 * This is a sample implementation of the learning optimizer.
 * The interface ILearning may be found in the interact.jar library.
 *
 * To actually use this implementation, select ExternalLearning as the optimizationType in the offerServing node
 * of the Interact application within the Platform configuration.  Within the offerserving node there is also
 * an External Learning config category - within there you must set the name of the class to this:
 * com.unicacorp.interact.samples.learning.v2.SampleLearning.  Please note however, this implementation is just a sample
 * and was not designed to be used in a production environment.
 *
 *
 * This example keeps track of accept and contact counts and uses the ratio of accept to contacts
 * for a particular offer as the acceptance probability rate for the offer.
 *
 *
 * Offers not presented will get higher priority for recommending.
 * Offers with at least one contact will be ordered based on descending acceptance probability rate.
 *
 * Note: all counts are kept in memory.  This is not a realistic scenario since you would run out of memory sooner or
 * later.  In a real production scenario, the counts should be persisted into a database.
 *
 */

public class SampleLearning implements ILearning
{

    // A map of offer ids to contact count for the offer id
    private Map<Long,Integer> _offerToContactCount = new HashMap<Long, Integer>();

    // A map of offer ids to contact count for the offer id
    private Map<Long,Integer> _offerToAcceptCount = new HashMap<Long, Integer>();


    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#initialize
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void initialize(ILearningConfig config, boolean debug) throws LearningException
    {
        // If any remote connections are required, this is a good place to initialize those connections as this
        // method is called once at the start of the interact runtime webapp.
        // This example does not have any remote connections and prints for debugging purposes that this method will
        // be called
        System.out.println("Calling initialize for SampleLearning");
    }

    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#reinitialize
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void reinitialize(ILearningConfig config, boolean debug) throws LearningException
```

```
{
    // If an IC is deployed, this reinitialize method is called to allow the implementation to
    // refresh any updated configuration settings
    System.out.println("Calling reinitialize for SampleLearning");
}


/* (non-Javadoc)
 * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#logEvent
 *   (com.unicacorp.interact.treatment.optimization.v2.ILearningContext,
 *   com.unicacorp.interact.treatment.optimization.v2.IOffer,
 *   com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
 *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
 */
public void logEvent(ILearningContext context, IOffer offer, IClientArgs clientArgs,
IInteractSession session, boolean debug) throws LearningException
{
    System.out.println("Calling logEvent for SampleLearning");


    if(context.getLearningContext()==ILearningContext.LOG_AS_CONTACT)
    {
        System.out.println("adding contact");

        // Keep track of all contacts in memory
        synchronized(_offerToAcceptCount)
        {
           Integer count = _offerToAcceptCount.get(offer.getOfferId());
           if(count == null)
               count = new Integer(1);
           else
               count++;
           _offerToAcceptCount.put(offer.getOfferId(), ++count);
        }

    }
    else if(context.getLearningContext()==ILearningContext.LOG_AS_ACCEPT)
    {
        System.out.println("adding accept");
        // Keep track of all accept counts in memory by adding to the map
        synchronized(_offerToAcceptCount)
        {
           Integer count = _offerToAcceptCount.get(offer.getOfferId());
           if(count == null)
               count = new Integer(1);
           else
               count++;
           _offerToAcceptCount.put(offer.getOfferId(), ++count);
        }
        }

}

/* (non-Javadoc)
 * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#optimizeRecommendList
 *   (java.util.List, com.unicacorp.interact.treatment.optimization.v2.IClientArgs,
 *   com.unicacorp.interact.treatment.optimization.IInteractSession, boolean)
 */
public List<ITreatment> optimizeRecommendList(List<ITreatment> recList,
    IClientArgs clientArgs, IInteractSession session, boolean debug)
    throws LearningException
{
    System.out.println("Calling optimizeRecommendList for SampleLearning");

    // Sort the candidate treatments by calling the sorter defined in this class and return the sorted list
    Collections.sort(recList,new MyOfferSorter());

    // now just return what was asked for via "numberRequested" variable
    List<ITreatment> result = new ArrayList<ITreatment>();

    for(int x=0;x<(Integer)clientArgs.getValue(IClientArgs.NUMBER_OF_OFFERS_REQUESTED) && x<recList.size();x++)
    {
        result.add(recList.get(x));
    }
    return result;
}
```

```
    /* (non-Javadoc)
     * @see com.unicacorp.interact.treatment.optimization.v2.ILearning#shutdown
     *   (com.unicacorp.interact.treatment.optimization.v2.ILearningConfig, boolean)
     */
    public void shutdown(ILearningConfig config, boolean debug) throws LearningException
    {
        // If any remote connections exist, this would be a good place to gracefully
        // disconnect from them as this method is called at the shutdown of the Interact runtime
        // webapp.   For this example, there is nothing really to do
        // except print out a statement for debugging.
        System.out.println("Calling shutdown for SampleLearning");

    }
    // Sort by:
    // 1. offers with zero contacts  - for ties, order is based on original input
    // 2. descending accept probability rate - for ties, order is based on original input

    public class MyOfferSorter implements Comparator<ITreatment>
    {
        private static final long serialVersionUID = 1L;

        /* (non-Javadoc)
         * @see java.lang.Comparable#compareTo(java.lang.Object)
         */
        public int compare(ITreatment treatment1, ITreatment treatment2)
        {

            // get contact count for both treatments
            Integer contactCount1 = _offerToContactCount.get(treatment1.getOffer().getOfferId());
            Integer contactCount2 = _offerToContactCount.get(treatment2.getOffer().getOfferId());

            // if treatment hasn't been contacted, then that wins
            if(contactCount1 == null || contactCount1 == 0)
                return -1;

            if(contactCount2 == null || contactCount2 == 0)
                return 1;

            // get accept counts
            Integer acceptCount1 = _offerToAcceptCount.get(treatment1.getOffer().getOfferId());
            Integer acceptCount2 = _offerToAcceptCount.get(treatment2.getOffer().getOfferId());

            float acceptProbability1 = (float) acceptCount1 / (float) contactCount1;
            float acceptProbability2 = (float) acceptCount2 / (float) contactCount2;

            // descending order
            return (int) (acceptProbability2 - acceptProbability1);

        }
    }

}
```

부록 A. IBM Unica Interact WSDL

```
<?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">
```

© Copyright IBM Corp. 2001, 2012

```
 <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>
```

부록 B. Interact 런타임 환경 구성 등록 정보

이 섹션에서는 Interact 런타임 환경에 대한 모든 구성 등록 정보를 설명합니다.

Interact | 일반

이 구성 등록 정보는 기본 로그 수준과 로케일 설정을 포함한, 런타임 환경의 일반 설정을 정의합니다.

log4jConfig

설명

log4j 등록 정보를 포함한 파일의 위치입니다. 이 경로는 INTERACT_HOME 환경 변수에 대한 상대 경로여야 합니다. INTERACT_HOME은 Interact 설치 디렉토리의 위치입니다.

기본값

./conf/interact_log4j.properties

asmUserForDefaultLocale

설명

asmUserForDefaultLocale 등록 정보는 Interact가 로케일 설정을 받아온 IBM Unica Marketing 사용자를 정의합니다.

로케일 설정은 Interact API의 언어 권고 메시지 및 디자인 시간의 언어 표시를 정의합니다. 로케일 설정이 운영 체제 설정과 일치하지 않는 경우 Interact는 여전히 기능하지만 디자인 시간 표시 및 권고 메시지가 다른 언어일 수 있습니다.

기본값

정의된 기본값이 없습니다.

Interact | 일반 | learningTablesDataSource

이 구성 등록 정보는 기본 제공 학습 테이블의 데이터 소스 설정을 정의합니다. Interact 기본 제공 학습을 사용 중인 경우 이 데이터 소스를 정의해야 합니다.

학습 API를 사용하여 직접 학습 구현을 생성하는 경우 ILearningConfig 인터페이스를 사용하여 이 값을 읽도록 사용자 정의 학습 구현을 구성할 수 있습니다.

© Copyright IBM Corp. 2001, 2012

jndiName

설명

이 jndiName 등록 정보를 사용하여 Interact 런타임 서버가 액세스하는 학습 테이블의 응용 프로그램 서버(Websphere 또는 WebLogic)에 정의된 JNDI(Java Naming and Directory Interface) 데이터 소스를 식별하십시오.

학습 테이블은 aci_lrntab ddl 파일로 생성되며 여러 테이블 중 UACI_AttributeValue 및 UACI_OfferStats 테이블을 포함합니다.

기본값

정의된 기본값이 없습니다.

type

설명

Interact 런타임 서버가 액세스하는 학습 테이블에 사용되는 데이터 소스의 데이터베이스 유형입니다.

학습 테이블은 aci_lrntab ddl 파일로 생성되며 여러 테이블 중 UACI_AttributeValue 및 UACI_OfferStats 테이블을 포함합니다.

기본값

SQLServer

올바른 값

SQLServer | DB2 | ORACLE

connectionRetryPeriod

설명

ConnectionRetryPeriod 등록 정보는 학습 테이블에 장애가 발생할 경우 Interact가 자동으로 데이터베이스 연결 요청을 재시도하는 시간(초)을 지정합니다. Interact는 데이터베이스 오류나 실패를 보고하기 전에 이 시간 동안 자동으로 데이터베이스에 다시 연결하려 시도합니다. 0 값을 설정하면 Interact가 무한정 재시도하며 값을 -1로 설정하는 경우에는 재시도하지 않습니다.

학습 테이블은 aci_lrntab ddl 파일로 생성되며 여러 테이블 중 UACI_AttributeValue 및 UACI_OfferStats 테이블을 포함합니다.

기본값

-1

connectionRetryDelay

설명

ConnectionRetryDelay 등록 정보는 학습 테이블에 장애가 발생한 후 Interact가 데이터베이스에 다시 연결하려 시도하기 전에 대기하는 시간(초)을 지정합니다. 값을 -1로 설정하면 재시도하지 않습니다.

학습 테이블은 aci_lrntab ddl 파일로 생성되며 여러 테이블 중 UACI_AttributeValue 및 UACI_OfferStats 테이블을 포함합니다.

기본값

-1

schema

설명

기본 제공 학습 모듈에 대한 테이블을 포함한 스키마의 이름입니다. Interact는 이 등록 정보의 값을 모든 테이블 이름 앞에 삽입합니다. 예를 들어, UACI_IntChannel이 schema.UACI_IntChannel이 됩니다.

스키마를 정의할 필요는 없습니다. 스키마를 정의하지 않으면 Interact는 테이블의 소유자가 스키마와 동일하다고 가정합니다. 모호하지 않도록 이 값을 설정해야 합니다.

기본값

정의된 기본값이 없습니다.

Interact | 일반 | prodUserDataSource

이 구성 등록 정보는 운용 학습 테이블의 데이터 소스 설정을 정의합니다. 이 데이터 소스를 정의해야 합니다. 이는 배포 후 대화식 플로차트를 실행할 때 런타임 환경에 참조되는 데이터 소스입니다.

jndiName

설명

이 jndiName 등록 정보를 사용하여 Interact 런타임 서버가 액세스하는 고객 테이블의 응용 프로그램 서버(Websphere 또는 WebLogic)에 정의된 JNDI(Java Naming and Directory Interface) 데이터 소스를 식별하십시오.

기본값

정의된 기본값이 없습니다.

type

설명

Interact 런타임 서버가 액세스하는 고객 테이블의 데이터베이스 유형입니다.

기본값

SQLServer

올바른 값

SQLServer | DB2 | ORACLE

aliasPrefix

설명

AliasPrefix 등록 정보는 Interact 런타임 서버가 액세스하는 고객 테이블에 새 테이블을 쓰고 차원 테이블을 사용할 때 Interact가 자동으로 생성하는 별명 이름을 Interact가 형성하는 방식을 지정합니다.

각 데이터베이스에는 최대 ID 길이가 있음에 유의하십시오. 사용 중인 데이터베이스의 문서를 검토하여 설정한 값이 데이터베이스의 최대 ID 길이를 초과하지 않는지 확인하십시오.

기본값

A

connectionRetryPeriod

설명

ConnectionRetryPeriod 등록 정보는 런타임 고객 테이블에 장애가 발생할 경우 Interact가 자동으로 데이터베이스 연결 요청을 재시도하는 시간(초)을 지정합니다. Interact는 데이터베이스 오류나 실패를 보고하기 전에 이 시간 동안 자동으로 데이터베이스에 다시 연결하려 시도합니다. 0 값을 설정하면 Interact가 무한정 재시도하며 값을 -1로 설정하는 경우에는 재시도하지 않습니다.

기본값

-1

connectionRetryDelay

설명

ConnectionRetryDelay 등록 정보는 Interact 런타임 고객 테이블에 장애가 발생한 후 Interact가 데이터베이스에 다시 연결하려 시도하기 전에 대기하는 시간(초)을 지정합니다. 값을 -1로 설정하면 재시도하지 않습니다.

기본값

-1

schema

설명

프로파일 데이터 테이블을 포함한 스키마의 이름입니다. Interact는 이 등록 정보의 값을 모든 테이블 이름 앞에 삽입합니다. 예를 들어, UACI_IntChannel이 schema.UACI_IntChannel이 됩니다.

스키마를 정의할 필요는 없습니다. 스키마를 정의하지 않으면 Interact는 테이블의 소유자가 스키마와 동일하다고 가정합니다. 모호하지 않도록 이 값을 설정해야 합니다.

기본값

정의된 기본값이 없습니다.

Interact | 일반 | systemTablesDataSource

이 구성 등록 정보는 런타임 환경에 대한 시스템 테이블의 데이터 소스 설정을 정의합니다. 이 데이터 소스를 정의해야 합니다.

jndiName

설명

jndiName 등록 정보를 사용하여 런타임 환경 테이블에 대한 응용 프로그램 서버(Websphere 또는 WebLogic)에 정의된 JNDI(Java Naming and Directory Interface) 데이터 소스를 식별하십시오.

런타임 환경 데이터베이스는 aci_runtime 및 aci_populate_runtime dll 스크립트로 채워진 데이터베이스이며 예를 들어, 여러 테이블 중 다음 UACI_CHOfferAttrib 및 UACI_DefaultedStat 테이블을 포함합니다.

기본값

정의된 기본값이 없습니다.

type

설명

런타임 환경 시스템 테이블의 데이터베이스 유형입니다.

런타임 환경 데이터베이스는 aci_runtime 및 aci_populate_runtime dll 스크립트로 채워진 데이터베이스이며 예를 들어, 여러 테이블 중 다음 UACI_CHOfferAttrib 및 UACI_DefaultedStat 테이블을 포함합니다.

기본값

SQLServer

올바른 값

SQLServer | DB2 | ORACLE

connectionRetryPeriod

설명

ConnectionRetryPeriod 등록 정보는 런타임 시스템 테이블에 장애가 발생할 경우 Interact가 자동으로 데이터베이스 연결 요청을 재시도하는 시간(초)을 지정합니다. Interact는 데이터베이스 오류나 실패를 보고하기 전에 이 시간 동안 자동으로 데이터베이스에 다시 연결하려 시도합니다. 0 값을 설정하면 Interact가 무한정 재시도하며 값을 -1로 설정하는 경우에는 재시도하지 않습니다.

런타임 환경 데이터베이스는 aci_runtime 및 aci_populate_runtime dll 스크립트로 채워진 데이터베이스이며 예를 들어, 여러 테이블 중 다음 UACI_CHOfferAttrib 및 UACI_DefaultedStat 테이블을 포함합니다.

기본값

-1

connectionRetryDelay

설명

ConnectionRetryDelay 등록 정보는 Interact 런타임 시스템 테이블에 장애가 발생한 후 Interact가 데이터베이스에 다시 연결하려 시도하기 전에 대기하는 시간(초)을 지정합니다. 값을 -1로 설정하면 재시도하지 않습니다.

런타임 환경 데이터베이스는 aci_runtime 및 aci_populate_runtime dll 스크립트로 채워진 데이터베이스이며 예를 들어, 여러 테이블 중 다음 UACI_CHOfferAttrib 및 UACI_DefaultedStat 테이블을 포함합니다.

기본값

-1

schema

설명

런타임 환경에 대한 테이블을 포함한 스키마의 이름입니다. Interact는 이 등록 정보의 값을 모든 테이블 이름 앞에 삽입합니다. 예를 들어, UACI_IntChannel이 schema.UACI_IntChannel이 됩니다.

스키마를 정의할 필요는 없습니다. 스키마를 정의하지 않으면 Interact는 테이블의 소유자가 스키마와 동일하다고 가정합니다. 모호하지 않도록 이 값을 설정해야 합니다.

기본값

정의된 기본값이 없습니다.

Interact | 일반 | systemTablesDataSource | loaderProperties

이 구성 등록 정보는 런타임 환경에 대한 시스템 테이블의 데이터베이스 로더 유틸리티 설정을 정의합니다. 데이터베이스 로더 유틸리티를 사용 중인 경우에만 이 등록 정보를 정의해야 합니다.

databaseName

설명

데이터베이스 로더가 연결하는 데이터베이스의 이름입니다.

기본값

정의된 기본값이 없습니다.

LoaderCommandForAppend

설명

LoaderCommandForAppend 매개변수는 Interact의 컨택 및 응답 기록 스테이징 데이터베이스 테이블에 레코드를 추가할 데이터베이스 로드 유틸리티를 호출하기 위해 실행되는 명령을 지정합니다. 컨택 및 응답 기록 데이터에 대한 데이터베이스 로더 유틸리티를 사용하려면 이 매개변수를 설정해야 합니다.

이 매개변수는 데이터베이스 로드 유틸리티 실행 파일이나 데이터베이스 로드 유틸리티를 실행하는 스크립트에 대한 전체 경로 이름으로 지정됩니다. 스크립트를 사용하면 로드 유틸리티를 호출하기 전에 추가 설정을 수행할 수 있습니다.

대부분의 데이터베이스 로드 유틸리티는 여러 인수를 실행해야 합니다. 여기에는 로드할 데이터 파일과 컨트롤 파일 및 로드할 데이터베이스와 테이블을 지정하는 작업이 포함될 수 있습니다. 토큰은 명령이 실행될 때 지정된 요소로 대체됩니다.

데이터베이스 로드 유틸리티를 호출할 때 사용할 올바른 구문은 데이터베이스 로드 유틸리티 문서를 참조하십시오.

이 매개변수는 기본적으로 정의되어 있지 않습니다.

LoaderCommandForAppend에 사용 가능한 토큰은 다음 테이블에 설명되어 있습니다.

토큰	설명
<CONTROLFILE>	이 토큰은 Interact가 LoaderControlFileTemplate 매개변수에 지정된 템플릿에 따라 생성하는 임시 컨트롤 파일에 대한 전체 경로 및 파일 이름으로 대체됩니다.

토큰	설명
<DATABASE>	이 토큰은 Interact가 데이터를 로드 중인 데이터 소스의 이름으로 대체됩니다. 이 이름은 이 데이터 소스의 범주 이름에 사용된 데이터 소스 이름과 동일합니다.
<DATAFILE>	이 토큰은 프로세스 로드 중 Interact가 생성한 임시 데이터 파일에 대한 전체 경로 및 파일 이름으로 대체됩니다. 이 파일은 Interact Temp 디렉토리, UNICA_ACTMPDIR에 있습니다.
<DBCOLUMNNUMBER>	이 토큰은 데이터베이스의 열 서수로 대체됩니다.
<FIELDLENGTH>	이 토큰은 데이터베이스에 로드 중인 필드의 길이로 대체됩니다.
<FIELDNAME>	이 토큰은 데이터베이스에 로드 중인 필드의 이름으로 대체됩니다.
<FIELDNUMBER>	이 토큰은 데이터베이스에 로드 중인 필드의 번호로 대체됩니다.
<FIELDTYPE>	이 토큰은 리터럴 "CHAR( )"로 대체됩니다. 이 필드의 길이는 () 사이에 지정됩니다. 데이터베이스에서 필드 유형, CHAR가 인식되지 않는 경우 수동으로 필드 유형에 대한 해당 텍스트를 지정하고 <FIELDLENGTH> 토큰을 사용할 수 있습니다. 예를 들어, SQLSVR 및 SQL2000의 경우 "SQLCHAR(<FIELDLENGTH>)"를 사용합니다.
<NATIVETYPE>	이 토큰은 이 필드가 로드되는 데이터베이스의 유형으로 대체됩니다.
<NUMFIELDS>	이 토큰은 테이블의 필드 수로 대체됩니다.
<PASSWORD>	이 토큰은 데이터 소스에 대한 현재 플로차트 연결의 데이터베이스 암호로 대체됩니다.
<TABLENAME>	이 토큰은 Interact가 데이터를 로드 중인 데이터베이스 테이블 이름으로 대체됩니다.
<USER>	이 토큰은 데이터 소스에 대한 현재 플로차트 연결의 데이터베이스 사용자로 대체됩니다.

기본값

정의된 기본값이 없습니다.

LoaderControlFileTemplateForAppend

설명

LoaderControlFileTemplateForAppend 등록 정보는 이전에 Interact에 구성된 컨트롤 파일 템플릿에 대한 전체 경로 및 파일 이름을 지정합니다. 이 매개변수가 설정되면 Interact는 여기에 지정된 템플릿에 기초하여 임시 컨트롤 파일을 동적으로 빌드합니다. 이 임시 컨트롤 파일의 경로와 이름은 LoaderCommandForAppend 등록 정보에 사용 가능한 <CONTROLFILE> 토큰에 사용할 수 있습니다.

Interact를 데이터베이스 로더 유틸리티 모드에서 사용하기 전에 이 매개변수로 지정된 컨트롤 파일 템플릿을 구성해야 합니다. 컨트롤 파일 템플릿은 Interact가 임시 컨트롤 파일을 생성할 때 동적으로 대체되는 다음 토큰을 지원합니다.

컨트롤 파일에 필요한 올바른 구문은 데이터베이스 로더 유틸리티 문서를 참조하십시오. 컨트롤 파일 템플릿에 사용 가능한 토큰은 LoaderControlFileTemplate 등록 정보에 대한 토큰과 동일합니다.

이 매개변수는 기본적으로 정의되어 있지 않습니다.

기본값

정의된 기본값이 없습니다.

LoaderDelimiterForAppend

설명

LoaderDelimiterForAppend 등록 정보는 임시 Interact 데이터 파일이 고정 너비 또는 구분된 플랫 파일인지 여부를 지정하고 구분된 경우 구분 기호로 사용된 문자 또는 문자 세트를 지정합니다.

값이 정의되지 않은 경우 Interact는 임시 데이터 파일을 고정 너비 플랫 파일로 생성합니다.

값을 지정하면 비어 있는 것으로 알려지지 않은 테이블을 채우기 위해 로더를 호출할 때 이 값이 사용됩니다. Interact는 이 등록 정보의 값을 구분 기호로 사용하여 임시 데이터 파일을 구분된 플랫 파일로 생성합니다.

이 등록 정보는 기본적으로 정의되어 있지 않습니다.

기본값

올바른 값

문자, 원하는 경우 큰따옴표로 묶을 수 있음

LoaderDelimiterAtEndForAppend

설명

일부 외부 로드 유틸리티는 데이터 파일을 구분하고 각 행이 구분 기호로 끝나야 합니다. 이 요구 사항을 충족시키려면 LoaderDelimiterAtEndForAppend

값을 TRUE로 설정해서 비어 있는 것으로 알려지지 않은 테이블을 채우기 위해 로더가 호출될 때 Interact가 각 행의 끝에 구분 기호를 사용하게 하십시오.

기본값

FALSE

올바른 값

TRUE | FALSE

LoaderUseLocaleDP

설명

LoaderUseLocaleDP 등록 정보는 Interact가 데이터베이스 로드 유틸리티로 로드할 파일에 숫자 값을 쓸 때 소수점에 로케일 특정 기호가 사용되는지 여부를 지정합니다.

마침표(.)가 소수점으로 사용됨을 지정하려면 이 값을 FALSE로 설정하십시오.

로케일에 해당하는 소수점 기호가 사용됨을 지정하려면 이 값을 TRUE로 설정하십시오.

기본값

FALSE

올바른 값

TRUE | FALSE

Interact | 일반 | testRunDataSource

이 구성 등록 정보는 Interact 디자인 환경에 대한 테스트 실행 테이블의 데이터 소스 설정을 정의합니다. 최소 하나의 런타임 환경에 대해 이 데이터 소스를 정의해야 합니다. 다음은 대화식 플로차트의 테스트 실행을 수행할 때 사용되는 테이블입니다.

jndiName

설명

이 jndiName 등록 정보를 사용하여 대화식 플로차트 테스트 실행을 실행할 때 디자인 환경이 액세스하는 고객 테이블의 응용 프로그램 서버(Websphere 또는 WebLogic)에 정의된 JNDI(Java Naming and Directory Interface) 데이터 소스를 식별하십시오.

기본값

정의된 기본값이 없습니다.

type

설명

대화식 플로차트 테스트 실행을 실행할 때 디자인 환경이 액세스하는 고객 테이블의 데이터베이스 유형입니다.

기본값

SQLServer

올바른 값

SQLServer | DB2 | ORACLE

aliasPrefix

설명

AliasPrefix 등록 정보는 대화식 플로차트 테스트 실행을 실행할 때 디자인 환경이 액세스하는 고객 테이블에 새 테이블을 쓰고 차원 테이블을 사용할 때 Interact가 자동으로 생성하는 별명 이름을 Interact가 형성하는 방식을 지정합니다.

각 데이터베이스에는 최대 ID 길이가 있음에 유의하십시오. 사용 중인 데이터베이스의 문서를 검토하여 설정한 값이 데이터베이스의 최대 ID 길이를 초과하지 않는지 확인하십시오.

기본값

A

connectionRetryPeriod

설명

ConnectionRetryPeriod 등록 정보는 테스트 실행 테이블에 장애가 발생할 경우 Interact가 자동으로 데이터베이스 연결 요청을 재시도하는 시간(초)을 지정합니다. Interact는 데이터베이스 오류나 실패를 보고하기 전에 이 시간 동안 자동으로 데이터베이스에 다시 연결하려 시도합니다. 0 값을 설정하면 Interact가 무한정 재시도하며 값을 -1로 설정하는 경우에는 재시도하지 않습니다.

기본값

-1

connectionRetryDelay

설명

ConnectionRetryDelay 등록 정보는 테스트 실행 테이블에 장애가 발생한 후 Interact가 데이터베이스에 다시 연결하려 시도하기 전에 대기하는 시간(초)을 지정합니다. 값을 -1로 설정하면 재시도하지 않습니다.

기본값

-1

schema

설명

대화식 플로차트 테스트 실행에 대한 테이블을 포함한 스키마의 이름입니다. Interact는 이 등록 정보의 값을 모든 테이블 이름 앞에 삽입합니다. 예를 들어, UACI_IntChannel이 schema.UACI_IntChannel이 됩니다.

스키마를 정의할 필요는 없습니다. 스키마를 정의하지 않으면 Interact는 테이블의 소유자가 스키마와 동일하다고 가정합니다. 모호하지 않도록 이 값을 설정해야 합니다.

기본값

정의된 기본값이 없습니다.

Interact | 일반 | contactAndResponseHistoryDataSource

이 구성 등록 정보는 Interact 교차 세션 응답 추적에 필요한 컨택 및 응답 기록 데이터 소스의 연결 설정을 정의합니다.

이 설정은 컨택 및 응답 기록 모듈과는 관련이 없습니다.

jndiName

설명

이 jndiName 등록 정보를 사용하여 Interact 교차 세션 응답 추적에 필요한 컨택 및 응답 기록 데이터 소스에 대한 응용 프로그램 서버(WebSphere 또는 WebLogic)에 정의된 JNDI(Java Naming and Directory Interface) 데이터 소스를 식별하십시오.

기본값

type

설명

Interact 교차 세션 응답 추적에 필요한 컨택 및 응답 기록 데이터 소스에 사용된 데이터 소스의 데이터베이스 유형입니다.

기본값

SQLServer

올바른 값

SQLServer | DB2 | ORACLE

connectionRetryPeriod

설명

ConnectionRetryPeriod 등록 정보는 Interact 교차 세션 응답 추적에 실패할 경우 Interact가 자동으로 데이터베이스 연결 요청을 재시도하는 시간(초)을 지정합니다. Interact는 데이터베이스 오류나 실패를 보고하기 전에 이 시간 동안 자동으로 데이터베이스에 다시 연결하려 시도합니다. 0 값을 설정하면 Interact가 무한정 재시도하며 값을 -1로 설정하는 경우에는 재시도하지 않습니다.

기본값

-1

connectionRetryDelay

설명

ConnectionRetryDelay 등록 정보는 Interact 교차 세션 응답 추적이 실패한 후 Interact가 데이터베이스에 다시 연결하려 시도하기 전에 대기하는 시간(초)을 지정합니다. 값을 -1로 설정하면 재시도하지 않습니다.

기본값

-1

schema

설명

Interact 교차 세션 응답 추적에 대한 테이블을 포함한 스키마의 이름입니다. Interact는 이 등록 정보의 값을 모든 테이블 이름 앞에 삽입합니다. 예를 들어, UACI_IntChannel이 schema.UACI_IntChannel이 됩니다.

스키마를 정의할 필요는 없습니다. 스키마를 정의하지 않으면 Interact는 테이블의 소유자가 스키마와 동일하다고 가정합니다. 모호하지 않도록 이 값을 설정해야 합니다.

기본값

정의된 기본값이 없습니다.

Interact | 일반 | idsByType

이 구성 등록 정보는 컨택 및 응답 기록 모듈에 사용되는 ID 번호의 설정을 정의합니다.

initialValue

설명

UACI_IDsByType 테이블을 사용하여 ID를 생성할 때 사용되는 초기 ID 값입니다.

기본값

1

올바른 값

0보다 큰 모든 값

retries

설명

UACI_IDsByType 테이블을 사용하여 ID를 생성할 때 예외가 생성되기 전의 재시도 수입니다.

기본값

20

올바른 값

0보다 큰 정수

Interact | 플로차트

이 섹션은 대화식 플로차트의 구성 설정을 정의합니다.

defaultDateFormat

설명

Interact에서 날짜를 문자열로 그리고 문자열을 날짜로 변환하기 위해 사용하는 기본 날짜 형식입니다.

기본값

MM/dd/yy

idleFlowchartThreadTimeoutInMinutes

설명

Interact에서 스레드를 해제하기 전 대화식 플로차트 전용 스레드의 유휴가 허용되는 시간(분)입니다.

기본값

5

idleProcessBoxThreadTimeoutInMinutes

설명

Interact에서 스레드를 해제하기 전 대화식 플로차트 프로세스 전용 스레드의 유휴가 허용되는 시간(분)입니다.

기본값

5

maxSizeOfFlowchartEngineInboundQueue

설명

Interact가 대기열에 보유하는 플로차트 실행 요청의 최대 수입니다. 이 요청 수에 도달하면 Interact가 더 이상 요청을 받지 않습니다.

기본값

1000

maxNumberOfFlowchartThreads

설명

대화식 플로차트 요청 전용 스레드의 최대 수입니다.

기본값

25

maxNumberOfProcessBoxThreads

설명

대화식 플로차트 프로세스 전용 스레드의 최대 수입니다.

기본값

50

maxNumberOfProcessBoxThreadsPerFlowchart

설명

플로차트 인스턴스별 대화식 플로차트 프로세스 전용 스레드의 최대 수입니다.

기본값

3

minNumberOfFlowchartThreads

설명

대화식 플로차트 요청 전용 스레드의 최소 수입니다.

기본값

10

minNumberOfProcessBoxThreads

설명

대화식 플로차트 프로세스 전용 스레드의 최소 수입니다.

기본값

20

sessionVarPrefix

설명

세션 변수의 접두부입니다.

기본값

SessionVar

Interact | 플로차트 | ExternalCallouts | [ExternalCalloutName]

이 섹션은 외부 콜아웃 API로 쓴 사용자 정의 외부 콜아웃에 대한 클래스 설정을 정의합니다.

class

설명

이 외부 콜아웃이 표시하는 Java 클래스의 이름입니다.

IBM Unica Macro `EXTERNALCALLOUT`으로 액세스할 수 있는 Java 클래스입니다.

기본값

정의된 기본값이 없습니다.

classpath

설명

이 외부 콜아웃이 표시하는 Java 클래스의 클래스 경로입니다. 클래스 경로는 런타임 환경 서버의 jar 파일을 참조해야 합니다. 서버 그룹을 사용 중이며 모든 런타임 서버가 동일한 Marketing Platform을 사용하는 경우 모든 서버의 jar 파일 사본이 같은 위치에 있어야 합니다. 클래스 경로는 런타임 환경 서버의 경로 구분 기호(예를 들어, Windows의 세미콜론(;) 및 UNIX 시스템의 콜론(:))로 구분된 jar 파일의 절대 위치로 이루어집니다. 클래스 파일이 포함된 디렉토리는 액세스되지 않습니다. 예를 들어, Unix 시스템에서 `/path1/file1.jar:/path2/file2.jar`입니다.

이 클래스 경로는 1024자 미만이어야 합니다. .jar 파일의 속성 정의 파일을 사용하여 다른 .jar 파일을 지정해서 클래스 경로에 .jar 파일이 하나만 표시되게 할 수 있습니다.

IBM Unica Macro EXTERNALCALLOUT으로 액세스할 수 있는 Java 클래스입니다.

기본값

정의된 기본값이 없습니다.

Interact | 플로차트 | ExternalCallouts | [ExternalCalloutName] | 매개변수 데이터 | [parameterName]

이 섹션은 외부 콜아웃 API로 쓴 사용자 정의 외부 콜아웃에 대한 매개변수 설정을 정의합니다.

value

설명

외부 콜아웃의 클래스에 필요한 매개변수의 값입니다.

기본값

정의된 기본값이 없습니다.

예제

외부 콜아웃에 외부 서버의 호스트 이름이 필요하면 호스트라는 매개변수 범주를 생성하고 value 등록 정보를 서버 이름으로 정의하십시오.

Interact | 모니터링

이 구성 등록 정보 세트로 JMX 모니터링 설정을 정의할 수 있습니다. JMX 모니터링을 사용 중인 경우에만 이 등록 정보를 구성해야 합니다.

Interact 디자인 환경의 구성 등록 정보에는 컨택 및 응답 기록 모듈에 대해 정의하는 별도의 JMX 모니터링 등록 정보가 있습니다.

protocol

설명

Interact 메시징 서비스의 프로토콜을 정의합니다.

JMXMP를 선택하면 다음 JAR 파일을 클래스 경로에 다음 순서대로 포함해야 합니다.

```
Interact/lib/InteractJMX.jar;Interact/lib/jmxremote_optional.jar
```

기본값

JMXMP

올바른 값

JMXMP | RMI

port

설명

메시징 서비스의 포트 번호입니다.

기본값

9998

enableSecurity

설명

Interact 런타임 서버의 메시징 서비스 보안을 설정하거나 비활성화하는 부울입니다. true로 설정하면 Interact 런타임 JMX 서비스에 액세스할 사용자 이름과 암호를 제공해야 합니다. 이 사용자 신임 정보는 런타임 서버의 Marketing Platform에서 인증됩니다. Jconsole에서는 비어 있는 암호 로그인이 허용되지 않습니다.

프로토콜이 RMI인 경우에는 이 등록 정보가 아무런 영향이 없습니다. Campaign에 대한 JMX(Interact 디자인 시간)에는 이 등록 정보가 아무런 영향이 없습니다.

기본값

True

올바른 값

True | False

Interact | 프로파일

이 구성 등록 정보 세트는 오퍼 제외와 점수 재정의를 포함한 여러 선택적 오퍼 제공 기능을 컨트롤합니다.

enableScoreOverrideLookup

설명

True로 설정하면 세션을 생성할 때 Interact가 scoreOverrideTable에서 점수 재정의 데이터를 로드합니다. False인 경우에는 Interact가 세션을 생성할 때 마케팅 점수 재정의 데이터를 로드하지 않습니다.

true일 경우 Unica > Interact > 프로파일 > 대상 수준 > (대상 수준) > scoreOverrideTable 등록 정보도 구성해야 합니다. scoreOverrideTable 등록 정보를 필요한 대상 수준에 대해서만 정의해야 합니다. 대상 수준에 대한 scoreOverrideTable을 공백으로 두면 대상 수준에 대한 점수 재정의 테이블이 비활성화됩니다.

기본값

False

올바른 값

True | False

enableOfferSuppressionLookup

설명

True로 설정하면 세션을 생성할 때 Interact가 offerSuppressionTable에서 오퍼 제외 데이터를 로드합니다. False인 경우에는 Interact가 세션을 생성할 때 오퍼 제외 데이터를 로드하지 않습니다.

true일 경우 Unica > Interact > 프로파일 > 대상 수준 > (대상 수준) > offerSuppressionTable 등록 정보도 구성해야 합니다. enableOfferSuppressionLookup 등록 정보를 필요한 대상 수준에 대해서만 정의해야 합니다.

기본값

False

올바른 값

True | False

enableProfileLookup

설명

Interact의 새 설치에서는 이 등록 정보가 더 이상 사용되지 않습니다. Interact의 업그레이드된 설치에서는 첫 번째 배포 때까지 이 등록 정보가 유효합니다.

대화식 플로차트에 사용되지만 대화식 채널에서 맵핑되지 않은 테이블의 로드 작동입니다. True로 설정하면 세션을 생성할 때 Interact가 profileTable에서 프로파일 데이터를 로드합니다.

true일 경우 Unica > Interact > 프로파일 > 대상 수준 > (대상 수준) > profileTable 등록 정보도 구성해야 합니다.

대화식 채널 테이블 맵핑 마법사의 **방문 세션이 시작될 때 메모리에 이 데이터 로드** 설정은 이 구성 등록 정보를 재정의합니다.

기본값

False

올바른 값

True | False

defaultOfferUpdatePollPeriod

설명

시스템이 기본 오퍼 테이블에서 캐시의 기본 오퍼를 업데이트하기 전에 대기하는 시간(초)입니다. -1로 설정하면 런타임 서버가 시작될 때 초기 목록이 캐시로 로드된 후 시스템이 캐시의 기본 오퍼를 업데이트하지 않습니다.

기본값

-1

Interact | 프로파일 | 대상 수준 | [AudienceLevelName]

이 구성 등록 정보 세트로 추가 Interact 기능에 필요한 테이블 이름을 정의할 수 있습니다. 연관된 기능을 사용 중인 경우 테이블 이름만 정의합니다.

scoreOverrideTable

설명

이 대상 수준에 대한 점수 재정의 정보를 포함한 테이블의 이름입니다. 이 등록 정보는 enableScoreOverrideLookup을 true로 설정한 경우에 적용할 수 있습니다. 점수 재정의 테이블을 사용하려는 대상 수준에 대해 이 등록 정보를 정의해야 합니다. 이 대상 수준에 대한 점수 재정의 테이블이 없으면 enableScoreOverrideLookup이 true로 설정되어 있어도 이 등록 정보를 정의되지 않은 채로 둘 수 있습니다.

Interact는 Interact 런타임 서버가 액세스하며 prodUserDataSource 등록 정보로 정의된 고객 테이블에서 이 테이블을 찾습니다.

이 데이터 소스에 대한 schema 등록 정보를 정의한 경우에는 Interact가 이 테이블 이름 앞에 schema를 추가합니다(예: schema.UACI_ScoreOverride). 완전한 이름을 입력하면(예: mySchema.UACI_ScoreOverride) Interact가 스키마 이름을 추가하지 않습니다.

기본값

UACI_ScoreOverride

offerSuppressionTable

설명

이 대상 수준에 대한 오퍼 제외 정보를 포함한 테이블의 이름입니다. 오퍼 제외 테이블을 사용하려는 대상 수준에 대해 이 등록 정보를 정의해야 합니다. 이 대상 수준에 대한 오퍼 제외 테이블이 없으면 enableOfferSuppressionLookup이 true로 설정되어 있어도 이 등록 정보를 정의되지 않은 채로 둘 수 있습니다.

Interact는 런타임 서버가 액세스하며 prodUserDataSource 등록 정보로 정의된 고객 테이블에서 이 테이블을 찾습니다.

기본값

UACI_BlackList

profileTable

설명

Interact의 새 설치에서는 이 등록 정보가 더 이상 사용되지 않습니다. Interact의 업그레이드된 설치에서는 첫 번째 배포 때까지 이 등록 정보가 유효합니다.

이 대상 수준에 대한 프로파일 데이터를 포함한 테이블의 이름입니다.

Interact는 런타임 서버가 액세스하며 prodUserDataSource 등록 정보로 정의된 고객 테이블에서 이 테이블을 찾습니다.

이 데이터 소스에 대한 schema 등록 정보를 정의한 경우에는 Interact가 이 테이블 이름 앞에 schema를 추가합니다(예: schema.UACI_usrProd). 완전한 이름을 입력하면(예: mySchema.UACI_usrProd) Interact가 스키마 이름을 추가하지 않습니다.

기본값

정의된 기본값이 없습니다.

contactHistoryTable

설명

이 대상 수준에 대한 컨택 기록 데이터의 스테이징 테이블 이름입니다.

이 테이블은 런타임 환경 테이블(systemTablesDataSource)에 저장됩니다.

이 데이터 소스에 대한 schema 등록 정보를 정의한 경우에는 Interact가 이 테이블 이름 앞에 schema를 추가합니다(예: schema.UACI_CHStaging). 완전한 이름을 입력하면(예: mySchema.UACI_CHStaging) Interact가 스키마 이름을 추가하지 않습니다.

기본값

UACI_CHStaging

chOfferAttribTable

설명

이 대상 수준에 대한 컨택 기록 오퍼 속성 테이블의 이름입니다.

이 테이블은 런타임 환경 테이블(systemTablesDataSource)에 저장됩니다.

이 데이터 소스에 대한 schema 등록 정보를 정의한 경우에는 Interact가 이 테이블 이름 앞에 schema를 추가합니다(예: schema.UACI_CHOfferAttrib). 완전한 이름을 입력하면(예: mySchema.UACI_CHOfferAttrib) Interact가 스키마 이름을 추가하지 않습니다.

기본값

UACI_CHOfferAttrib

responseHistoryTable

설명

이 대상 수준에 대한 응답 기록 스테이징 테이블의 이름입니다.

이 테이블은 런타임 환경 테이블(systemTablesDataSource)에 저장됩니다.

이 데이터 소스에 대한 schema 등록 정보를 정의한 경우에는 Interact가 이 테이블 이름 앞에 schema를 추가합니다(예: schema.UACI_RHStaging). 완전한 이름을 입력하면(예: mySchema.UACI_RHStaging) Interact가 스키마 이름을 추가하지 않습니다.

기본값

UACI_RHStaging

crossSessionResponseTable

설명

응답 추적 기능에 액세스 가능한 컨택 및 응답 기록 테이블의 교차 세션 응답 추적에 필요한 이 대상 수준에 대한 테이블의 이름입니다.

이 데이터 소스에 대한 schema 등록 정보를 정의한 경우에는 Interact가 이 테이블 이름 앞에 schema를 추가합니다(예: schema.UACI_XSessResponse). 완전한 이름을 입력하면(예: mySchema.UACI_XSessResponse) Interact가 스키마 이름을 추가하지 않습니다.

기본값

UACI_XSessResponse

Interact | 프로파일 | 대상 수준 | [AudienceLevelName] | 원시 SQL 기준 오퍼

이 구성 등록 정보 세트로 추가 Interact 기능에 필요한 테이블 이름을 정의할 수 있습니다. 연관된 기능을 사용 중인 경우 테이블 이름만 정의하면 합니다.

enableOffersByRawSQL

설명

True로 설정하면 Interact가 이 대상 수준의 offersBySQL 기능을 사용하여 사용자가 런타임에 원하는 후보 오퍼 세트를 생성하기 위해 실행할 SQL 코드를 구성할 수 있습니다. False인 경우에는 Interact가 offersBySQL 기능을 사용하지 않습니다.

이 등록 정보를 true로 설정하는 경우 Unica | Interact | 프로파일 | 대상 수준 | (대상 수준1) | 원시 SQL 기준 오퍼 | SQL 템플릿 등록 정보도 구성하여 하나 이상의 SQL 템플릿을 정의해야 합니다.

기본값

False

올바른 값

True | False

cacheSize

설명

OfferBySQL 쿼리의 결과를 저장하는 데 사용되는 캐시의 크기입니다. 캐시를 사용하면 대부분의 세션에서 쿼리 결과가 고유한 경우 부정적인 영향이 미칠 수 있음에 유의하십시오.

기본값

-1(해제)

올바른 값

-1 | 값

cacheLifeInMinutes

설명

캐시를 사용하는 경우 이는 시스템이 실효를 피하기 위해 캐시를 지우기 전의 시간(분)을 나타냅니다.

기본값

-1(해제)

올바른 값

-1 | 값

defaultSQLTemplate

설명

API 호출을 통해 지정되지 않은 경우에 사용할 SQL 템플릿의 이름입니다.

기본값

없음

올바른 값

SQL 템플릿 이름

Interact | 프로파일 | 대상 수준 | [AudienceLevelName] | SQL 템플릿

이 구성 등록 정보는 Interact의 offersBySQL 기능에 사용되는 하나 이상의 SQL 쿼리 템플릿을 정의합니다.

name

설명

이 SQL 쿼리 템플릿에 할당하려는 이름입니다. API 호출에서 이 SQL 템플릿을 사용할 때 의미 있는 설명하는 이름을 입력하십시오. offerBySQL 처리의 대화식 목록 프로세스 상자에 정의된 이름과 동일한 이름을 여기에 사용하면 여기에 입력한 SQL 대신 프로세스 상자의 SQL이 사용됨에 유의하십시오.

기본값

없음

SQL

설명

이 템플릿으로 호출할 SQL 쿼리를 포함합니다. SQL 쿼리는 방문자 세션 데이터(프로파일)의 일부인 변수 이름에 대한 참조를 포함할 수 있습니다. 예를 들어, `select * from MyOffers where category = ${preferredCategory}`는 `preferredCategory`라는 변수를 포함한 세션에 의존합니다.

디자인 시간에 사용하도록 이 기능으로 생성한 특정 오퍼 테이블을 쿼리하도록 SQL을 구성해야 합니다. 스토어드 프로시저는 여기에서 지원되지 않음에 유의하십시오.

기본값

없음

Interact | 프로파일 | 대상 수준 | [AudienceLevelName] | 프로파일 데이터 서비스 | [DataSource]

이 구성 등록 정보 세트로 추가 Interact 기능에 필요한 테이블 이름을 정의할 수 있습니다. 연관된 기능을 사용 중인 경우 테이블 이름만 정의합니다. 프로파일 데이터 서비스 범주는 모든 대상 수준에 대해 생성되는 기본 제공 데이터 소스(데이터베이스라고 함)와 어떤 것이 우선순위가 100으로 사전 구성되어 있는지에 대한 정보를 제공합니다. 하지만, 이를 수정하거나 비활성화할 수 있습니다. 이 범주에는 추가 외부 데이터 소스에 대한 템플릿도 포함됩니다. **외부 데이터 서비스**라고 하는 템플릿을 클릭하면 여기서 설명한 구성 설정을 완료할 수 있습니다.

New category name

설명

(기본 데이터베이스 항목에는 사용할 수 없습니다.) 정의하고 있는 데이터 소스의 이름입니다. 여기서 입력하는 이름은 같은 대상 수준에 대해 데이터 소스 사이에서 고유해야 합니다.

기본값

없음

올바른 값

어떤 텍스트 문자열이라도 허용됩니다.

enabled

설명

`True`로 설정되어 있는 경우, 이 데이터 소스 Interact은 자신이 할당된 대상 수준에 대해 활성화됩니다. `False`인 경우 Interact에서는 이 대상 수준에 대해 이 데이터 소스를 사용하지 않습니다.

기본값

`True`

올바른 값

True | False

className

설명

(기본 데이터베이스 항목에는 사용할 수 없습니다.)
`IInteractProfileDataService`를 구현하는 데이터 소스 클래스의 완전한 이름입니다.

기본값

없음

올바른 값

완전한 클래스 이름을 제공하는 문자열입니다.

classPath

설명

(기본 데이터베이스 항목에는 사용할 수 없습니다.) 이 데이터 소스 구현 클래스를 로드하기 위한 경로를 제공하는 선택적 구성 설정입니다. 이것을 생략하면 포함하는 응용 프로그램 서버의 클래스 경로가 기본적으로 사용됩니다.

기본값

여기서 아무런 값도 입력하지 않으면 포함하는 응용 프로그램 서버의 클래스 경로가 표시되지는 않지만 기본적으로 사용됩니다.

올바른 값

클래스 경로를 제공하는 문자열입니다.

priority

설명

이 대상 수준 내부에서 이 데이터 소스의 우선순위입니다. 각 대상 수준에 대한 모든 데이터 소스 사이에서 고유한 값이어야 합니다. (즉, 어떤 데이터 소스에 대해 우선순위가 100으로 설정된 경우 해당 대상 수준 내에 있는 다른 데이터 소스는 그 어떤 것도 100의 우선순위를 가질 수 없습니다.)

기본값

기본 데이터베이스는 100, 사용자 정의 데이터 소스는 200이 기본값입니다.

올바른 값

음수가 아닌 어떤 정수라도 허용됩니다.

Interact | offerserving

이 구성 등록 정보는 일반 학습 구성 등록 정보를 정의합니다.

기본 제공 학습을 사용 중인 경우 학습 구현을 조정하려면 디자인 환경의 구성 등록 정보를 사용하십시오.

optimizationType

설명

optimizationType 등록 정보는 Interact가 오퍼 지정을 지원할 학습 엔진을 사용하는지 여부를 정의합니다. NoLearning으로 설정하면 Interact가 학습을 사용하지 않습니다. BuiltInLearning으로 설정한 경우에는 Interact가 Interact로 빌드된 베이지안 학습 엔진을 사용합니다. ExternalLearning으로 설정하면 Interact는 사용자가 제공한 학습 엔진을 사용합니다. ExternalLearning을 선택하는 경우 externalLearningClass 및 externalLearningClassPath 등록 정보를 정의해야 합니다.

기본값

NoLearning

올바른 값

NoLearning | BuiltInLearning | ExternalLearning

segmentationMaxWaitTimeInMS

설명

런타임 서버가 오퍼를 받기 전에 대화식 플로차트가 완료될 때까지 대기하는 최대 시간(밀리초)입니다.

기본값

5000

treatmentCodePrefix

설명

처리 코드에 미리 추가되는 접두부입니다.

기본값

정의된 기본값이 없습니다.

Interact | offerserving | 기본 제공 학습 구성

이 구성 등록 정보는 기본 제공 학습의 데이터베이스 쓰기 설정을 정의합니다.

학습 구현을 조정하려면 디자인 환경의 구성 등록 정보를 사용하십시오.

insertRawStatsIntervalInMinutes

설명

Interact 학습 모듈이 학습 스테이징 테이블에 더 많은 행을 삽입하기 전에 대기하는 시간(분)입니다. 학습 모듈이 환경에서 처리 중인 데이터 양에 따라 이 시간을 수정해야 합니다.

기본값

5

aggregateStatsIntervalInMinutes

설명

Interact 학습 모듈이 학습 스테이징 테이블에 데이터를 집계하는 사이에 대기하는 시간(분)입니다. 학습 모듈이 환경에서 처리 중인 데이터 양에 따라 이 시간을 수정해야 합니다.

기본값

15

올바른 값

0보다 큰 정수

Interact | offerserving | 외부 학습 구성

이 구성 등록 정보는 학습 API를 사용하여 쓴 외부 학습 모듈의 클래스 설정을 정의합니다.

class

설명

optimizationType이 ExternalLearning으로 설정된 경우 externalLearningClass를 외부 학습 엔진의 클래스 이름으로 설정하십시오.

기본값

정의된 기본값이 없습니다.

가용성

이 등록 정보는 optimizationType이 ExternalLearning으로 설정된 경우에만 적용할 수 있습니다.

classPath

설명

optimizationType이 ExternalLearning으로 설정된 경우 externalLearningClass를 외부 학습 엔진의 클래스 경로로 설정하십시오.

클래스 경로는 런타임 환경 서버의 jar 파일을 참조해야 합니다. 서버 그룹을 사용 중이며 모든 런타임 서버가 동일한 Marketing Platform을 사용하는 경우 모든 서버의 jar 파일 사본이 같은 위치에 있어야 합니다. 클래스 경로는 런타임 환경 서버의 경로 구분 기호(예를 들어, Windows의 세미콜론(;) 및 UNIX 시스템의 콜론(:))로 구분된 jar 파일의 절대 위치로 이루어집니다. 클래스 파일

이 포함된 디렉토리는 액세스되지 않습니다. 예를 들어, Unix 시스템에서 /path1/file1.jar:/path2/file2.jar입니다.

이 클래스 경로는 1024자 미만이어야 합니다. .jar 파일의 속성 정의 파일을 사용하여 다른 .jar 파일을 지정해서 클래스 경로에 .jar 파일이 하나만 표시되게 할 수 있습니다.

기본값

정의된 기본값이 없습니다.

가용성

이 등록 정보는 optimizationType이 ExternalLearning으로 설정된 경우에만 적용할 수 있습니다.

Interact | offerserving | 외부 학습 구성 | 매개변수 데이터 | [parameterName]

이 구성 등록 정보는 외부 학습 모듈의 매개변수를 정의합니다.

value

설명

외부 학습 모듈의 클래스에 필요한 매개변수의 값입니다.

기본값

정의된 기본값이 없습니다.

예제

외부 학습 모듈에 알고리즘 해석 응용 프로그램이 필요한 경우 solverPath라는 매개변수 범주를 생성하고 value 등록 정보를 응용 프로그램에 대한 경로로 정의합니다.

Interact | 서비스

이 범주의 구성 등록 정보는 컨택 및 응답 기록 데이터를 수집하고 런타임 환경 시스템 테이블에 이 데이터를 보고하고 쓰는 것에 대한 통계를 관리하는 모든 서비스의 설정을 정의합니다.

externalLoaderStagingDirectory

설명

이 등록 정보는 데이터베이스 로드 유틸리티의 스테이징 디렉토리 위치를 정의합니다.

기본값

정의된 기본값이 없습니다.

올바른 값

Interact 설치 디렉토리의 상대 경로 또는 스테이징 디렉토리의 절대 경로.

데이터베이스 로드 유틸리티를 사용하는 경우 contactHist 및 responstHist 범주의 cacheType 등록 정보를 외부 로더 파일로 설정해야 합니다.

Interact | 서비스 | contactHist

이 범주의 구성 등록 정보는 컨택 기록 스테이징 테이블에 대한 데이터를 수집하는 서비스의 설정을 정의합니다.

enableLog

설명

true이면 컨택 기록 데이터 레코딩을 위해 데이터를 수집하는 서비스가 사용됩니다. false인 경우에는 데이터가 수집되지 않습니다.

기본값

True

올바른 값

True | False

cacheType

설명

컨택 기록을 위해 수집된 데이터가 메모리(메모리 캐시) 또는 파일(외부 로더 파일)에 보관되는지 여부를 정의합니다. 외부 로더 파일은 Interact를 데이터베이스 로더 유틸리티를 사용하도록 구성한 경우에만 사용할 수 있습니다.

메모리 캐시를 선택한 경우 캐시 범주 설정을 사용하십시오. 외부 로더 파일을 선택하는 경우에는 fileCache 범주 설정을 사용하십시오.

기본값

메모리 캐시

올바른 값

메모리 캐시 | 외부 로더 파일

Interact | 서비스 | contactHist | 캐시

이 범주의 구성 등록 정보는 컨택 기록 스테이징 테이블에 대한 데이터를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 컨택 기록 데이터를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | contactHist | fileCache

이 범주의 구성 등록 정보는 데이터베이스 로더 유틸리티를 사용 중인 경우 컨택 기록 데이터를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 컨택 기록 데이터를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | defaultedStats

이 범주의 구성 등록 정보는 상호작용 지점에 대한 기본 문자열이 사용된 횟수에 관한 통계를 수집하는 서비스의 설정을 정의합니다.

enableLog

설명

true이면 상호작용 지점에 대한 기본 문자열이 UACI_DefaultedStat 테이블에 사용된 횟수에 관한 통계를 수집하는 서비스가 사용됩니다. false인 경우에는 기본 문자열 통계가 수집되지 않습니다.

IBM 보고를 사용 중이 아니면 데이터 콜렉션이 필요하지 않으므로 이 등록 정보를 false로 설정할 수 있습니다.

기본값

True

올바른 값

True | False

Interact | 서비스 | defaultedStats | 캐시

이 범주의 구성 등록 정보는 상호작용 지점에 대한 기본 문자열이 사용된 횟수에 관한 통계를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 기본 문자열 통계를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | eligOpsStats

이 범주의 구성 등록 정보는 적합한 오퍼에 대한 통계를 쓰는 서비스의 설정을 정의합니다.

enableLog

설명

true이면 적합한 오퍼에 관한 통계를 수집하는 서비스가 사용됩니다. false인 경우에는 적합한 오퍼 통계가 수집되지 않습니다.

IBM 보고를 사용 중이 아니면 데이터 콜렉션이 필요하지 않으므로 이 등록 정보를 false로 설정할 수 있습니다.

기본값

True

올바른 값

True | False

Interact | 서비스 | eligOpsStats | 캐시

이 범주의 구성 등록 정보는 적합한 오퍼 통계를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 적합한 오퍼 통계를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | eventActivity

이 범주의 구성 등록 정보는 이벤트 활동 통계를 수집하는 서비스의 설정을 정의합니다.

enableLog

설명

true이면 이벤트 활동 통계를 수집하는 서비스가 사용됩니다. false인 경우에는 이벤트 통계가 수집되지 않습니다.

IBM 보고를 사용 중이 아니면 데이터 콜렉션이 필요하지 않으므로 이 등록 정보를 false로 설정할 수 있습니다.

기본값

True

올바른 값

True | False

Interact | 서비스 | eventActivity | 캐시

이 범주의 구성 등록 정보는 이벤트 활동 통계를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 이벤트 활동 통계를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | customLogger

이 범주의 구성 등록 정보는 테이블에 쓸 사용자 정의 데이터를 수집하는 서비스의 설정을 정의합니다(UACICustomLoggerTableName 이벤트 매개변수를 사용하는 이벤트).

enableLog

설명

true이면 테이블에 대한 사용자 정의 로그 기능이 사용됩니다. false인 경우에는 UACICustomLoggerTableName 이벤트 매개변수가 적용되지 않습니다.

기본값

True

올바른 값

True | False

Interact | 서비스 | customLogger | 캐시

이 범주의 구성 등록 정보는 테이블에 쓸 사용자 정의 데이터를 수집하는 서비스의 캐시 설정을 정의합니다(UACICustomLoggerTableName 이벤트 매개변수를 사용하는 이벤트).

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 사용자 정의 데이터를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | responseHist

이 범주의 구성 등록 정보는 응답 기록 스테이징 테이블에 쓰는 서비스의 설정을 정의합니다.

enableLog

설명

true이면 응답 기록 스테이징 테이블에 쓰는 서비스가 사용됩니다. false인 경우에는 응답 기록 스테이징 테이블에 데이터를 쓰지 않습니다.

응답 기록 스테이징 테이블은 대상 수준에 대한 responseHistoryTable 등록 정보로 정의됩니다. 기본값은 UACI_RHStaging입니다.

기본값

True

올바른 값

True | False

cacheType

설명

캐시가 메모리 또는 파일에 보관되는지 여부를 정의합니다. 외부 로더 파일은 Interact를 데이터베이스 로더 유틸리티를 사용하도록 구성한 경우에만 사용할 수 있습니다.

메모리 캐시를 선택한 경우 캐시 범주 설정을 사용하십시오. 외부 로더 파일을 선택하는 경우에는 fileCache 범주 설정을 사용하십시오.

기본값

메모리 캐시

올바른 값

메모리 캐시 | 외부 로더 파일

Interact | 서비스 | responseHist | 캐시

이 범주의 구성 등록 정보는 응답 기록 데이터를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 응답 기록 데이터를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | responseHist | fileCache

이 범주의 구성 등록 정보는 데이터베이스 로더 유틸리티를 사용 중인 경우 응답 기록 데이터를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

Interact가 데이터베이스에 레코드를 쓰기 전의 누적된 레코드 수입니다.

responseHist - 대상 수준에 대한 responseHistoryTable 등록 정보로 정의된 테이블입니다. 기본값은 UACI_RHStaging입니다.

기본값

100

insertPeriodInSecs

설명

데이터베이스 쓰기를 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | crossSessionResponse

이 범주의 구성 등록 정보는 crossSessionResponse 서비스와 xsession 프로세스의 일반 설정을 정의합니다. Interact 교차 세션 응답 추적을 사용 중인 경우에만 이 설정을 구성해야 합니다.

enableLog

설명

true이면 crossSessionResponse 서비스가 사용되고 Interact는 교차 세션 응답 추적 스테이징 테이블에 데이터를 씁니다. false인 경우에는 crossSessionResponse 서비스를 사용하지 않습니다.

기본값

False

xsessionProcessIntervalInSecs

설명

xsession 프로세스의 실행 간격(초)입니다. 이 프로세스는 교차 세션 응답 추적 스테이징 테이블에서 응답 기록 스테이징 테이블 및 기본 제공 학습 모듈로 데이터를 이동시킵니다.

기본값

180

올바른 값

0보다 큰 정수

purgeOrphanResponseThresholdInMinutes

설명

crossSessionResponse 서비스가 컨택 및 응답 기록 테이블의 컨택에 일치하지 않는 응답을 표시하기 전에 대기하는 시간(분)입니다.

컨택 및 응답 기록 테이블의 응답이 일치하지 않으면 purgeOrphanResponseThresholdInMinutes분 후 Interact는 xSessResponse 스테이징 테이블의 표시 열에 -1 값으로 응답을 표시합니다. 그러면 사용자는 수동으로 이 응답을 일치시키거나 삭제할 수 있습니다.

기본값

180

Interact | 서비스 | crossSessionResponse | 캐시

이 범주의 구성 등록 정보는 교차 세션 응답 데이터를 수집하는 서비스의 캐시 설정을 정의합니다.

threshold

설명

flushCacheToDB 서비스가 데이터베이스에 수집된 교차 세션 응답 데이터를 쓰기 전의 누적된 레코드 수입니다.

기본값

100

insertPeriodInSecs

설명

XSessResponse 테이블에 쓰도록 강제 실행하는 간격(초)입니다.

기본값

3600

Interact | 서비스 | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byTreatmentCode

이 섹션의 등록 정보는 교차 세션 응답 추적이 처리 코드를 컨택 및 응답 기록에 일치시키는 방법을 정의합니다.

SQL

설명

이 등록 정보는 Interact가 시스템 생성 SQL 또는 OverrideSQL 등록 정보에 정의된 사용자 정의 SQL을 사용하는지 여부를 정의합니다.

기본값

시스템 생성 SQL 사용

올바른 값

시스템 생성 SQL 사용 | SQL 재정의

OverrideSQL

설명

컨택 및 응답 기록에 처리 코드를 일치시키기 위해 기본 SQL 명령을 사용하지 않는 경우 SQL 또는 스토어드 프로시저를 여기에 입력하십시오.

SQL이 시스템 생성 SQL 사용으로 설정된 경우에는 이 값이 무시됩니다.

기본값

useStoredProcedure

설명

true로 설정하면 컨택 및 응답 기록에 처리 코드를 일치시키는 스토어드 프로시저에 대한 참조가 OverrideSQL에 포함되어 있어야 합니다.

false로 설정하는 경우에는 OverrideSQL이 SQL 쿼리여야 합니다(사용된 경우).

기본값

false

올바른 값

true | false

Type

설명

런타임 환경 테이블의 UACI_TrackingType 테이블에 정의된 연관이 있는 TrackingCodeType입니다. UACI_TrackingType 테이블을 수정하는 경우가 아니면 Type은 1이어야 합니다.

기본값

1

올바른 값

UACI_TrackingType 테이블에 정의된 정수

Interact | 서비스 | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byOfferCode

이 섹션의 등록 정보는 교차 세션 응답 추적이 오퍼 코드를 컨택 및 응답 기록에 일치시키는 방법을 정의합니다.

SQL

설명

이 등록 정보는 Interact가 시스템 생성 SQL 또는 OverrideSQL 등록 정보에 정의된 사용자 정의 SQL을 사용하는지 여부를 정의합니다.

기본값

시스템 생성 SQL 사용

올바른 값

시스템 생성 SQL 사용 | SQL 재정의

OverrideSQL

설명

컨택 및 응답 기록에 오퍼 코드를 일치시키기 위해 기본 SQL 명령을 사용하지 않는 경우 SQL 또는 스토어드 프로시저를 여기에 입력하십시오.

SQL이 시스템 생성 SQL 사용으로 설정된 경우에는 이 값이 무시됩니다.

기본값

useStoredProcedure

설명

true로 설정하면 컨택 및 응답 기록에 오퍼 코드를 일치시키는 스토어드 프로시저에 대한 참조가 OverrideSQL에 포함되어 있어야 합니다.

false로 설정하는 경우에는 OverrideSQL이 SQL 쿼리여야 합니다(사용된 경우).

기본값

false

올바른 값

true | false

Type

설명

런타임 환경 테이블의 UACI_TrackingType 테이블에 정의된 연관이 있는 TrackingCodeType입니다. UACI_TrackingType 테이블을 수정하는 경우가 아니면 Type은 2여야 합니다.

기본값

2

올바른 값

UACI_TrackingType 테이블에 정의된 정수

Interact | 서비스 | crossSessionResponse | OverridePerAudience | [AudienceLevel] | TrackingCodes | byAlternateCode

이 섹션의 등록 정보는 교차 세션 응답 추적이 사용자 정의 대체 코드를 컨택 및 응답 기록에 일치시키는 방법을 정의합니다.

Name

설명

이 등록 정보는 대체 코드의 이름을 정의합니다. 런타임 환경 테이블의 UACI_TrackingType 테이블에 있는 이름 값과 일치해야 합니다.

기본값

OverrideSQL

설명

오퍼 코드나 처리 코드로 컨택 및 응답 기록에 대체 코드를 일치시킬 SQL 명령 또는 스토어드 프로시저입니다.

기본값

useStoredProcedure

설명

true로 설정하면 컨택 및 응답 기록에 대체 코드를 일치시키는 스토어드 프로시저에 대한 참조가 OverrideSQL에 포함되어 있어야 합니다.

false로 설정하는 경우에는 OverrideSQL이 SQL 쿼리여야 합니다(사용된 경우).

기본값

false

올바른 값

true | false

Type

설명

런타임 환경 테이블의 UACI_TrackingType 테이블에 정의된 연관이 있는 TrackingCodeType입니다.

기본값

3

올바른 값

UACI_TrackingType 테이블에 정의된 정수

Interact | 서비스 | threadManagement | contactAndResponseHist

이 범주의 구성 등록 정보는 컨택 및 응답 기록 스테이징 테이블에 대한 데이터를 수집하는 서비스의 스레드 관리 설정을 정의합니다.

corePoolSize

설명

컨택 및 응답 기록 데이터 수집을 위해 풀에 보관할 스레드 수입니다(유휴 상태도 포함).

기본값

5

maxPoolSize

설명

컨택 및 응답 기록 데이터 수집을 위해 풀에 보관할 스레드의 최대 수입니다.

기본값

5

keepAliveTimeSecs

설명

스레드 수가 코어보다 큰 경우 이는 컨택 및 응답 기록 데이터 수집을 위한 초과 유휴 스레드가 종료 전에 새 작업을 대기할 최대 시간입니다.

기본값

5

queueCapacity

설명

컨택 및 응답 기록 데이터 수집을 위한 스레드 풀에 사용되는 대기열의 크기입니다.

기본값

1000

termWaitSecs

설명

런타임 서버 종료 시 이는 서비스 스레드가 컨택 및 응답 기록 데이터 수집을 완료할 때까지 대기할 시간(초)입니다.

기본값

5

Interact | 서비스 | threadManagement | allOtherServices

이 범주의 구성 등록 정보는 오퍼 자격 통계, 이벤트 활동 통계, 기본 문자열 사용 통계, 테이블 데이터에 대한 사용자 정의 로그를 수집하는 서비스의 스레드 관리 설정을 정의합니다.

corePoolSize

설명

오퍼 자격 통계, 이벤트 활동 통계, 기본 문자열 사용 통계, 테이블 데이터에 대한 사용자 정의 로그를 수집하는 서비스의, 풀에 보관할 스레드 수입니다(유휴 상태도 포함).

기본값

5

maxPoolSize

설명

오퍼 자격 통계, 이벤트 활동 통계, 기본 문자열 사용 통계, 테이블 데이터에 대한 사용자 정의 로그를 수집하는 서비스의, 풀에 보관할 최대 스레드 수입니다.

기본값

5

keepAliveTimeSecs

설명

스레드 수가 코어보다 큰 경우 이는 오퍼 자격 통계, 이벤트 활동 통계, 기본 문자열 사용 통계, 테이블 데이터에 대한 사용자 정의 로그를 수집하는 서비스의 초과된 유휴 스레드가 종료 전에 새 작업을 대기하는 최대 시간입니다.

기본값

5

queueCapacity

설명

오퍼 자격 통계, 이벤트 활동 통계, 기본 문자열 사용 통계, 테이블 데이터에 대한 사용자 정의 로그를 수집하는 서비스의 스레드 풀에 사용되는 대기열의 크기입니다.

기본값

1000

termWaitSecs

설명

런타임 서버 종료 시 이는 오퍼 자격 통계, 이벤트 활동 통계, 기본 문자열 사용 통계, 테이블 데이터에 대한 사용자 정의 로그를 수집하는 서비스의 서비스 스레드가 완료될 때까지 대기하는 시간(초)입니다.

기본값

5

Interact | 서비스 | threadManagement | flushCacheToDB

이 범주의 구성 등록 정보는 캐시의 수집된 데이터를 런타임 환경 데이터베이스 테이블에 쓰는 스레드의 스레드 관리 설정을 정의합니다.

corePoolSize

설명

캐시된 데이터를 데이터 저장소에 쓰는 예약된 스레드의 풀에 보관할 스레드 수입니다.

기본값

5

maxPoolSize

설명

캐시된 데이터를 데이터 저장소에 쓰는 예약된 스레드의 풀에 보관할 스레드의 최대 수입니다.

기본값

5

keepAliveTimeSecs

설명

스레드 수가 코어보다 큰 경우 이는 캐시된 데이터를 데이터 저장소에 쓰는 예약된 스레드의 초과 유휴 스레드가 종료 전 새 작업을 대기하는 최대 시간입니다.

기본값

5

queueCapacity

설명

캐시된 데이터를 데이터 저장소에 쓰는 예약된 스레드의 스레드 풀에 사용되는 대기열의 크기입니다.

기본값

1000

termWaitSecs

설명

런타임 서버 종료 시 이는 캐시된 데이터를 데이터 저장소에 쓰는 예약된 스레드의 서비스 스레드가 완료될 때까지 대기할 시간(초)입니다.

기본값

5

Interact | sessionManagement

이 구성 등록 정보 세트는 런타임 세션에 대한 설정을 정의합니다.

cacheType

설명

런타임 서버의 캐시 접근 유형을 정의합니다.

기본값

지역

올바른 값

배포 | 지역

maxNumberOfSessions

설명

어느 한 시점에 캐시에 보유되는 런타임 세션의 최대 수입니다. 캐시가 이 최대치에 도달했을 때 새 런타임 세션 추가 요청이 발생하면 캐시는 가장 오래된 비활성 런타임 세션을 제거합니다.

기본값

999999999

올바른 값

0보다 큰 정수

multicastIPAddress

설명

cacheType이 배포이면 배포된 캐시에 사용된 IP 주소를 입력하십시오. multicastPort도 정의해야 합니다.

cacheType이 지역인 경우에는 multicastIPAddress를 정의되지 않은 채로 둘 수 있습니다.

기본값

230.0.0.1

올바른 값

올바른 IP 주소

multicastPort

설명

cacheType이 배포이면 배포된 캐시에 사용된 포트 번호를 입력하십시오. multicastIPAddress도 정의해야 합니다.

cacheType이 지역인 경우에는 multicastPort를 정의되지 않은 채로 둘 수 있습니다.

기본값

6363

올바른 값

1024 – 49151

sessionTimeoutInSecs

설명

세션이 비활성 상태를 유지할 수 있는 시간(초)입니다. sessionTimeout 시간(초)이 경과하면 Interact는 세션을 종료합니다.

기본값

300

올바른 값

0보다 큰 정수

부록 C. Interact 디자인 환경 구성 등록 정보

이 섹션에서는 Interact 디자인 환경에 대한 모든 구성 등록 정보를 설명합니다.

Campaign | partitions | partition[n] | reports

이 구성 등록 정보는 보고서의 폴더를 정의합니다.

offerAnalysisTabCachedFolder

설명

offerAnalysisTabCachedFolder 등록 정보는 탐색 분할창에서 분석 링크를 클릭하면 표시되는 분석 탭에 나열되어 있는 버스트된(확장된) 오퍼 보고서의 스펙을 포함한 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']/folder[@name='cached']
```

segmentAnalysisTabOnDemandFolder

설명

segmentAnalysisTabOnDemandFolder 등록 정보는 세그먼트의 분석 탭에 나열된 세그먼트를 포함한 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']/folder[@name='cached']
```

offerAnalysisTabOnDemandFolder

설명

offerAnalysisTabOnDemandFolder 등록 정보는 오퍼의 분석 탭에 나열된 오퍼 보고서를 포함한 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='offer']
```

© Copyright IBM Corp. 2001, 2012

segmentAnalysisTabCachedFolder

설명

segmentAnalysisTabCachedFolder 등록 정보는 탐색 분할창에서 분석 링크를 클릭하면 표시되는 분석 탭에 나열되어 있는 버스트된(확장된) 세그먼트 보고서의 스펙을 포함한 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='segment']
```

analysisSectionFolder

설명

analysisSectionFolder 등록 정보는 보고서 사양이 저장된 루트 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

```
/content/folder[@name='Affinium Campaign']
```

campaignAnalysisTabOnDemandFolder

설명

campaignAnalysisTabOnDemandFolder 등록 정보는 캠페인의 분석 탭에 나열된 캠페인 보고서를 포함한 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']
```

campaignAnalysisTabCachedFolder

설명

campaignAnalysisTabCachedFolder 등록 정보는 탐색 분할창에서 분석 링크를 클릭하면 표시되는 분석 탭에 나열되어 있는 버스트된(확장된) 캠페인 보고서의 스펙을 포함한 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

```
/content/folder[@name='Affinium Campaign - Object Specific
Reports']/folder[@name='campaign']/folder[@name='cached']
```

campaignAnalysisTabEmessageOnDemandFolder

설명

campaignAnalysisTabEmessageOnDemandFolder 등록 정보는 캠페인의 분석 탭에 나열된 eMessage 보고서를 포함한 폴더의 위치를 지정합니다. XPath 표기법을 사용하여 경로가 지정됩니다.

기본값

/content/folder[@name='Affinium Campaign']/folder[@name='eMessage Reports']

campaignAnalysisTabInteractOnDemandFolder

설명

Interact 보고서의 보고서 서버 폴더 문자열입니다.

기본값

/content/folder[@name='Affinium Campaign']/folder[@name='Interact Reports']

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

interactiveChannelAnalysisTabOnDemandFolder

설명

대화식 채널 분석 탭 보고서의 보고서 서버 폴더 문자열입니다.

기본값

/content/folder[@name='Affinium Campaign - Object Specific Reports']/folder[@name='interactive channel']

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking

이 구성 등록 정보는 Interact 컨택 및 응답 기록 모듈의 설정을 정의합니다.

isEnabled

설명

예로 설정하면 Interact 런타임의 스테이징 테이블에서 Campaign 컨택 및 응답 기록 테이블로 Interact 컨택 및 응답 기록을 복사하는 Interact 컨택 및 응답 기록 모듈이 설정됩니다. interactInstalled 등록 정보도 예로 설정해야 합니다.

기본값

아니요

올바른 값

예 | 아니요

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

runOnceADay

설명

컨택 및 응답 기록 ETL을 하루에 한 번 실행할지 여부를 지정합니다. 이 등록 정보를 예로 설정하면 preferredStartTime 및 preferredEndTime에 지정되어 있는 예약된 간격 동안 ETL이 실행됩니다.

ETL을 실행하는 데 24시간이 넘게 소요되어 다음 날의 시작 시간을 놓친 경우에는 해당 일을 건너뛰고 그 다음 날 예약된 시간에 실행됩니다. 예를 들어, ETL이 1AM - 3AM 사이에 실행하도록 구성되어 있고 프로세스가 월요일 1AM에 시작되어 화요일 2AM에 완료되면, 원래 화요일 1AM으로 예약된 다음 실행을 건너뛰고 수요일 1AM에 다음 ETL이 시작됩니다.

ETL 예약은 일광 절약 시간 변경을 설명하지 않습니다. 예를 들어, ETL이 1AM - 3AM 사이에 실행하도록 예약된 경우 DST 변경이 발생하면 12AM 또는 2AM에 실행될 수 있습니다.

기본값

아니요

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

processSleepIntervalInMinutes

설명

Interact 런타임 스테이징 테이블에서 Campaign 컨택 및 응답 기록 테이블로 데이터를 복사하는 동안 Interact 컨택 및 응답 기록 모듈이 대기하는 시간(분)입니다.

기본값

60

올바른 값

0보다 큰 정수

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

preferredStartTime

설명

일일 ETL 프로세스를 시작할 선호하는 시간입니다. 이 등록 정보는 preferredEndTime 등록 정보와 함께 사용되어 ETL을 실행할 선호하는 시간 간격을 설정합니다. ETL이 지정된 시간 간격 중에 시작되며 maxJDBCFetchBatchSize를 사용하여 지정한 최대 수의 레코드를 처리합니다. 형식은 12시간 시계를 사용한 HH:mm:ss AM 또는 PM입니다.

기본값

12:00:00 AM

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

preferredEndTime

설명

일일 ETL 프로세스를 완료할 선호하는 시간입니다. 이 등록 정보는 preferredStartTime 등록 정보와 함께 사용되어 ETL을 실행할 선호하는 시간 간격을 설정합니다. ETL이 지정된 시간 간격 중에 시작되며 maxJDBCFetchBatchSize를 사용하여 지정한 최대 수의 레코드를 처리합니다. 형식은 12시간 시계를 사용한 HH:mm:ss AM 또는 PM입니다.

기본값

2:00:00 AM

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

purgeOrphanResponseThresholdInMinutes

설명

해당 컨택이 없는 응답이 제거될 때까지 Interact 컨택 및 응답 기록 모듈이 대기하는 시간(분)입니다. 컨택 로깅 없이 응답을 로깅하지 않도록 합니다.

기본값

180

올바른 값

0보다 큰 정수

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

maxJDBCInsertBatchSize

설명

쿼리를 커밋하기 전 JDBC 일괄처리의 최대 레코드 수입니다. 이 수는 Interact 응답 및 기록 모듈이 한 반복에서 처리하는 최대 레코드 수가 아닙니다. 각 반복 중 Interact 컨택 및 응답 기록 모듈은 스테이징 테이블에서 사용 가능한 모든 레코드를 처리합니다. 하지만 이 모든 레코드는 maxJDBCInsertSize 청크로 분할됩니다.

기본값

1000

올바른 값

0보다 큰 정수

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

maxJDBCFetchBatchSize

설명

스테이징 데이터베이스에서 페치할 JDBC 일괄처리의 최대 레코드 수입니다. 컨택 및 응답 기록 모듈의 성능 조정을 위해 이 값을 늘려야 할 수 있습니다.

예를 들어, 250만 컨택 기록 레코드를 하루에 처리하려면 maxJDBCFetchBatchSize를 2.5M보다 큰 수로 설정해야 하루의 모든 레코드가 처리됩니다.

그런 다음 maxJDBCFetchChunkSize 및 maxJDBCInsertBatchSize를 보다 작은 값으로 설정할 수 있습니다(이 예제에서는 각각 50,000 및 10,000으로). 다음 날의 일부 레코드도 처리될 수 있지만 다음 날까지 보유됩니다.

기본값

1000

올바른 값

0보다 큰 정수

maxJDBCFetchChunkSize

설명

ETL(추출, 변환, 로드) 중 읽은 데이터 JDBC 청크 *크기*의 최대 수입니다. 일부 경우 삽입 *크기*보다 큰 청크 *크기*가 ETL 프로세스의 속도를 개선할 수 있습니다.

기본값

1000

올바른 값

0보다 큰 정수

deleteProcessedRecords

설명

컨택 기록 및 응답 기록 레코드를 처리한 후 보유할지 여부를 지정합니다.

기본값

예

completionNotificationScript

설명

ETL이 완료될 때 실행할 스크립트의 절대 경로를 지정합니다. 스크립트를 지정하면 네 개의 인수 즉, 시작 시간, 종료 시간, 처리한 총 CH 레코드 수, 처리한 총 RH 레코드 수가 완료 공지 스크립트로 전달됩니다. 시작 시간과 종료 시간은 1970 이후의 경과된 밀리초 수를 나타내는 숫자 값입니다.

기본값

없음

fetchSize

설명

스테이징 테이블에서 레코드를 검색할 때 JDBC fetchSize를 설정하도록 합니다.

특히 Oracle 데이터베이스에서는 JDBC가 각 네트워크 라운드트립마다 검색해야 하는 레코드 수로 설정을 조정하십시오. 100K가 넘는 대규모 일괄처리의 경우에는 10000을 시도해보십시오. 여기에 너무 큰 값을 사용하면 메모리 사용량에 영향을 미치고 얻는 이득이 사소하므로 크게 문제가 되지 않는다면 너무 큰 값을 사용하지 않도록 주의하십시오.

기본값

없음

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | runtimeDataSources | [runtimeDataSource]

이 구성 등록 정보는 Interact 컨택 및 응답 기록 모듈의 데이터 소스를 정의합니다.

jndiName

설명

systemTablesDataSource 등록 정보를 사용하여 Interact 런타임 테이블의 응용 프로그램 서버(Websphere 또는 WebLogic)에 정의된 JNDI(Java Naming and Directory Interface) 데이터 소스를 식별하십시오.

Interact 런타임 데이터베이스는 aci_runtime 및 aci_populate_runtime dll 스크립트로 채워진 데이터베이스이며 예를 들어, 여러 테이블 중 다음 UACI_CHOfferAttrib 및 UACI_DefaultedStat 테이블을 포함합니다.

기본값

정의된 기본값이 없습니다.

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

databaseType

설명

Interact 런타임 데이터 소스의 데이터베이스 유형입니다.

기본값

SQLServer

올바른 값

SQLServer | Oracle | DB2

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

schemaName

설명

컨택 및 응답 기록 모듈 스테이징 테이블을 포함한 스키마의 이름입니다. 이 이름은 런타임 환경 테이블의 이름과 동일해야 합니다.

스키마를 정의할 필요는 없습니다.

기본값

정의된 기본값이 없습니다.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | contactTypeMappings

이 구성 등록 정보는 보고 또는 학습을 위해 '컨택'으로 맵핑하는 캠페인의 컨택 유형을 정의합니다.

contacted

설명

오퍼 컨택의 Campaign 시스템 테이블에서 UA_DtlContactHist 테이블의 ContactStatusID 열에 할당된 값입니다. 이 값은 UA_ContactStatus 테이블의 올바른 항목이어야 합니다. 컨택 유형 추가에 대한 세부 정보는 *Campaign 관리자 가이드*의 내용을 참조하십시오.

기본값

2

올바른 값

0보다 큰 정수

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | contactAndResponseHistTracking | responseTypeMappings

이 구성 등록 정보는 보고 및 학습에 대한 수락 또는 거부의 응답을 정의합니다.

accept

설명

수락된 오퍼의 Campaign 시스템 테이블에 있는 UA_ResponseHistory 테이블의 ResponseTypeID 열에 할당된 값입니다. 이 값은 UA_UsrResponseType 테이블의 올바른 항목이어야 합니다. CountsAsResponse 열에 1, 응답을 할당해야 합니다.

응답 유형 추가에 대한 세부 정보는 *Campaign 관리자 가이드*의 내용을 참조하십시오.

기본값

3

올바른 값

0보다 큰 정수

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

reject

설명

거부된 오퍼의 Campaign 시스템 테이블에 있는 UA_ResponseHistory 테이블의 ResponseTypeID 열에 할당된 값입니다. 이 값은 UA_UsrResponseType 테이블의 올바른 항목이어야 합니다. CountsAsResponse 열에 2, 거부를 할당해야 합니다. 응답 유형 추가에 대한 세부 정보는 *Campaign 관리자 가이드*의 내용을 참조하십시오.

기본값

8

올바른 값

0보다 큰 정수

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | report

이 구성 등록 정보는 Cognos와 상호작용할 때 보고서 이름을 정의합니다.

interactiveCellPerformanceByOfferReportName

설명

오퍼 보고서의 대화식 셀 성과의 이름입니다. 이 이름은 Cognos 서버의 이 보고서 이름과 일치해야 합니다.

기본값

오퍼별 대화식 셀 성과

treatmentRuleInventoryReportName

설명

처리 규칙 인벤토리 보고서의 이름입니다. 이 이름은 Cognos 서버의 이 보고서 이름과 일치해야 합니다.

기본값

채널 처리 규칙 인벤토리

deploymentHistoryReportName

설명

배포 기록 보고서란 보고서의 이름입니다. 이 이름은 Cognos 서버의 이 보고서 이름과 일치해야 합니다.

기본값

채널 배포 기록

Campaign | partitions | partition[n] | Interact | learning

이 구성 등록 정보는 기본 제공 학습 모듈을 조정합니다.

confidenceLevel

설명

탐색에서 이용으로 전환하기 전 학습 유틸리티를 얼마나 필요로 하는지 정도를 나타내는 비율입니다. 0 값은 사실상 탐색을 종료합니다.

이 등록 정보는 Interact 런타임의 `Interact > offerserving > optimizationType` 등록 정보가 `BuiltInLearning`으로 설정된 경우에만 적용할 수 있습니다.

기본값

95

올바른 값

5로 나눌 수 있는 0과 95 사이의 정수 또는 99

enableLearning

설명

예로 설정하면 Interact 디자인 시간에 학습을 사용할 것으로 예상합니다. `enableLearning`을 예로 설정하는 경우 `Interact > offerserving > optimizationType`을 `BuiltInLearning` 또는 `ExternalLearning`으로 구성해야 합니다.

아니요로 설정하면 Interact 디자인 시간에 학습을 사용하지 않을 것으로 예상합니다. `enableLearning`을 아니요로 설정하는 경우에는 `Interact > offerserving > optimizationType`을 `NoLearning`으로 구성해야 합니다.

기본값

아니요

maxAttributeNames

설명

Interact 학습 유틸리티가 모니터하는 학습 속성의 최대 수입니다.

이 등록 정보는 Interact 런타임의 Interact > offerserving > optimizationType 등록 정보가 BuiltInLearning으로 설정된 경우에만 적용할 수 있습니다.

기본값

10

올바른 값

임의의 정수

maxAttributeValues

설명

Interact 학습 모듈이 각 학습 속성에 대해 추적하는 값의 최대치입니다.

이 등록 정보는 Interact 런타임의 Interact > offerserving > optimizationType 등록 정보가 BuiltInLearning으로 설정된 경우에만 적용할 수 있습니다.

기본값

5

otherAttributeValue

설명

maxAttributeValues를 벗어난 모든 속성 값을 표시하는 데 사용된 속성 값의 기본 이름입니다.

이 등록 정보는 Interact 런타임의 Interact > offerserving > optimizationType 등록 정보가 BuiltInLearning으로 설정된 경우에만 적용할 수 있습니다.

기본값

기타

올바른 값

문자열 또는 숫자

예제

maxAttributeValues가 3으로 설정되고 otherAttributeValue가 기타로 설정되면 학습 모듈은 처음 세 개의 값을 추적합니다. 다른 모든 값은 기타 범주에 할당

됩니다. 예를 들어, 방문자 속성 모발 색상을 추적 중이며 처음 다섯 명 방문자의 모발 색상이 검은색, 갈색, 금발, 빨간색, 회색이면 학습 유틸리티는 검은색, 갈색, 금발의 모발 색상을 추적합니다. 빨간색과 회색 색상은 otherAttributeValue, 기타로 그룹화됩니다.

percentRandomSelection

설명

학습 모듈이 무작위 오퍼를 제시하는 시간의 백분율입니다. 예를 들어, percentRandomSelection을 5로 설정하면 학습 모듈이 시간의 5%(전체 100 권장사항 중 5)에만 무작위 오퍼를 제시함을 의미합니다.

기본값

5

올바른 값

0 - 100의 정수

recencyWeightingFactor

설명

recencyWeightingPeriod로 정의된 데이터 세트의 10진수 백분율 표시입니다. 예를 들어, 기본값 .15는 학습 유틸리티에 사용되는 데이터의 15%가 recencyWeightingPeriod에서 나왔음을 의미합니다.

이 등록 정보는 Interact 런타임의 Interact > offerserving > optimizationType 등록 정보가 BuiltInLearning으로 설정된 경우에만 적용할 수 있습니다.

기본값

0.15

올바른 값

1 미만의 10진수 값

recencyWeightingPeriod

설명

학습 모듈이 recencyWeightingFactor 비율의 가중치를 부여한 데이터의 시간 크기입니다. 예를 들어, 기본값 120은 학습 모듈에 사용된 데이터의 recencyWeightingFactor가 지난 120시간에 나온 것임을 의미합니다.

이 등록 정보는 optimizationType이 builtInLearning으로 설정된 경우에만 적용할 수 있습니다.

기본값

120

minPresentCountThreshold

설명

데이터가 계산에 사용되고 학습 모듈이 탐색에 들어서기 전에 오퍼를 제시해야 하는 최소 횟수입니다.

기본값

0

올바른 값

0 이상의 정수

enablePruning

설명

예로 설정하면 Interact 학습 모듈이 학습 속성(표준 또는 동적)이 예측되지 않는 시기를 알고리즘 방식으로 판별합니다. 학습 속성이 예측되지 않으면 학습 모듈은 오퍼의 가중치를 판별할 때 이 속성을 고려하지 않습니다. 이는 학습 모듈이 학습 데이터를 집계할 때까지 계속됩니다.

아니요로 설정하는 경우에는 학습 모듈이 항상 모든 학습 속성을 사용합니다. 예측되지 않는 속성을 정리하지 않으면 학습 모듈이 최대한으로 정확해질 수 없습니다.

기본값

예

올바른 값

예 | 아니요

Campaign | partitions | partition[n] | Interact | learning | learningAttributes | [learningAttribute]

이 구성 등록 정보는 학습 속성을 정의합니다.

attributeName

설명

각 attributeName은 학습 모듈을 모니터하려는 방문자 속성의 이름입니다. 세션 데이터의 이름-값 쌍의 이름과 일치해야 합니다.

이 등록 정보는 Interact 런타임의 Interact > offerserving > optimizationType 등록 정보가 BuiltInLearning으로 설정된 경우에만 적용할 수 있습니다.

기본값

정의된 기본값이 없습니다.

Campaign | partitions | partition[n] | Interact | deployment

이 등록 정보는 배포 설정을 정의합니다.

chunkSize

설명

각 Interact 배포 패키지의 최대 단편화 크기(KB)입니다.

기본값

500

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup]

이 등록 정보는 서버 그룹 설정을 정의합니다.

serverGroupName

설명

Interact 런타임 서버 그룹의 이름입니다. 대화식 채널 요약 탭에 표시되는 이름입니다.

기본값

정의된 기본값이 없습니다.

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | serverGroups | [serverGroup] | instanceURLs | [instanceURL]

이 구성 등록 정보는 Interact 런타임 서버를 정의합니다.

instanceURL

설명

Interact 런타임 서버의 URL입니다. 서버 그룹은 여러 Interact 런타임 서버를 포함할 수 있지만 각 서버를 새 범주 아래에 생성해야 합니다.

기본값

정의된 기본값이 없습니다.

예제

`http://server:port/interact`

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | flowchart

이 구성 등록 정보는 대화식 플로차트의 테스트 실행에 사용되는 Interact 런타임 환경을 정의합니다.

serverGroup

설명

테스트 실행을 실행할 때 Campaign에 사용되는 Interact 서버 그룹의 이름입니다. 이 이름은 `serverGroups` 아래에 생성하는 범주 이름과 일치해야 합니다.

기본값

정의된 기본값이 없습니다.

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

dataSource

설명

대화식 플로차트의 테스트 실행을 수행할 때 사용할 Campaign의 물리적 데이터 소스를 식별하려면 `dataSource` 등록 정보를 사용하십시오. 이 등록 정보는 Interact 디자인 시간에 대해 정의된 테스트 실행 데이터 소스의 `Campaign > partitions > partitionN > dataSources` 등록 정보에 정의된 데이터 소스와 일치해야 합니다.

기본값

정의된 기본값이 없습니다.

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | DefaultOffers

이 구성 등록 정보는 기본 오퍼 테이블의 기본 셀 코드를 정의합니다. 전역 오퍼 지정을 정의하고 있는 경우에만 이 등록 정보를 구성해야 합니다.

DefaultCellCode

설명

기본 오퍼 테이블에 셀 코드를 정의하지 않은 경우 Interact에 사용되는 기본 셀 코드입니다.

기본값

정의된 기본값이 없습니다.

올바른 값

Campaign에 정의된 셀 코드 형식에 일치하는 문자열

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | offersBySQL

이 구성 등록 정보는 offersBySQL 테이블의 기본 셀 코드를 정의합니다. SQL 쿼리를 사용하여 원하는 후보 오퍼 세트를 얻는 경우에만 이 등록 정보를 구성해야 합니다.

DefaultCellCode

설명

기본 셀 코드 Interact는 셀 코드 열에 널값이 있는(또는 셀 코드 열이 전부 누락된 경우) OffersBySQL 테이블의 모든 오퍼에 사용됩니다. 이 값은 올바른 셀 코드여야 합니다.

기본값

정의된 기본값이 없습니다.

올바른 값

Campaign에 정의된 셀 코드 형식에 일치하는 문자열

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | Interact | whiteList | [AudienceLevel] | ScoreOverride

이 구성 등록 정보는 점수 재정의 테이블의 기본 셀 코드를 정의합니다. 개별 오퍼 지정을 정의하고 있는 경우에만 이 등록 정보를 구성해야 합니다.

DefaultCellCode

설명

점수 재정의 테이블에 셀 코드를 정의하지 않은 경우 Interact에 사용되는 기본 셀 코드입니다.

기본값

정의된 기본값이 없습니다.

올바른 값

Campaign에 정의된 셀 코드 형식에 일치하는 문자열

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

Campaign | partitions | partition[n] | server | internal

이 범주의 등록 정보는 선택된 Campaign 파티션의 통합 설정과 internalID 제한사항을 지정합니다. Campaign 설치에 여러 파티션이 있는 경우 적용하려는 각 파티션에 대해 다음 등록 정보를 설정하십시오.

internalIdLowerLimit

설명

`internalIdUpperLimit` 및 `internalIdLowerLimit` 등록 정보는 Campaign 내부 ID를 지정된 범위 내로 제한합니다. Campaign에 하한 및 상한이 모두 사용될 수 있는 포함적 값임에 유의하십시오.

기본값

`0`(영)

internalIdUpperLimit

설명

`internalIdUpperLimit` 및 `internalIdLowerLimit` 등록 정보는 Campaign 내부 ID를 지정된 범위 내로 제한합니다. Campaign에 하한 및 상한이 모두 사용될 수 있는 포함적 값임에 유의하십시오.

기본값

4294967295

eMessageInstalled

설명

eMessage가 설치됨을 표시합니다. 예를 선택하면 eMessage 기능이 Campaign 인터페이스에서 사용 가능합니다.

IBM 설치 프로그램은 eMessage 설치의 기본 파티션의 경우 이 등록 정보를 예로 설정합니다. eMessage를 설치한 추가 파티션에는 이 등록 정보를 수동으로 구성해야 합니다.

기본값

아니요

올바른 값

예 | 아니요

interactInstalled

설명

Interact 디자인 환경을 설치한 후 Campaign에서 Interact 디자인 환경을 사용하려면 이 구성 등록 정보를 예로 설정해야 합니다.

Interact를 설치하지 않았으면 아니요로 설정하십시오. 이 등록 정보를 아니요로 설정해도 Interact 메뉴와 옵션은 사용자 인터페이스에서 제거되지 않습니다. 메뉴와 옵션을 제거하려면 `configTool` 유틸리티를 사용하여 Interact를 수동으로 등록 해제해야 합니다.

기본값

아니요

올바른 값

예 | 아니요

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

MO_UC_integration

설명

이 파티션에 대해 Marketing Operations와의 통합을 설정합니다. 다음 세 가지 중 어느 옵션을 예로 설정할 계획이면 **MO_UC_integration**을 예로 설정

해야 합니다. 이 통합 구성에 대한 자세한 정보는 *IBM Unica Marketing Operations* 및 *Campaign* 통합 가이드의 내용을 참조하십시오.

기본값

아니요

올바른 값

예 | 아니요

MO_UC_BottomUpTargetCells

설명

이 파티션의 대상 셀 스프레드시트에 대한 상향식 셀을 허용합니다. 예로 설정하면 상향식과 하향식 대상 셀이 모두 표시되지만 상향식 대상 셀은 읽기 전용입니다. **MO_UC_integration**을 사용 가능하게 설정해야 함에 유의하십시오. 이 통합 구성에 대한 자세한 정보는 *IBM Unica Marketing Operations* 및 *Campaign* 통합 가이드의 내용을 참조하십시오.

기본값

아니요

올바른 값

예 | 아니요

Legacy_campaigns

설명

MO_UC_integration 등록 정보가 **예**로 설정되면 **Legacy_campaigns** 등록 정보는 Campaign 7.x에 생성된 캠페인과 Plan 7.x 프로젝트에 링크된 캠페인을 포함하여, 통합을 사용하기 전 생성된 캠페인에 대한 액세스를 사용 가능하게 합니다. 이 통합 구성에 대한 자세한 정보는 *IBM Unica Marketing Operations* 및 *Campaign* 통합 가이드의 내용을 참조하십시오.

기본값

아니요

올바른 값

예 | 아니요

IBM Unica® Marketing Operations - Offer 통합

설명

Marketing Operations를 사용하여 이 파티션에서 오퍼 수명 주기 관리 작업을 수행하는 기능을 사용 가능하게 합니다. (**MO_UC_integration**을 사용 가능하게 해야 합니다. 또한 **설정 > 구성 > Unica > 플랫폼**에서 **캠페인 통합**을

사용 가능하게 해야 합니다.) 이 통합 구성에 대한 자세한 정보는 *IBM Unica Marketing Operations* 및 *Campaign* 통합 가이드의 내용을 참조하십시오.

기본값

아니요

올바른 값

예 | 아니요

UC_CM_integration

설명

캠페인 파티션의 IBM Coremetrics® 온라인 세그먼트 통합을 사용 가능하게 합니다. 이 옵션을 예로 설정하면 플로차트의 프로세스 상자 선택에 **IBM Coremetrics 세그먼트**를 입력으로 선택하는 옵션이 제공됩니다. 각 파티션의 통합을 구성하려면 **설정 > 구성 > Campaign | 파티션 | partition[n] | Coremetrics**를 선택하십시오.

기본값

아니요

올바른 값

예 | 아니요

Campaign | 모니터링

이 범주의 등록 정보는 조작 모니터링 기능이 사용되는지 여부, 조작 모니터링 서버의 URL, 캐싱 작동을 지정합니다. 조작 모니터링은 활성 플로차트를 표시하고 이를 컨트롤하도록 합니다.

cacheCleanupInterval

설명

cacheCleanupInterval 등록 정보는 플로차트 상태 캐시의 자동 정리 간격(초)을 지정합니다.

Campaign 7.0 이전 버전에서는 이 등록 정보를 사용할 수 없습니다.

기본값

600(10분)

cacheRunCompleteTime

설명

cacheRunCompleteTime 등록 정보는 완료된 실행이 캐시되고 모니터링 페이지에 표시되는 시간(분)을 지정합니다.

Campaign 7.0 이전 버전에서는 이 등록 정보를 사용할 수 없습니다.

기본값

4320

monitorEnabled

설명

monitorEnabled 등록 정보는 모니터가 켜져 있는지 여부를 지정합니다.

Campaign 7.0 이전 버전에서는 이 등록 정보를 사용할 수 없습니다.

기본값

예

serverURL

설명

Campaign > 모니터링 > serverURL 등록 정보는 조작 모니터링 서버의 URL을 지정합니다. 필수 설정입니다. 조작 모니터링 서버 URL이 기본값이 아닌 경우 값을 수정하십시오.

Campaign이 SSL(Secure Sockets Layer) 통신을 사용하도록 구성되어 있으면 이 등록 정보의 값을 HTTPS를 사용하도록 설정하십시오. 예: serverURL=https://*host:SSL_port*/Campaign/OperationMonitor 여기서,

- *host*는 웹 응용 프로그램이 설치된 시스템의 이름 또는 IP 주소입니다.
- *SSL_Port*는 웹 응용 프로그램의 SSL 포트입니다.

URL의 https에 유의하십시오.

기본값

http://localhost:7001/Campaign/OperationMonitor

monitorEnabledForInteract

설명

예로 설정하면 Campaign JMX 커넥터 서버가 Interact에 사용됩니다. Campaign에는 JMX 보안이 없습니다.

아니요로 설정하는 경우에는 Campaign JMX 커넥터 서버에 연결할 수 없습니다.

이 JMX 모니터링은 Interact 컨택 및 응답 기록 모듈에만 사용됩니다.

기본값

False

올바른 값

True | False

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

protocol

설명

monitorEnabledForInteract가 예로 설정된 경우 Campaign JMX 커넥터 서버의 청취 프로토콜입니다.

이 JMX 모니터링은 Interact 컨택 및 응답 기록 모듈에만 사용됩니다.

기본값

JMXMP

올바른 값

JMXMP | RMI

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

port

설명

monitorEnabledForInteract가 예로 설정된 경우 Campaign JMX 커넥터 서버의 청취 포트입니다.

이 JMX 모니터링은 Interact 컨택 및 응답 기록 모듈에만 사용됩니다.

기본값

2004

올바른 값

1025 - 65535의 정수

가용성

이 등록 정보는 Interact를 설치한 경우에만 적용할 수 있습니다.

부록 D. 클라이언트 측에서 실시간 오퍼 사용자 개인 설정

Interact 서버에 대한 SOAP 호출 또는 하위 수준 Java 코드를 구현하지 않고 실시간 오퍼 사용자 개인 설정을 제공할 경우가 있습니다. 예를 들어, 방문자가 Javascript 컨텐츠가 유일하게 사용 가능한 확장 프로그래밍인 웹 페이지를 처음에 로드하거나 방문자가 HTML 컨텐츠만 가능한 전자 메일 메시지를 여는 경우입니다. IBM Unica Interact는 클라이언트 측에 로드된 웹 컨텐츠만 제어하거나 Interact에 인터페이스를 단순화하려는 상황에서 실시간 오퍼 관리를 제공하는 몇 개의 커넥터를 제공합니다.

Interact 설치는 클라이언트 측에서 시작된 오퍼 사용자 개인 설정을 위한 다음 두 개의 커넥터를 포함합니다.

- 『Interact Message Connector 정보』. Message Connector를 사용하면, 전자 메일 메시지 또는 기타 전자 미디어의 웹 컨텐츠가 페이지 로드 오퍼 프리젠테이션 및 다른 사이트로 연결 랜딩 페이지에 대해 Interact 서버를 호출하는 이미지 및 링크 태그를 포함할 수 있습니다.
- 245 페이지의 『Interact Web Connector 정보』. Web Connector(JS Connector라고도 함)를 사용하면, 웹 페이지가 클라이언트 측 JavaScript를 사용하여 페이지 로드 오퍼 프리젠테이션 및 다른 사이트로 연결 랜딩 페이지를 통해 오퍼 중재, 프리젠테이션, 컨택/응답 기록을 관리할 수 있습니다.

Interact Message Connector 정보

Interact Message Connector는 전자 메일 메시지 및 기타 전자 미디어가 IBM Unica Interact를 호출하여 열 때 그리고 고객이 지정된 사이트로 연결하는 메시지를 클릭할 때 사용자 개인 설정된 오퍼를 제공할 수 있게 합니다. 이는 두 개의 주요 태그를 사용하여 수행되는데, 열 때 사용자 개인 설정된 오퍼를 로드하는 이미지 태그(IMG)와 다른 사이트로 연결에 대한 정보를 캡처하고 고객을 특정 랜딩 페이지로 경로 재지정하는 링크 태그(A)입니다.

예제

다음 예제는 IMG 태그 URL(문서가 Interact 서버에 열리면 정보를 전달하고 응답하여 해당 오퍼 이미지를 검색)과 A 태그 URL(다른 사이트로 연결 시 Interact 서버로 전달되는 정보를 판별)을 모두 포함하는 마케팅 지점(예: 전자 메일 메시지 내)에 포함시킬 수 있는 일부 HTML 코드를 표시합니다.

```
<a  href="http://www.example.com/MessageConnector/
offerClickthru.jsp?msgId=1234&linkId=1&userid=1&referral=test"><img
```

© Copyright IBM Corp. 2001, 2012

```
src="http:// www.example.com/MessageConnector/offerImage.jsp?msgId=1234
&linkId=1&userid=1&incomeLevel=5&incomeType=numeric"/></a>
```

다음 예제에서 IMG 태그는 A 태그로 묶여 있으며 이로 인해 다음 동작이 발생합니다.

1. 전자 메일 메시지가 열리면, Message Connector가 IMG 태그로 인코딩된 정보(이 메시지의 msgID 및 linkID, userid, 소득 수준, 소득 유형을 포함하는 고객 매개변수)가 들어 있는 요청을 수신합니다.
2. 이 정보가 API 호출을 통해 Interact 런타임 서버로 전달됩니다.
3. 런타임 서버가 Message Connector에 오퍼를 반환합니다. Message Connector는 오퍼 이미지 URL을 검색하고 해당 URL(추가 매개변수 포함)을 제공하며 해당 오퍼 URL로 이미지 요청을 경로 재지정합니다.
4. 고객에게 오퍼가 이미지로 표시됩니다.

이 때 고객은 해당 이미지를 클릭하여 몇 가지 방법으로 오퍼에 응답할 수 있습니다. A 태그 및 지정된 HREF 속성(대상 URL을 지정)을 사용하여 다른 사이트로 연결하면 해당 오퍼의 URL에 연결된 랜딩 페이징 대한 다른 요청을 Message Connector로 보냅니다. 그러면 고객 브라우저가 오퍼에 구성된 랜딩 페이지로 경로 재지정됩니다.

다른 사이트로 연결 A 태그는 반드시 필수는 아닙니다. 오퍼는 이미지(예: 고객이 인쇄할 쿠폰)만으로 구성될 수 있습니다.

Message Connector 설치

Message Connector를 설치, 배포, 실행하는 데 필요한 파일은 IBM Unica Interact 런타임 서버 설치에 자동으로 포함되었습니다. 이 섹션에서는 Message Connector 사용 준비에 필요한 단계를 요약합니다.

Message Connector 설치 및 배포에 관련된 작업은 다음과 같습니다.

- 선택적으로, 『Message Connector 구성』에 설명된 대로 Message Connector의 기본 설정 구성
- 240 페이지의 『Message Connector 테이블 작성』에 설명된 대로 Message Connector 트랜잭션 데이터를 저장하는 데 필요한 데이터베이스 테이블 작성
- 242 페이지의 『Message Connector 배포 및 실행』에 설명된 대로 Message Connector 웹 응용 프로그램 설치
- 242 페이지의 『Message Connector 링크 작성』에 설명된 대로 열 때 그리고 다른 사이트로 연결할 때 Message Connector 오퍼를 호출하는 데 필요한 IMG 및 A 태그를 마케팅 지점(예: 전자 메일 또는 웹 페이지)에 작성

Message Connector 구성

Message Connector를 배포하기 전에 설치에 포함된 구성 파일을 특정 환경에 맞게 사용자 정의해야 합니다. Interact 런타임 서버의 Message Connector 디렉토리에 있는

MessageConnectorConfig.xml XML 파일(<Interact_home>/msgconnector/config/MessageConnectorConfig.xml과 유사)을 수정할 수 있습니다.

MessageConnectorConfig.xml 파일은 필요한 일부 구성 설정과 선택적인 일부 구성 설정을 포함합니다. 사용하는 설정을 특정 설치에 맞게 사용자 정의해야 합니다. 여기 설명된 단계를 따라 구성을 수정하십시오.

1. Message Connector가 웹 응용 프로그램 서버에 배포되어 실행 중인 경우, Message Connector를 배포 제거한 후 계속하십시오.
2. Interact 런타임 서버의 텍스트 또는 XML 편집기에서 MessageConnectorConfig.xml 파일을 여십시오.
3. 필요에 따라 구성 설정을 수정하고 다음 *필수* 설정이 설치에 올바른지 확인하십시오.
   - <interactUrl> - Message Connector 페이지 태그를 연결해야 하고 Message Connector가 실행 중인 Interact 런타임 서버 URL입니다.
   - <imageErrorLink> - 오퍼 이미지에 대한 요청을 처리하는 동안 오류가 발생하면 Message Connector가 경로 재지정할 URL입니다.
   - <landingPageErrorLink> - 오퍼 랜딩 이미지에 대한 요청을 처리하는 동안 오류가 발생하면 Message Connector가 경로 재지정할 URL입니다.
   - <audienceLevels> - 하나 이상의 대상 수준 설정 집합을 포함하고 Message Connector 링크에 아무것도 지정되지 않은 경우 기본 대상 수준을 지정하는 구성 파일 섹션입니다. 한 개 이상의 대상 수준이 구성되어 있어야 합니다.

   모든 구성 설정은 『Message Connector 구성 설정』에 보다 자세히 설명되어 있습니다.
4. 구성 변경을 완료했으면 MessageConnectorConfig.xml 파일을 저장 후 닫으십시오.
5. Message Connector 설정 및 배포를 계속하십시오.

Message Connector 구성 설정:

Message Connector를 구성하려면, Interact 런타임 서버의 Message Connector 디렉토리에 있는 MessageConnectorConfig.xml XML 파일(일반적으로 <Interact_home>/msgconnector/config/MessageConnectorConfig.xml)을 수정할 수 있습니다. 여기서는 이 XML 파일의 구성을 각각 설명합니다. Message Connector가 배포되어 실행된 후에 이 파일을 수정하는 경우, 파일 수정을 완료한 후 Message Connector를 배포 제거한 후 다시 배포하거나 응용 프로그램 서버를 다시 시작하여 이러한 설정을 다시 로드하십시오.

일반 설정

다음 테이블에는 MessageConnectorConfig.xml 파일의 generalSettings 섹션에 포함된 선택적 및 필수 설정 목록이 들어 있습니다.

표 20. Message Connector 일반 설정

요소	설명	기본값
<interactURL>	Message Connector 페이지 태그로부터의 호출을 처리할 Interact 런타임 서버(예: Message Connector가 실행 중인 런타임 서버) URL입니다. 이 요소는 필수입니다.	http://localhost:7001/interact
<defaultDateTimeFormat>	기본 날짜 형식입니다.	MM/dd/yyyy
<log4jConfigFileLocation>	Log4j 등록 정보 파일의 위치입니다. 설정된 경우 *$MESSAGE_CONNECTOR_HOME* 환경 변수에 상대적입니다. 그렇지 않으면, 이 값은 Message Connector 웹 응용 프로그램의 루트 경로에 상대적입니다.	config/MessageConnectorLog4j.properties

기본 매개변수 값

다음 테이블에는 MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 포함된 선택적 및 필수 설정 목록이 들어 있습니다.

표 21. Message Connector 기본 매개변수 설정

요소	설명	기본값
<interactiveChannel>	기본 대화식 채널 이름입니다.	
<interactionPoint>	기본 상호작용 지점 이름입니다.	
<debugFlag>	디버깅 활성화 여부를 판별합니다. 허용된 값은 true 및 false입니다.	false
<contactEventName>	게시된 컨택 이벤트의 기본 이름입니다.	
<acceptEventName>	게시된 수락 이벤트의 기본 이름입니다.	
<imageUrlAttribute>	Message Connector 링크에 아무것도 지정되지 않은 경우 오퍼 이미지 URL을 포함하는 기본 오퍼 속성입니다.	
<landingPageUrlAttribute>	Message Connector 링크에 아무것도 지정되지 않은 경우 다른 사이트로 연결 랜딩 페이지의 기본 URL입니다.	

동작 설정

다음 테이블에는 MessageConnectorConfig.xml 파일의 behaviorSettings 섹션에 포함된 선택적 및 필수 설정 목록이 들어 있습니다.

표 22. Message Connector 동작 설정

요소	설명	기본값
<imageErrorLink>	오퍼 이미지에 대한 요청을 처리하는 동안 오류가 발생하면 커넥터가 경로 재지정하는 URL입니다. 이 설정은 필수입니다.	/images/default.jpg
<landingPageErrorLink>	다른 사이트로 연결 랜딩 페이지에 대한 요청을 처리하는 동안 오류가 발생하면 커넥터가 경로 재지정하는 URL입니다. 이 설정은 필수입니다.	/jsp/default.jsp
<alwaysUseExistingOffer>	이미 만료된 경우에도 캐시된 오퍼를 반환해야 하는지 여부를 판별합니다. 허용된 값은 true 및 false입니다.	false
<offerExpireAction>	원래 오퍼를 찾았지만 이미 만료된 경우 수행할 작업입니다. 허용된 값은 다음과 같습니다. • GetNewOffer • RedirectToErrorPage • ReturnExpiredOffer	RedirectToErrorPage

스토리지 설정

다음 테이블에는 MessageConnectorConfig.xml 파일의 storageSettings 섹션에 포함된 선택적 및 필수 설정 목록이 들어 있습니다.

표 23. MessageConnector 스토리지 설정

요소	설명	기본값
<persistenceMode>	캐시가 데이터베이스에 새 항목을 지속하는 경우입니다. 허용된 값은 WRITE-BEHIND(데이터가 처음에 캐시에 기록되고 나중에 데이터베이스에 업데이트됨) 및 WRITE-THROUGH(데이터가 캐시에 기록되는 동시에 데이터베이스에 기록됨)입니다.	WRITE-THROUGH
<maxCacheSize>	메모리 캐시의 최대 항목 수입니다.	5000
<maxPersistenceBatchSize>	데이터베이스에 항목을 지속하는 동안 최대 일괄처리 크기입니다.	200
<macCachePersistInterval>	데이터베이스에 지속되지 전에 항목이 캐시에 유지되는 최대 시간(초)입니다.	3
<maxElementOnDisk>	디스크 캐시의 최대 항목 수입니다.	5000
<cacheEntryTimeToExpireInSeconds>	만료 전 디스크 캐시의 항목이 지속될 최대 시간입니다.	60000
<jdbcSettings>	JDBC 연결이 사용되는 경우 특정 정보가 들어 있는 XML 파일 섹션입니다. <dataSourceSettings> 섹션과 상호 배타적입니다.	기본적으로 로컬 서버에 구성된 SQLServer 데이터베이스에 연결하도록 구성되지만, 이 섹션을 설정하는 경우에는 로그인하려면 실제 JDBC 설정 및 신임 정보를 제공해야 합니다.
<dataSourceSettings>	데이터 소스 연결이 사용되는 경우 특정 정보가 들어 있는 XML 파일 섹션입니다. <jdbcSettings> 섹션과 상호 배타적입니다.	기본적으로 로컬 웹 응용 프로그램 서버에 정의된 InteractDS 데이터 소스에 연결하도록 구성됩니다.

대상 수준

다음 테이블에는 MessageConnectorConfig.xml 파일의 audienceLevels 섹션에 포함된 선택적 및 필수 설정 목록이 들어 있습니다.

audienceLevels 요소는 다음 예제에서와 같이 Message Connector 링크에 아무것도 지정되지 않은 경우 사용할 기본 대상 수준을 지정하는 데 선택적으로 사용됩니다.

```
<audienceLevels default="Customer">
```

이 예제에서 default 속성 값은 이 섹션에 정의된 audienceLevel 이름과 일치합니다. 이 구성 파일에 한 개 이상의 대상 수준이 정의되어 있어야 합니다.

표 24. MessageConnector 대상 수준 설정

요소	요소	설명	기본값
<audienceLevel>		대상 수준 구성을 포함하는 요소입니다. <audienceLevel name="Customer">에서와 같이 name 속성을 제공하십시오.	
	<messageLogTable>	로그 테이블 이름입니다. 이 값은 필수입니다.	UACI_MESSAGE_CONNECTOR_LOG
<fields>	<field>	이 audienceLevel에 대한 하나 이상의 대상 ID 필드 정의입니다.	
	<name>	Interact 런타임에 지정된 대상 ID 필드 이름입니다.	
	<httpParameterName>	이 대상 ID 필드에 대한 해당 매개변수 이름입니다.	
	<dbColumnName>	이 대상 ID 필드에 대한 데이터베이스의 해당 열 이름입니다.	
	<type>	Interact 런타임에 지정된 대상 ID 필드 유형입니다. 값은 string 또는 numeric입니다.	

Message Connector 테이블 작성

먼저 Interact 런타임 데이터가 저장된 데이터베이스에 테이블을 작성해야 IBM Unica Interact Message Connector를 배포할 수 있습니다. 정의한 각 대상 수준마다 하나의 테이블을 작성합니다. 각 대상 수준마다 Interact는 사용자가 작성하는 테이블을 사용하여 Message Connector 트랜잭션에 대한 정보를 기록합니다.

데이터베이스 클라이언트를 사용하여 해당 데이터베이스 또는 스키마에 대해 Message Connector SQL 스크립트를 실행하여 필수 테이블을 작성하십시오. 지원되는 데이터베이스용 SQL 스크립트는 Interact 런타임 서버를 설치할 때 자동으로 설치됩니다. Interact 런타임 테이블을 포함하는 데이터베이스에 연결에 대한 자세한 내용은 *IBM Unica Interact 설치 가이드*에서 완료한 워크시트를 참조하십시오.

1. 데이터베이스 클라이언트를 실행하고 Interact 런타임 테이블이 현재 저장된 데이터베이스에 연결하십시오.
2. *<Interact_home>*/msgconnector/scripts/ddl 디렉토리에서 해당 스크립트를 실행하십시오. 여기서 *<Interact_home>*은 Interact 런타임을 설치한 디렉토리입니다 (예: C:\Unica\Interact 또는 /Unica/Interact). 다음 테이블은 Message Connector 테이블을 수동으로 작성하는 데 사용할 수 있는 샘플 SQL 스크립트를 나열합니다.

표 25. Message Connector 테이블 작성을 위한 스크립트

데이터 소스 유형	스크립트 이름
IBM DB2	db_scheme_db2.sql
Microsoft SQL Server	db_scheme_sqlserver.sql
Oracle	db_scheme_oracle.sql

이러한 스크립트는 샘플로 제공됨에 유의하십시오. 대상 ID 값에 다른 이름 지정 규칙 또는 구조를 사용할 수 있으므로 실행 전에 스크립트를 수정해야 할 수도 있습니다. 일반적으로 각 대상 수준에 하나의 전용 테이블을 두는 것이 모범 사례입니다.

테이블은 다음 정보를 포함하도록 작성됩니다.

표 26. 샘플 SQL 스크립트가 작성한 정보

열 이름	설명
LogId	이 항목의 기본 키입니다.
MessageId	각 메시징 인스턴스의 고유 ID입니다.
LinkId	전자 미디어(예: 전자 메일 메시지)에 있는 각 링크의 고유 ID입니다.
OfferImageUrl	반환된 오퍼의 관련 이미지 URL입니다.
OfferLandingPageUrl	반환된 오퍼의 관련 랜딩 이미지 URL입니다.
TreatmentCode	반환된 오퍼의 처리 코드입니다.
OfferExpirationDate	반환된 오퍼의 만료 날짜 및 시간입니다.
OfferContactDate	클라이언트로 오퍼가 반환된 날짜 및 시간입니다.
AudienceId	전자 미디어의 대상 ID입니다.

이 테이블에 대해 다음을 유의하십시오.

- 대상 수준에 따라 대상 키의 각 구성 요소마다 하나의 AudienceId 열이 있습니다.
- MessageId, LinkId, AudienceId(s) 조합은 이 테이블의 고유 키를 형성합니다.

스크립트 실행이 완료되면, Message Connector에 필요한 테이블을 작성했습니다.

이제 Message Connector 웹 응용 프로그램을 배포할 준비가 되었습니다.

Message Connector 배포 및 실행

IBM Unica Interact Message Connector는 지원되는 웹 응용 프로그램 서버에서 독립형 웹 응용 프로그램으로 배포됩니다.

Message Connector를 배포하기 전에 다음 작업이 완료되었는지 확인하십시오.

- IBM Unica Interact 런타임 서버를 설치했어야 합니다. 배포 가능한 Message Connector 응용 프로그램은 런타임 서버와 함께 자동으로 설치되며 Interact 홈 디렉토리에서 배포할 준비가 되었습니다.
- 240 페이지의 『Message Connector 테이블 작성』에 설명된 대로 Message Connector가 사용할 Interact 런타임 데이터베이스에 필수 테이블을 작성하기 위해 설치와 함께 제공되는 SQL 스크립트도 실행했어야 합니다.

실행 전에 웹 응용 프로그램 서버에 기타 IBM Unica 응용 프로그램을 배포하는 경우와 마찬가지로 오퍼 제공에 사용할 수 있도록 Message Connector 응용 프로그램을 배포해야 합니다.

1. 응용 프로그램을 배포하는 데 필요한 권한을 사용하여 웹 응용 프로그램 서버 관리 인터페이스에 연결하십시오.
2. 웹 응용 프로그램 서버에 대한 지침을 따라 `<Interact_home>/msgconnector/MessageConnector.war` 파일을 배포 및 실행하십시오. *<Interact_home>*을 Interact 런타임 서버가 설치된 실제 디렉토리(예: `C:\Unica\Interact` 또는 `/Unica/Interact`)로 바꾸십시오.

이제 Message Connector를 사용할 수 있습니다. Interact 설치를 구성하여 Message Connector가 오퍼(예: 대화식 채널 및 전략, 플로차트, 오퍼 등)를 제공하는 데 사용할 기본 데이터를 작성한 후에는 Message Connector가 수락할 링크를 전자 미디어에 작성할 수 있습니다.

Message Connector 링크 작성

최종 사용자가 전자 미디어와 상호작용할 때(예를 들어, 전자 메일 메시지를 열어서) Message Connector를 사용하여 사용자 정의 오퍼 이미지를 제공하고 최종 사용자가 오퍼를 클릭할 때 사용자 정의 랜딩 페이지를 제공하려면, 메시지에 임베드할 링크를 작성해야 합니다. 이 섹션에서는 해당 링크의 HTML 태그 지정 요약을 제공합니다.

최종 사용자에게 발신되는 메시지를 생성하는 데 사용하는 시스템에 관계없이 Interact 런타임 서버로 전달할 정보가 들어 있는 해당 필드(HTML 태그에서 속성으로 제공됨)를 포함하도록 HTML 태그 지정을 생성해야 합니다. 아래 단계를 따라 Message Connector 메시지에 필요한 최소 정보를 구성하십시오.

여기 설명된 지침은 특히 Message Connector 링크를 포함하는 메시지에 적용되지만 동일한 단계 및 구성을 따라 웹 페이지 및 기타 전자 미디어에 링크를 추가할 수 있습니다.

1. 최소한 다음 매개변수를 사용하여 메시지에 나타날 IMG 링크를 작성하십시오.
   - msgID - 이 메시지의 고유 ID를 표시합니다.
   - linkID - 이 메시지에 있는 링크의 고유 ID를 표시합니다.
   - audienceID - 메시지 수신자가 속한 대상 ID입니다.

     대상 ID가 컴포지트 ID이면, 해당 구성 요소가 모두 링크에 포함되어야 합니다.

   대상 수준, 대화식 채널 이름, 상호작용 지점 이름, 이미지 위치 URL을 포함하는 선택적 매개변수 및 Message Connector에서 명확하게 사용하지 않는 자체 사용자 정의 매개변수도 포함시킬 수 있습니다.
2. 선택적으로, 사용자가 해당 이미지를 클릭하면 브라우저가 사용자에 대해 오퍼를 포함하는 페이지를 로드하도록 IMG 링크를 묶는 A 링크를 작성하십시오. A 링크는 앞서 나열한 세 개의 매개변수(msgID, linkID, audienceID)는 물론 선택적 매개변수(대상 수준, 대화식 채널 이름, 상호작용 지점 이름) 및 Message Connector에서 명확하게 사용하지 않는 사용자 정의 매개변수도 포함해야 합니다. A 링크는 Message Connector IMG 링크를 포함할 가능성이 높지만 필요에 따라 페이지에서 단독으로도 사용될 수 있습니다. 이 링크가 IMG 링크를 포함하지 않으면, IMG 링크가 묶는 A 링크와 동일한 매개변수 집합(선택적 또는 사용자 정의 매개변수 포함)을 포함해야 합니다.
3. 링크가 올바로 정의되면 전자 메일 메시지를 생성하여 보내십시오.

사용 가능한 매개변수 및 샘플 링크에 대한 자세한 정보는 『"IMG" 및 "A" 태그 HTTP 요청 매개변수』의 내용을 참조하십시오.

"IMG" 및 "A" 태그 HTTP 요청 매개변수

최종 사용자가 Message Connector 인코딩된 IMG 태그가 포함된 전자 메일을 열었거나 최종 사용자가 A 태그를 클릭했기 때문에 Message Connector가 요청을 수신하면, Message Connector는 요청에 포함된 매개변수를 구문 분석하여 해당 오퍼 데이터를 리턴합니다. 이 절에서는 요청 URL(IMG 태그(전자 메일이 열릴 때 태그 지정된 이미지가 표시되면 자동으로 로드됨) 또는 A 태그(전자 메일을 보는 사람이 지정된 사이트로 연결하는 메시지를 클릭하면 로드됨))에 포함될 수 있는 매개변수 목록을 제공합니다.

매개변수

Message Connector는 요청을 수신하면 요청에 포함된 매개변수를 구문 분석합니다. 이러한 매개변수는 다음 중 일부 또는 모두를 포함합니다.

매개변수 이름	설명	필수 여부	기본값
msgId	전자 메일 인스턴스 또는 웹 페이지의 고유 ID입니다.	예	없음. 태그가 포함된 전자 메일 메시지 또는 웹 페이지의 고유 인스턴스를 작성하는 시스템이 제공합니다.

매개변수 이름	설명	필수 여부	기본값
linkId	이 전자 메일 또는 웹 페이지에 있는 링크의 고유 ID입니다.	예	없음. 태그가 포함된 전자 메일 메시지 또는 웹 페이지의 고유 인스턴스를 작성하는 시스템이 제공합니다.
audienceLevel	이 통신 수신자가 속한 대상 수준입니다.	아니요	MessageConnectorConfig.xml 파일에 있는 audienceLevels 요소에 기본값으로 지정된 audienceLevel.
ic	대상 대화식 채널(IC) 이름입니다.	아니요	MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 있는 interactiveChannel 요소 값(기본적으로 "interactiveChannel"임).
ip	적용 중인 상호작용 지점(IC) 이름입니다.	아니요	MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 있는 interactionPoint 요소 값(기본적으로 "headBanner"임).
offerImageUrl	메시지의 IMG URL에 대한 대상 오퍼 이미지 URL입니다.	아니요	없음.
offerImageUrlAttr	대상 오퍼 이미지 URL을 갖는 오퍼 속성 이름입니다.	아니요	MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 있는 imageUrlAttribute 요소 값.
offerLandingPageUrl	대상 오퍼에 대응하는 랜딩 페이지 URL입니다.	아니요	없음.
offerLandingPageUrlAttr	대상 오퍼에 대응하는 랜딩 페이지 URL을 갖는 오퍼 속성 이름입니다.	아니요	MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 있는 landingPageUrlAttribute 요소 값.
contactEvent	컨택 이벤트 이름입니다.	아니요	MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 있는 contactEventName 요소 값(기본적으로 "contact"임).
responseEvent	수락 이벤트 이름입니다.	아니요	MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 있는 acceptEventName 요소 값(기본적으로 "accept"임).
debug	디버그 플래그입니다. 문제점 해결 및 IBM Unica 기술 지원의 지시가 있는 경우에만 이 매개변수를 "true"로 설정하십시오.	아니요	MessageConnectorConfig.xml 파일의 defaultParameterValues 섹션에 있는 debugFlag 요소 값(기본적으로 "false"임).
<audience id>	이 사용자의 대상 ID입니다. 이 매개변수 이름은 구성 파일에 정의됩니다.	예	없음.

Message Connector가 알 수 없는(즉, 위 목록에 나타나지 않은) 매개변수를 수신하는 경우, 매개변수는 다음 두 가지 방법 중 하나로 처리됩니다.

- 알 수 없는 매개변수가 제공되고(예: attribute="attrValue"에서의 "attribute")되고 동일한 이름과 "Type" 단어가 있는 일치하는 매개변수가 있으면 (예: attributeType="string"에서의 "attributeType"), Message Connector는 일치하는 Interact 매개변수를 작성하여 Interact 런타임에 전달합니다.

  Type 매개변수 값은 다음 중 하나입니다.

  – string
  – numeric
  – datetime

"datetime" 유형의 매개변수의 경우, Message Connector는 값이 유효한 날짜/시간 형식인 동일한 이름과 "Pattern" 단어(예: "attributePattern")가 있는 매개변수도 검색합니다. 예를 들어, `attributePattern="MM/dd/yyyy"` 매개변수를 제공할 수 있습니다.

"datetime" 유형의 매개변수를 지정하지만 일치하는 날짜 패턴을 제공하지 않으면, Interact 서버의 Message Connector 구성 파일 (`<installation_directory>/msgconnector/config/MessageConnectorConfig.xml`에 있음)에 지정된 값이 사용됩니다.

- 알 수 없는 매개변수가 제공되고 일치하는 Type 값이 없으면, Message Connector가 해당 매개변수를 대상 경로 재지정 URL로 전달합니다.

알 수 없는 모든 매개변수의 경우, Message Connector는 매개변수를 처리하거나 저장하지 않고 Interact 런타임 서버로 전달합니다.

Message Connector 코드 예제

다음 A 태그는 전자 메일 메시지에 나타날 수 있는 Message Connector 링크 집합 예제를 포함합니다.

```
<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>
```

이 예제에서 IMG 태그는 전자 메일 메시지가 열릴 때 자동으로 로드됩니다. 지정된 페이지에서 이미지를 검색하여 메시지는 전달될 두 개의 추가 매개변수(incomeCode 및 incomeType)와 함께 고유 메시지 ID(msgID), 고유 링크 ID(linkID), 고유 사용자 ID(userid)에 대한 매개변수를 Interact 런타임으로 전달합니다.

A 태그는 오퍼 이미지를 전자 메일 메시지에서 클릭 가능한 링크가 되게 하는 HREF(Hypertext Reference) 속성을 제공합니다. 메시지 뷰어가 이미지를 보는 즉시 랜딩 페이지로 연결하면, 대상 경로 재지정 URL로 전달되는 하나의 추가 매개변수(`referral`)는 물론 고유 메시지 ID(`msgId`), 링크 ID(`linkId`), 사용자 ID(`userid`)가 서버로 전달됩니다.

Interact Web Connector 정보

Interact WebConnector(JavaScript Connector, 또는 JSConnector라고도 함)는 Interact 런타임 서버에서 JavaScript 코드가 Interact Java API를 호출할 수 있게 하는 서비스를 제공합니다. 이를 통해 웹 페이지는 웹 개발 언어(예: Java, PHP, JSP 등)에 의존하지 않고 유일하게 임베드된 JavaScript 코드를 사용하여 실시간 오퍼 사용자 개인 설정에 대해 Interact를 호출할 수 있습니다. 예를 들어, 웹 사이트의 각 페이지에 Interact

에서 권장하는 오퍼를 제공하는 작은 JavaScript 코드 스니펫을 임베드할 수 있습니다. 따라서 사이트 방문자의 로딩 페이지에 최상의 오퍼가 표시되도록 페이지를 로드할 때마다 Interact API 호출이 수행됩니다.

페이지 표시를 통한 서버 측 프로그램 제어는 없지만(예를 들어, PHP 또는 기타 서버 기반 스크립팅의 경우) 방문자의 웹 브라우저를 통해 실행될 페이지 컨텐츠에 JavaScript 코드를 여전히 임베드할 수 있는 페이지에서 방문자에게 오퍼를 표시하려는 상황에서 Interact Web Connector를 사용하십시오.

팁: Interact Web Connector 파일은 Interact 런타임 서버의 *<Interact_home>*/jsconnector 디렉토리에 자동으로 설치됩니다. *<Interact_home>*/jsconnector 디렉토리에서 Web Connector 기능에 대한 중요한 메모 및 세부 정보는 물론 자체 솔루션 개발의 기초로 사용할 샘플 파일과 Web Connector 소스 코드가 들어 있는 ReadMe.txt를 찾을 수 있습니다. 여기서 질문에 대답할 정보를 찾지 못할 경우, 자세한 정보는 jsconnector 디렉토리를 참조하십시오.

런타임 서버에 Web Connector 설치

Web Connector 인스턴스는 IBM Unica Interact 런타임 서버와 함께 자동으로 설치되며, 기본적으로 활성화되어 있습니다. 그러나 일부 설정을 수정해야 Web Connector를 구성 및 사용할 수 있습니다.

런타임 서버에 설치된 Web Connector를 구성 및 사용하기 전에 수정해야 하는 일부 설정은 웹 응용 프로그램 서버의 구성에 추가됩니다. 이러한 이유로 다음 단계를 완료한 후에는 웹 응용 프로그램 서버를 다시 시작해야 합니다.

1. Interact 런타임 서버가 설치된 웹 응용 프로그램 서버의 경우, 다음 Java 등록 정보를 설정하십시오.

   ```
   -DUI_JSCONNECTOR_ENABLE_INPROCESS=true
   ```

   ```
   -DUI_JSCONNECTOR_HOME=<jsconnectorHome>
   ```

   *<jsconnectorHome>*을 런타임 서버의 jsconnector 디렉토리(<Interact_installation_directory>/jsconnector)로 바꾸십시오.

   예를 들어, Windows 설치에서는 C:\Unica\Interact\jsconnector입니다. UNIX 시스템에서는 이 값에 대해 /Unica/Interact/jsconnector를 입력하십시오.

   Java 등록 정보 설정 방법은 해당 웹 응용 프로그램 서버에 따라 다릅니다. 예를 들어, WebLogic에서는 startWebLogic.sh 또는 startWebLogic.cmd 파일을 편집하여 다음 예제에서와 같이 JAVA_OPTIONS 설정을 업데이트하십시오.

   ```
   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/
   jsconnector"
   ```

WebSphere Application Server에서는 관리 콘솔의 JVM(Java Virtual Machine) 패널에서 이 등록 정보를 설정하십시오.

Java 등록 정보 설정에 대한 특정 지침은 해당 웹 응용 프로그램 서버 문서를 참조하십시오.

2. 웹 응용 프로그램 서버가 이미 실행되고 있었으면 다시 시작하거나 웹 응용 프로그램 서버를 지금 시작하여 새 Java 등록 정보가 사용되는지 확인하십시오.

웹 응용 프로그램 서버가 시작 프로세스를 완료했으면 런타임 서버에 Web Connector 설치를 완료했습니다. 다음 단계는 Web Connector 구성 웹 페이지 (`http://<host>:<port>/interact/jsp/WebConnector.jsp`)에 연결하는 것입니다. 여기서 *<host>*는 Interact 런타임 서버 이름이고 *<port>*는 웹 응용 프로그램 서버가 지정한 대로 Web Connector가 청취 중인 포트입니다.

별도 웹 응용 프로그램으로 Web Connector 설치

Web Connector 인스턴스는 IBM Unica Interact 런타임 서버와 함께 자동으로 설치되며, 기본적으로 활성화되어 있습니다. 그러나 Web Connector를 자체 웹 응용 프로그램(예: 별도 시스템의 웹 응용 프로그램 서버)으로도 배포하고 원격 Interact 런타임 서버와 통신하도록 구성할 수 있습니다.

다음 지침은 Web Connector를 Interact 런타임 서버에 대한 액세스 권한을 가진 별도 웹 응용 프로그램으로 배포하는 프로세스를 설명합니다.

IBM Unica Interact 런타임 서버를 설치했어야 하며 Interact 런타임 서버에 대한 네트워크 액세스 권한(방화벽으로 차단되지 않음)을 가진 다른 시스템에 웹 수신인 서버가 있어야 Web Connector를 배포할 수 있습니다.

1. Interact 런타임 서버에서 웹 응용 프로그램 서버(예: WebSphere Application Server)가 이미 구성되어 실행 중인 시스템으로 Web Connector 파일이 포함된 `jsconnector` 디렉토리를 복사하십시오. Interact 설치 디렉토리(예: `C:\Unica\Interact` 또는 `/Unica/Interact`)에서 `jsconnector` 디렉토리를 찾을 수 있습니다.
2. Web Connector 인스턴스를 배포할 시스템에서 텍스트 또는 XML 편집기를 사용하여 `jsconnector/jsconnector.xml` 파일을 구성하여 `interactURL` 속성을 수정하십시오.

   기본적으로 `http://localhost:7001/interact`로 설정되어 있지만 원격 Interact 런타임 서버 URL과 일치하도록 수정해야 합니다 (예: `http://runtime.example.com:7011/interact`).

   Web Connector를 배포한 후에는 웹 인터페이스를 사용하여 `jsconnector.xml` 파일의 나머지 설정을 사용자 정의할 수 있습니다. 자세한 정보는 248 페이지의 『Web Connector 구성』의 내용을 참조하십시오.

3. Web Connector를 배포할 웹 응용 프로그램 서버의 경우, 다음 Java 등록 정보를 설정하십시오.

   -DUI_JSCONNECTOR_HOME=<*jsconnectorHome*>

   <*jsconnectorHome*>을 웹 응용 프로그램 서버로 jsconnector 디렉토리를 복사한 실제 경로로 바꾸십시오.

   예를 들어, Windows 설치에서는 C:\Unica\Interact\jsconnector입니다. UNIX 시스템에서는 이 값에 대해 /Unica/Interact/jsconnector를 입력하십시오.

   Java 등록 정보 설정 방법은 해당 웹 응용 프로그램 서버에 따라 다릅니다. 예를 들어, WebLogic에서는 startWebLogic.sh 또는 startWebLogic.cmd 파일을 편집하여 다음 예제에서와 같이 JAVA_OPTIONS 설정을 업데이트하십시오.

   JAVA_OPTIONS="${SAVE_JAVA_OPTIONS} -DUI_JSCONNECTOR_HOME=/UnicaFiles/jsconnector"

   WebSphere Application Server에서는 관리 콘솔의 JVM(Java Virtual Machine) 패널에서 이 등록 정보를 설정하십시오.

   Java 등록 정보 설정에 대한 특정 지침은 해당 웹 응용 프로그램 서버 문서를 참조하십시오.
4. 웹 응용 프로그램 서버가 이미 실행되고 있었으면 다시 시작하거나 이 단계에서 웹 응용 프로그램 서버를 시작하여 새 Java 등록 정보가 사용되는지 확인하십시오.

   웹 응용 프로그램 서버가 시작 프로세스를 완료하기를 기다렸다가 계속하십시오.
5. 응용 프로그램을 배포하는 데 필요한 권한을 사용하여 웹 응용 프로그램 서버 관리 인터페이스에 연결하십시오.
6. 웹 응용 프로그램 서버에 대한 지침을 따라 다음 파일을 배포 및 실행하십시오. jsConnector/jsConnector.war

이제 Web Connector가 웹 응용 프로그램에 배포됩니다. 완전히 구성한 Interact 서버가 작동되어 실행된 후 다음 단계는 Web Connector 구성 웹 페이지(http:// <*host*>: <*port*>/interact/jsp/WebConnector.jsp)에 연결하는 것입니다. 여기서 <*host*>는 방금 Web Connector를 배포한 웹 응용 프로그램 서버를 실행 중인 시스템이고 <*port*>는 웹 응용 프로그램 서버가 지정한 대로 Web Connector가 청취 중인 포트입니다.

Web Connector 구성

Interact Web Connector 구성 설정은 Web Connector가 배포된 시스템(예: Interact 런타임 서버 자체 또는 웹 응용 프로그램 서버를 실행 중인 별도의 시스템)에 저장된 jsconnector.xml 파일에 저장됩니다. 텍스트 편집기 또는 XML 편집기를 사용하여

직접 jsconnector.xml 파일을 편집할 수 있지만 사용 가능한 구성 설정을 거의 모두 구성하는 용이한 방법은 웹 브라우저에서 Web Connector 구성 페이지를 사용하는 것입니다.

Web Connector를 제공하는 응용 프로그램을 먼저 설치 및 배포해야 웹 인터페이스를 사용하여 Web Connector를 구성할 수 있습니다. Interact 런타임 서버에서는 Interact를 설치 및 배포하면 Web Connector 인스턴스가 자동으로 설치됩니다. 기타 웹 응용 프로그램 서버에서는 247 페이지의 『별도 웹 응용 프로그램으로 Web Connector 설치』에 설명된 대로 Web Connector 웹 응용 프로그램을 설치 및 배포해야 합니다.

1. 지원되는 웹 브라우저를 열고 다음과 유사한 URL을 여십시오.

   `http://<host>:<port>/interact/jsp/WebConnector.jsp`

   - <*host*>를 Web Connector가 실행 중인 서버(예: 런타임 서버의 호스트 이름 또는 별도의 Web Connector 인스턴스를 배포한 서버 이름)로 바꾸십시오.
   - <*port*>를 Web Connector 웹 응용 프로그램이 연결을 청취하고 있는 포트 번호(일반적으로 웹 응용 프로그램 서버의 기본 포트와 일치함)로 바꾸십시오.
2. 나타나는 구성 페이지에서 다음 섹션을 완료하십시오.

표 27. Web Connector 구성 설정 요약.

섹션	설정
기본 설정	태그 지정된 페이지를 롤아웃할 사이트에 대해 전체 Web Connector 동작을 구성하려면 기본 설정 페이지를 사용하십시오. 이 설정은 사이트의 기본 URL, Interact가 사용해야 하는 사이트 방문자에 대한 정보, Web Connector 코드를 사용하여 태그를 지정할 모든 페이지에 적용되는 유사한 설정을 포함합니다. 자세한 내용은 251 페이지의 『WebConnector Connector 구성 기본 옵션』의 내용을 참조하십시오.
HTML 표시 유형	페이지에서 각 상호작용 지점에 제공될 HTML 코드를 판별하려면 HTML 표시 유형 페이지를 사용하십시오. 각 상호작용 지점에 사용할 캐스케이딩 스타일시트(CSS) 코드, HTML 코드, Javascript 코드의 일부 조합을 포함하는 기본 템플릿(.flt 파일) 목록에서 선택할 수 있습니다. 템플릿을 제공된 대로 사용하고 필요에 따라 사용자 정의하거나 직접 작성할 수 있습니다. 이 페이지의 구성 설정은 jsconnector.xml 구성 파일의 interactionPoints 섹션에 해당합니다. 자세한 내용은 252 페이지의 『WebConnector 구성 HTML 표시 유형』의 내용을 참조하십시오.

표 27. Web Connector 구성 설정 요약 (계속).

섹션	설정
고급 페이지	페이지 고유 설정을 URL 패턴에 맵핑하려면 고급 페이지를 사용하십시오. 예를 들어, 해당 맵핑에 대해 특정 페이지 로드 이벤트 및 상호작용 지점을 정의하여 "index.htm" 텍스트를 포함하는 URL이 일반 시작 페이지를 표시하도록 페이지 맵핑을 설정할 수 있습니다. 이 페이지의 구성 설정은 `jsconnector.xml` 구성 파일의 `pageMapping` 섹션에 해당합니다. 자세한 내용은 255 페이지의 『WebConnector 구성 고급 페이지』의 내용을 참조하십시오.

3. 기본 설정 페이지에서 사이트 전체 설정이 설치에 유효한지 확인하고 디버그 모드(문제점을 해결하는 중이 아니면 권장하지 않음), NetInsight 페이지 태그 통합, 대부분의 상호작용 지점의 기본 설정을 선택적으로 지정한 후 구성 아래의 HTML 표시 유형 링크를 클릭하십시오.
4. HTML 표시 유형 페이지에서 다음 단계를 따라 고객 웹 페이지에 상호작용 지점을 정의하는 표시 템플릿을 추가하거나 수정하십시오.

   기본적으로 표시 템플릿(.flt 파일)은 *<jsconnector_home>*`/conf/html`에 저장됩니다.

   a. 시작점으로 사용하거나 검사할 목록에서 .flt 파일을 선택하거나 유형 추가를 클릭하여 사용할 비어 있는 새 상호작용 지점 템플릿을 작성하십시오.

      템플릿 컨텐츠에 대한 정보(있는 경우)가 템플릿 목록 옆에 나타납니다.

   b. 선택적으로 **이 표시 유형의 파일 이름** 필드에서 템플릿 이름을 수정하십시오. 새 템플릿의 경우, `CHANGE_ME.flt`를 보다 의미 있는 이름으로 업데이트하십시오.

      여기서 템플릿 이름을 변경하면, 다음 번에 템플릿이 작성될 때 Web Connector가 해당 이름을 사용하여 새 파일을 작성합니다. 템플릿은 텍스트 본문을 수정한 후 기타 필드로 이동하면 저장됩니다.

   c. 포함시킬 스타일시트(CSS), JavaScript HTML 코드를 포함하여 HTML 스니펫 정보를 필요에 따라 수정하거나 채우십시오. 런타임 시 Interact 매개변수로 바뀔 변수도 포함시킬 수 있습니다. 예를 들어, `${offer.HighlightTitle}`은 지정된 상호작용 지점 위치의 오퍼 제목으로 자동으로 바뀝니다.

      CSS, JavaScript 또는 HTML 코드 블록 형식화 방법에 대한 표시는 HTML 스니펫 필드 아래에 나타나는 예제를 사용하십시오.
5. 고급 페이지를 필요에 따라 사용하여 페이지에서 특정 URL 처리 방법을 결정하는 페이지 맵핑을 설정하십시오.
6. 구성 등록 정보 설정을 완료했으면 **변경 사항 롤아웃**을 클릭하십시오. **변경 사항 롤아웃**을 클릭하면 다음 작업이 수행됩니다.

- Web Connector 페이지에서 복사하여 웹 페이지에 삽입할 수 있는 JavaScript 코드를 포함하는 IBM Unica Interact Web Connector 페이지 태그가 표시됩니다.
- Interact 서버의 기존 Web Connector 구성 파일(Web Connector가 설치된 서버의 jsconnector.xml 파일)을 백업하고 사용자가 정의한 설정을 사용하여 새 구성 파일을 작성합니다.

  백업 구성 파일은 *<jsconnector_home>*/conf/archive/jsconnector.xml.*<date>*.*<time>*(예: jsconnector.xml.20111113.214933.750-0500)에 저장됩니다(여기서 날짜 문자열은 20111113이고 시간대 표시기를 포함하여 시간 문자열은 214933.750-0500임).

이제 Web Connector 구성을 완료했습니다.

구성을 수정하려면 이 단계의 처음으로 돌아가 새 값을 사용하여 단계를 다시 수행하거나 텍스트 또는 XML 편집기에서 구성 파일(<Interact_home>/jsconnector/conf/jsconnector.xml)을 열고 필요에 따라 수정할 수 있습니다.

WebConnector Connector 구성 기본 옵션

태그 지정된 페이지를 롤아웃할 사이트에 대해 전체 Web Connector 동작을 구성하려면 Web Connector 구성 페이지의 기본 설정 페이지를 사용하십시오. 이 설정은 사이트의 기본 URL, Interact가 사용해야 하는 사이트 방문자에 대한 정보, Web Connector 코드를 사용하여 태그를 지정할 모든 페이지에 적용되는 유사한 설정을 포함합니다.

사이트 전체 설정

사이트 전체 설정 구성 옵션은 구성 중인 Web Connector 설치의 전체 동작에 영향을 주는 전역 설정입니다. 지정할 수 있는 값은 다음과 같습니다.

표 28. Web Connector 설치의 사이트 전체 설정

설정	설명	**jsconnector.xml**의 동등한 설정
Interact API URL	Interact 런타임 서버의 기본 URL입니다. **참고:** 이 설정은 Web Connector가 Interact 런타임 서버 내에서 실행되고 있지 않은 경우(즉, 별도로 배포한 경우)에만 사용됩니다.	*<interactURL>*
Web Connector URL	다른 사이트로 연결 URL을 생성하는 데 사용되는 기본 URL입니다.	*<jsConnectorURL>*
대상 웹 사이트의 대화식 채널 이름	Interact 서버에 정의한 이 페이지 맵핑을 나타내는 대화식 채널 이름입니다.	*<interactiveChannel>*
방문자의 대상 수준	인바운드 방문자의 Campaign 대상 수준으로, Interact 런타임에 대한 API 호출에서 사용됩니다.	*<audienceLevel>*

표 28. Web Connector 설치의 사이트 전체 설정 (계속)

설정	설명	**jsconnector.xml**의 동등한 설정
프로파일 테이블의 대상 ID 필드 이름	Interact 런타임에 대한 API 호출에서 사용될 audienceId 필드 이름입니다. 다중 필드 대상 ID는 현재 지원되지 않습니다.	<*audienceIdField*>
대상 ID 필드의 데이터 유형	Interact 런타임에 대한 API 호출에서 사용될 대상 ID 필드의 데이터 유형입니다 ("numeric" 또는 "string").	<*audienceIdFieldType*>
세션 ID를 나타내는 쿠키 이름	세션 ID를 포함할 쿠키 이름입니다.	<*sessionIdCookie*>
방문자 ID를 나타내는 쿠키 이름	방문자 ID를 포함할 쿠키 이름입니다.	<*visitorIdCookie*>

선택적 기능

선택적 기능 구성 옵션은 구성 중인 Web Connector 설치의 선택적 전역 설정입니다. 지정할 수 있는 값은 다음과 같습니다.

표 29. Web Connector 설치의 선택적 사이트 전체 설정

설정	설명	**jsconnector.xml**의 동등한 설정
디버그 모드 설정	`yes` 또는 `no` 응답을 사용하여 특수 디버그 모드 사용 여부를 지정합니다. 이 기능을 설정하는 경우, Web Connector에서 리턴된 컨텐츠는 클라이언트에게 방금 발생한 특정 페이지 맵핑을 알리는 '알림'에 대한 Javascript 호출을 포함합니다. 알림을 가져오려면 클라이언트의 <*authorizedDebugClients*> 설정이 지정한 파일에 항목이 있어야 합니다.	<*enableDebugMode*>
승인된 디버깅 클라이언트 호스트 파일	디버그 모드를 규정하는 호스트 또는 IP(Internet Protocol) 주소 목록을 포함하는 파일 경로입니다. 디버그 정보를 수집하려면 지정된 파일에 클라이언트의 호스트 이름 또는 IP 주소가 나타나야 합니다.	<*authorizedDebugClients*>
NetInsight 페이지 태그 통합 설정	`yes` 또는 `no` 응답을 사용하여 Web Connector가 지정된 IBM Unica NetInsight 태그를 페이지 컨텐츠 끝에 첨부해야 하는지 여부를 지정합니다.	`<enableNetInsightTagging>`
NetInsight 태그 HTML 템플릿 파일	NetInsight 태그 호출을 통합하는 데 사용되는 HTML/Javascript 템플릿입니다. 일반적으로 다른 템플릿을 제공하도록 지시하지 않으면 기본 설정을 적용해야 합니다.	`<netInsightTag>`

WebConnector 구성 HTML 표시 유형

페이지에서 각 상호작용 지점에 제공될 HTML 코드를 판별하려면 HTML 표시 유형 페이지를 사용하십시오. 각 상호작용 지점에 사용할 캐스케이딩 스타일시트(CSS) 코드, HTML 코드, JavaScript 코드의 일부 조합을 포함하는 기본 템플릿(.flt 파일) 목록에서 선택할 수 있습니다. 템플릿을 제공된 대로 사용하고 필요에 따라 사용자 정의하거나 직접 작성할 수 있습니다.

참고: 이 페이지의 구성 설정은 jsconnector.xml 구성 파일의 interactionPoints 섹션에 해당합니다.

상호작용 지점은 자동으로 삭제할 수 있는 오퍼 속성에 자리 표시자(영역)도 포함할 수 있습니다. 예를 들어, 상호작용 중에 해당 오퍼에 할당된 처리 코드로 바뀌는 ${offer.TREATMENT_CODE}를 포함시킬 수 있습니다.

이 페이지에 나타나는 템플릿은 Web Connector 서버의 <Interact_home>/jsconnector/conf/html 디렉토리에 저장된 파일에서 자동으로 로드됩니다. 여기서 작성하는 새 템플릿도 해당 디렉토리에 저장됩니다.

HTML 표시 유형 페이지를 사용하여 기존 템플릿을 보거나 수정하려면 목록에서 .flt 파일을 선택하십시오.

HTML 표시 유형 페이지에서 새 템플릿을 작성하려면 **유형 추가**를 클릭하십시오.

템플릿을 작성하거나 수정하기 위해 선택하는 방법에 관계없이 다음 정보가 템플릿 목록 옆에 나타납니다.

설정	설명	**jsconnector.xml**의 동등한 설정
이 표시 유형에 대한 파일 이름	편집 중인 템플릿에 할당된 이름입니다. 이 이름은 Web Connector가 실행 중인 운영 체제에 유효해야 합니다. 예를 들어, 운영 체제가 Microsoft Windows인 경우에는 이름에 슬래시(/)를 사용할 수 없습니다. 새 템플릿을 작성 중인 경우, 이 필드는 CHANGE_ME.flt로 미리 설정되어 있습니다. 이를 의미 있는 값으로 변경한 후 계속해야 합니다.	<*htmlSnippet*>

설정	설명	jsconnector.xml의 동등한 설정
HTML 스니펫	Web Connector가 웹 페이지에서 상호작용 지점에 삽입해야 하는 특정 컨텐츠입니다. 이 스니펫은 페이지에서 실행될 HTML 코드, CSS 형식화 정보 또는 JavaScript를 포함할 수 있습니다. 다음 예제에서와 같이 해당 세 가지 유형의 컨텐츠를 각각 BEGIN 및 END 코드로 묶어야 합니다. • ${BEGIN_HTML} <*your HTML content*> ${END_HTML} • ${BEGIN_CSS} <*your Interaction Point-specific stylesheet information*> ${END_CSS} • ${BEGIN_JAVASCRIPT} <*your Interaction Point-specific JavaScript code*> ${END_JAVASCRIPT} 또한 다음을 포함하여 페이지가 로드될 때 자동으로 바뀌는 미리 정의된 다수의 특수 코드를 입력할 수 있습니다. • `${logAsAccept}`: 두 개의 매개변수(오퍼 수락을 식별하는 데 사용되는 TreatmentCode 및 대상 URL)를 취하고 이를 다른 사이트로 연결 URL로 바꾸는 매크로. • `${offer.AbsoluteLandingPageURL}` • `${offer.OFFER_CODE}` • `${offer.TREATMENT_CODE}` • `${offer.TextVersion}` • `$offer.AbsoluteBannerURL}` 여기 나열된 오퍼 코드는 각각 마케팅 담당자가 Interact가 반환 중인 오퍼를 작성하는 데 사용한 IBM Unica Campaign의 오퍼 템플릿에 정의된 오퍼 속성을 나타냅니다. Web Connector는 페이지 템플릿에 코드 설정 시 유용한 다수의 추가 옵션을 제공하는 FreeMarker라는 템플릿 엔진을 사용합니다. 자세한 정보는 http://freemarker.org/docs/index.html의 내용을 참조하십시오.	HTML 스니펫은 jsconnector.xml과 별개의 자체 파일에 상주하기 때문에 동등한 설정이 없습니다.

설정	설명	jsconnector.xml의 동등한 설정
특수 코드 예제	블록을 HTML, CSS 또는 JAVASCRIPT로 식별하는 코드, 특정 오퍼 메타데이터를 참조하기 위해 삽입할 수 있는 삭제 가능한 영역을 포함하여 특수 코드 유형 샘플을 포함합니다.	동등한 설정이 없습니다.

이 페이지 변경 사항은 다른 Web Connector 구성 페이지로 이동하면 자동으로 저장됩니다.

WebConnector 구성 고급 페이지

페이지 고유 설정을 URL 패턴에 맵핑하려면 고급 페이지를 사용하십시오. 예를 들어, 해당 맵핑에 대해 특정 페이지 로드 이벤트 및 상호작용 지점을 정의하여 "index.htm" 텍스트를 포함하는 수신 URL이 일반 시작 페이지를 표시하도록 페이지 맵핑을 설정할 수 있습니다.

참고: 이 페이지의 구성 설정은 `jsconnector.xml` 구성 파일의 `pageMapping` 섹션에 해당합니다.

고급 페이지를 사용하여 새 페이지 맵핑을 작성하려면 **페이지 추가** 링크를 클릭하고 맵핑에 필요한 정보를 채우십시오.

페이지 정보

페이지 맵핑에 대한 페이지 정보 구성 옵션은 이 맵핑에 대한 트리거의 역할을 하는 URL 패턴과 Interact가 이 페이지 맵핑을 처리하는 방법에 대한 일부 추가 설정을 정의합니다.

설정	설명	jsconnector.xml의 동등한 설정
URL 포함 내용	Web Connector가 수신 페이지 요청을 감시해야 하는 URL 패턴입니다. 예를 들어, 요청 URL에 "mortgage.htm"이 포함된 경우 이를 모기지 정보 페이지와 일치시킬 수 있습니다.	*<urlPattern>*
이 페이지 또는 페이지 집합의 친숙한 이름	이 페이지 맵핑 목적을 설명하는 자체 참조를 위한 의미 있는 이름입니다(예: "모기지 정보 페이지")입니다.	*<friendlyName>*
JavaScript 사용을 위한 JSON 데이터로도 오퍼 반환	Web Connector가 JavaScript Object Notation(http://www.json.org/) 형식의 원시 오퍼 데이터를 페이지 컨텐츠 끝에 포함시킬지 여부를 표시하는 드롭다운 목록입니다.	*<enableRawDataReturn>*

이 페이지 또는 페이지 집합 방문 시 실행할(온로드) 이벤트

페이지 맵핑에 대한 이 구성 옵션 집합은 이 맵핑에 대한 트리거의 역할을 하는 URL 패턴과 Interact가 이 페이지 맵핑을 처리하는 방법에 대한 일부 추가 설정을 정의합니다.

참고: 이 섹션의 구성 설정은 `jsconnector.xml`의 `<pageLoadEvents>` 섹션에 해당합니다.

설정	설명	**jsconnector.xml**의 동등한 설정
개별 이벤트	이 페이지 또는 페이지 집합에 사용 가능한 이벤트 목록입니다. 이 목록의 이벤트는 Interact에 정의한 이벤트입니다. 페이지가 로드될 때 발생할 하나 이상의 이벤트를 선택하십시오. Interact API 호출 시퀀스는 다음과 같습니다. 1. `startSession` 2. 개별 페이지 로드 이벤트에 대한 `postEvent`(Interact에 개별 이벤트를 정의한 경우) 3. 각 상호작용 지점에 대해: • `getOffers` • `postEvent(ContactEvent)`	*<event>*

이 페이지 또는 페이지 집합의 상호작용 지점(오퍼 표시 위치)

페이지 맵핑에 대한 이 구성 옵션 집합을 사용하여 pageInteract에 나타나는 상호작용 지점을 선택할 수 있습니다.

참고: 이 섹션의 구성 설정은 `jsconnector.xml`의 `<pageMapping>` | `<page>` | `<interactionPoints>` 섹션에 해당합니다.

설정	설명	**jsconnector.xml**의 동등한 설정
상호작용 지점 이름 확인란	구성 파일에 정의된 각 상호작용 지점이 페이지의 이 섹션에 나타납니다. 상호작용 지점의 이름 옆에 이는 확인란을 선택하면 상호작용 지점에 사용 가능한 다수의 옵션이 표시됩니다.	*<interactionPoint>*
HTML 요소 ID(Interact가 innerHTML을 설정)	이 상호작용 지점에 대한 컨텐츠를 수신해야 하는 HTML 요소 이름입니다. 예를 들어, 페이지에 `<div id="welcomebanner">`를 지정한 경우 이 필드에 `welcomebanner`(ID 값)를 입력하십시오.	*<htmlElementId>*

설정	설명	jsconnector.xml의 동등한 설정
HTML 표시 유형	이 상호작용 지점에 사용할 HTML 표시 유형(이전 Web Connector 구성 페이지에 정의된 HTML 스니펫 또는 .flt 파일)을 선택할 수 있게 하는 드롭다운 목록입니다.	<*htmlSnippet*>
제공할 최대 오퍼 수(캐루젤 또는 플립북인 경우)	Web Connector가 이 상호작용 지점에 대해 Interact 서버에서 검색해야 하는 최대 오퍼 수입니다. 이 필드는 선택 사항이며, 한 번에 하나씩 사용할 있도록 다중 오퍼가 검색되는 캐루젤 시나리오에서와 같이 페이지를 다시 로드하지 않고 제공된 오퍼를 정기적으로 업데이트하는 상호작용 지점에 대해서만 적용됩니다.	<*maxNumberOfOffers*>
오퍼 제공 시 실행할 이벤트	이 상호작용 지점에 대해 게시될 컨택 이벤트 이름입니다	<*contactEvent*>
오퍼 수락 시 실행할 이벤트	오퍼 링크를 클릭하면 이 상호작용 지점에 대해 게시될 수락 이벤트 이름입니다	<*acceptEvent*>
오퍼 거부 시 실행할 이벤트	이 상호작용 지점에 대해 게시될 거부 이벤트 이름입니다 **참고:** 이때 이 기능은 아직 사용되지 않습니다.	<*rejectEvent*>

Web Connector 구성 옵션

일반적으로 그래픽 Web Connector 인터페이스를 사용하여 Web Connector 설정을 구성할 수 있습니다. 사용자가 지정하는 모든 설정 역시 `jsconnector/conf` 디렉토리에 있는 jsconnector.xml 파일에 저장됩니다. 여기서는 jsconnector.xml 구성 파일에 저장된 매개변수를 각각 설명합니다.

매개변수 및 설명

다음 매개변수는 `jsconnector.xml` 파일에 저장되며 Web Connector 상호작용에 사용됩니다. 이 설정을 수정하는 다음 두 가지 방법이 있습니다.

- Web Connector 응용 프로그램을 배포하여 시작한 후 자동으로 사용 가능한 Web Connector 구성 웹 페이지 사용. 구성 웹 페이지를 사용하려면 웹 브라우저를 사용하여 `http://<host>:<port>/interact/jsp/WebConnector.jsp`와 유사한 URL을 여십시오.

  관리 웹 페이지에서 수행하는 변경 사항은 Web Connector가 배포된 서버의 `jsconnector.xml` 파일에 저장됩니다.
- 텍스트 편집기 또는 XML 편집기를 사용하여 직접 `jsconnector.xml` 파일을 편집하십시오. 이 방법을 사용하기 전에 XML 태그 및 값을 편집할 수 있는지 확인하십시오.

참고: 언제든 jsconnector.xml 파일을 수동으로 편집하십시오. Web Connector 관리 페이지(http://*<host>*:*<port>*/interact/jsp/jsconnector.jsp에 있음)를 열고 **구성 다시 로드**를 클릭하여 해당 설정을 다시 로드할 수 있습니다.

다음 테이블에서는 jsconnector.xml 파일에 나타날 때 설정할 수 있는 구성 옵션을 설명합니다.

표 30. Web Connector 구성 옵션

매개변수 그룹	매개변수	설명
defaultPageBehavior		
	friendlyName	Web Connector의 웹 구성 페이지에 표시할 URL 패턴에 대한 사용자가 읽을 수 있는 ID입니다.
	interactURL	Interact 런타임 서버의 기본 URL입니다. 참고: Web Connector(jsconnector) 서비스가 배포된 웹 응용 프로그램으로 실행 중인 경우에만 이 매개변수를 설정해야 합니다. WebConnector가 Interact 런타임 서버의 일부로 자동으로 실행 중인 경우에는 이 매개변수를 설정하지 않아도 됩니다.
	jsConnectorURL	다른 사이트로 연결 URL을 생성하는 데 사용되는 기본 URL입니다(예: http://host:port/jsconnector/clickThru).
	interactiveChannel	이 페이지 맵핑을 나타내는 대화식 채널 이름입니다.
	sessionIdCookie	Interact에 대한 API 호출에서 사용되는 세션 ID를 포함하는 쿠키 이름입니다.
	visitorIdCookie	대상 ID를 포함하는 쿠키 이름입니다.
	audienceLevel	인바운드 방문자의 캠페인 대상 수준으로, Interact 런타임에 대한 API 호출에서 사용됩니다.
	audienceIdField	Interact 런타임에 대한 API 호출에서 사용되는 audienceId 필드 이름입니다. **참고:** 참고: 다중 필드 대상 ID는 현재 지원되지 않습니다.
	audienceIdFieldType	Interact 런타임에 대한 API 호출에서 사용되는 대상 ID 필드의 데이터 유형입니다([numeric | string]).
	audienceLevelCookie	대상 수준을 포함하는 쿠키 이름입니다. 선택 사항입니다. 이 매개변수를 설정하지 않으면, 시스템이 audienceLevel에 대해 정의된 것을 사용합니다.
	relyOnExistingSession	Interact 런타임에 대한 API 호출에서 사용됩니다. 일반적으로 이 매개변수는 "true"로 설정됩니다.
	enableInteractAPIDebug	로그 파일에 대한 디버깅 출력을 사용하기 위해 Interact 런타임에 대한 API 호출에서 사용됩니다.
	pageLoadEvents	이 특정 페이지가 로드되면 게시될 이벤트입니다. <event>event1</event>와 유사한 형식으로 이 태그 내에 하나 이상의 이벤트를 지정하십시오.
	interactionPointValues	이 범주의 모든 항목은 IP 특정 범주에 누락된 값에 대한 기본값으로 작용합니다.

표 30. Web Connector 구성 옵션 (계속)

매개변수 그룹	매개변수	설명
	interactionPointValuescontactEvent	이 특정 상호작용 지점에 대해 게시될 컨택 이벤트의 기본 이름입니다.
	interactionPointValuesacceptEvent	이 특정 상호작용 지점에 대해 게시될 수락 이벤트의 기본 이름입니다.
	interactionPointValuesrejectEvent	이 특정 상호작용 지점에 대해 게시될 거부 이벤트의 기본 이름입니다(참고: 이때 이 기능은 사용되지 않음).
	interactionPointValueshtmlSnippet	이 상호작용 지점에 대해 제공될 HTML 템플릿의 기본 이름입니다.
	interactionPointValuesmaxNumberOfOffers	이 상호작용 지점에 대해 Interact에서 검색될 기본 최대 오퍼 수입니다.
	interactionPointValueshtmlElementId	이 상호작용 지점에 대한 컨텐츠를 수신할 HTML 요소의 기본 이름입니다.
	interactionPoints	이 범주는 각 상호작용 지점에 대한 구성을 포함합니다. 누락된 등록 정보의 경우, 시스템은 interactionPointValues 범주 아래에 구성된 값을 사용합니다.
	interactionPointname	상호작용 지점(IP) 이름입니다.
	interactionPointcontactEvent	이 특정 IP에 대해 게시될 컨택 이벤트 이름입니다.
	interactionPointacceptEvent	이 특정 IP에 대해 게시될 수락 이벤트 이름입니다.
	interactionPointrejectEvent	이 특정 IP에 대해 게시될 거부 이벤트 이름입니다(이 기능은 아직 사용되지 않음).
	interactionPointhtmlSnippet	이 IP에 대해 제공될 HTML 템플릿 이름입니다.
	interactionPointmaxNumberOfOffers	이 IP에 대해 Interact에서 검색될 최대 오퍼 수입니다.
	interactionPointhtmlElementId	이 상호작용 지점에 대한 컨텐츠를 수신할 HTML 요소 이름입니다.
	enableDebugMode	특수 디버그 모드를 설정하기 위한 부울 플래그입니다(허용 가능한 값: true 또는 false). 이 매개변수를 true로 설정하면, Web Connector에서 리턴된 컨텐츠는 클라이언트에게 방금 발생한 특정 페이지 맵핑을 알리는 '알림'에 대한 JavaScript 호출을 포함합니다. 알림을 생성하려면 클라이언트의 authorizedDebugClients 파일에 항목이 있어야 합니다.
	authorizedDebugClients	디버그 모드를 규정하는 호스트 이름 또는 IP(Internet Protocol) 주소 목록을 포함하는 특수 디버그 모드에서 사용되는 파일입니다.
	enableRawDataReturn	Web Connector가 컨텐츠 후미에 JSON 형식의 원시 오퍼 데이터를 첨부하는지 여부를 판별하기 위한 부울 플래그입니다(허용 가능한 값: true 또는 false).
	enableNetInsightTagging	Web Connector가 컨텐츠 끝에 NetInsight 태그를 첨부하는지 여부를 판별하기 위한 부울 플래그입니다(허용 가능한 값: true 또는 false).

표 30. Web Connector 구성 옵션 (계속)

매개변수 그룹	매개변수	설명
	apiSequence	pageTag가 호출될 때 Web Connector에 의한 API 호출 시퀀스를 지시하는 APISequence 인터페이스 구현을 표시합니다. 기본적으로 이 구현은 StartSession, pageLoadEvents, getOffers, logContact 시퀀스를 사용합니다(여기서 마지막 두 개는 각 상호작용 지점에 특정함).
	clickThruApiSequence	clickThru가 호출될 때 Web Connector에 의한 API 호출 시퀀스를 지시하는 APISequence 인터페이스 구현을 표시합니다. 기본적으로 이 구현은 StartSession 및 logAccept 시퀀스를 사용합니다.
	netInsightTag	NetInsight 태그 호출을 통합하는 데 사용되는 HTML 및 JavaScript 템플릿을 표시합니다. 일반적으로 이 옵션을 변경해서는 안됩니다.

Web Connector 관리 페이지 사용

Web Connector에는 특정 URL 패턴과 함께 사용할 수 있는 구성을 쉽게 관리하고 테스트할 수 있는 일부 도구를 제공하는 관리 페이지가 있습니다. 또한 관리 페이지를 사용하여 수정한 구성을 다시 로드할 수 있습니다.

관리 페이지 정보

지원되는 웹 브라우저를 사용하여 `http://host:port/interact/jsp/jsconnector.jsp`를 열 수 있습니다. 여기서 *host:port*는 Web Connector가 실행 중인 호스트 이름 및 연결 청취 중인 포트입니다(예: `runtime.example.com:7001`).

다음과 같은 방법으로 관리 페이지를 사용할 수 있습니다.

표 31. Web Connector 관리 페이지 옵션

옵션	용도
구성 다시 로드	디스크에 저장된 구성 변경 사항을 메모리로 다시 로드하려면 **구성 다시 로드** 링크를 클릭하십시오. 구성 웹 페이지를 사용하지 않고 Web Connector jsconnector.xml 구성 파일을 직접 변경한 경우 이렇게 해야 합니다.
구성 보기	**구성 보기**에 입력하는 URL 패턴을 기준으로 WebConnector 구성을 볼 수 있습니다. 페이지 URL을 입력하고 **구성 보기**를 클릭하면, Web Connector는 시스템이 해당 패턴 일치를 기준으로 사용할 구성을 리턴합니다. 일치를 찾을 수 없으면 기본 구성이 리턴됩니다. 이는 올바른 구성이 특정 페이지에 사용되고 있는지 여부를 테스트하는 데 유용합니다.

표 31. Web Connector 관리 페이지 옵션 (계속)

옵션	용도
페이지 태그 실행	이 페이지의 필드를 채우고 **페이지 태그 실행**을 클릭하면 Web Connector가 URL 패턴을 기준으로 pageTag 결과를 리턴합니다. 이는 페이지 태그 호출을 시뮬레이션합니다. 이 도구에서 pageTag를 호출하는 것과 실제 웹 사이트를 사용하는 것 간의 차이점은 이 관리 페이지를 사용하면 오류 또는 예외가 표시된다는 것입니다. 실제 웹 사이트의 경우, 예외는 리턴되지 않고 Web Connector 로그 파일에만 표시됩니다.

Web Connector 페이지 샘플

한 예로, Web Connector의 기능이 몇 개나 페이지에 태그 지정되는지 설명하는 testPage.html 파일이 Interact Web Connector에 포함되었습니다. 편의상 여기에는 해당 샘플 페이지도 표시되어 있습니다.

Web Connector HTML 페이지 샘플

```
<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=us-ascii" />
   <meta http-equiv="CACHE-CONTROL" content="NO-CACHE" />
<script language="javascript" type="text/javascript">
//<![CDATA[
/* ###################################################################
This is a test page that  contains the WebConnector pageTag.  Because the
name of this file has TestPage embedded, the WebConnector will detect a URL
pattern match to the url pattern "testpage" in the default version of the
jsconnector.xml - the configuration definition mapped to that "testpage"
URL pattern will apply here. That means there should this page the
corresponding html element ids that correspond to the IPs for this URL
pattern (ie. 'welcomebanner', 'crosssellcarousel', and 'textservicemessage')
################################################################### */

/* ###################################################################
This section sets the cookies for sessionId and visitorId.
Note that in a real production website, this is done most likely by the login
component. For the sake of testing, it's done here...  the name of the cookie
has to match what's configured in the jsconnector xml.
################################################################### */
     function setCookie(c_name,value,expiredays)
     {
       var exdate=new Date();
       exdate.setDate(exdate.getDate()+expiredays);
       document.cookie=c_name+ "=" +escape(value)+
       ((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
     }
     setCookie("SessionID","123");
     setCookie("CustomerID","1");

   /* ###################################################
   Now set up the html element IDs that correspond to the IPs
   ################################################### */
     document.writeln("<div id='welcomebanner'> This should change, "
  + "otherwise something is wrong <\/div>");
     document.writeln("<div id='crosssellcarousel'> This should change, "
  + "otherwise something is wrong <\/div>");
```

```
     document.writeln("<div id='textservicemessage'> This should change, "
  + "otherwise something is wrong <\/div>");
   //]]&gt;
</script><!--
###############################################################################
this is what is pasted from the pageTag.txt file in the conf directory of
the WebConnector installation... the var unicaWebConnectorBaseURL needs to be
tweaked to conform to your local WebConnector environment
###############################################################################
-->
<!-- BEGIN: Unica Interact Web Connector Page Tag -->
<!-- Copyright 2011, IBM Corporation All rights reserved. -->
<script language="javascript" type="text/javascript">
//<![CDATA[
   var unicaWebConnectorBaseURL = "http://localhost:7001/interact/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: Unica Interact Web Connector Page Tag -->
   <!--
################################################################################
end of pageTag paste
################################################################################
-->
   </body>
</html>
```

부록 E. Interact 및 Intelligent Offer 통합 제품 권장사항

IBM Unica Interact은 IBM Coremetrics Intelligent Offer와 통합하여 Interact 기반 제품 권장사항을 제공할 수 있습니다. 두 제품 모두 오퍼에 대한 제품 권장사항을 제공할 수 있지만, 서로 다른 방법을 사용합니다. Intelligent Offer는 방문자의 웹 사용 기록(협업 필터)을 사용하여 방문자와 권장 오퍼 사이의 상관 관계를 빌드합니다. Interact은 고객의 과거 행동 패턴, 속성, 기록을 바탕으로 하고 보기 수준의 오퍼에는 덜 의존하면서, (고객에 대한 인구통계학적 정보와 기타 정보를 바탕으로) 고객의 행동 프로파일에 어떤 오퍼가 가장 적합할지 학습 방식으로 파악합니다. 오퍼 수락 비율은 자가 학습을 통해 예측 가능한 모델을 빌드하는 데 도움이 됩니다. Interact은 두 제품의 강점만을 바탕으로 개인 프로파일을 사용하여 Intelligent Offer에 범주 ID를 전달하고 선택한 오퍼의 일부로서 방문자에게 표시할 인기도("집단 지성")를 바탕으로 권장 제품을 검색하는 오퍼를 정의할 수 있습니다. 이 솔루션은 고객에게 더 나은 권장사항을 제시하여 홀로 작동하는 제품보다 사용자 클릭 수를 늘리고 더 나은 성과를 낼 수 있습니다.

다음 섹션에서는 통합의 작동 방식과 자체적인 사용자 정의 오퍼 통합을 생성하기 위해 제공되는 샘플 응용 프로그램의 사용 방법을 설명합니다.

Interact과 Intelligent Offer의 통합 개요

이 섹션에서는 프로세스에 대한 설명과 통합이 이루어지는 메커니즘을 포함하여 어떻게 IBM Unica Interact이 IBM Coremetrics Intelligent Offer와 통합하여 Interact 기반 제품 권장사항을 제시할 수 있는지 설명합니다.

IBM Unica Interact은 Intelligent Offer 설치에서 사용할 수 있는 REST(Representational state transfer) API(Application Programming Interface)를 통해 IBM Coremetrics Intelligent Offer와 통합합니다. Interact은 적절한 범주 ID로 REST API를 호출함으로써 권장 제품을 검색하고 방문자가 보고 있는 사용자 정의 페이지에 표시되는 오퍼 정보에 이런 권장 제품을 포함할 수 있습니다.

방문자가 웹 페이지(예: Interact 설치 시 포함되는 샘플 JSP 페이지)의 URL을 볼 때, 해당 페이지는 Interact을 호출하여 오퍼를 페치합니다. 가장 간단한 경우를 예로 들자면, 올바른 매개변수로 Interact 내에서 오퍼를 구성했다고 가정했을 때 다음과 같은 단계가 수행됩니다.

1. 페이지 논리가 방문자의 고객 ID를 식별합니다.
2. Interact에 대한 API 호출이 이루어지면서 그 고객을 위한 오퍼를 생성하는 데 필요한 정보를 전달합니다.

© Copyright IBM Corp. 2001, 2012

3. 반환되는 오퍼는 오퍼 이미지의 URL, 고객이 클릭할 때 연결되는 랜딩 페이지의 URL, 권장할 제품을 결정할 때 사용할 범주 ID라는 최소 세 개의 속성을 가진 웹 페이지를 제공합니다.
4. 그러면 범주 ID를 사용하여 권장 제품 검색을 위한 Intelligent Offer를 호출합니다. 이 제품 세트는 해당 범주에서 가장 잘 팔리는 제품순으로 JSON(JavaScript Object Notation) 형식으로 제공됩니다.
5. 그러면 방문자의 브라우저에 오퍼와 제품이 표시됩니다.

이 통합은 오퍼 권장사항과 제품 권장사항을 함께 결합하는 데 유용합니다. 예를 들어, 한 웹 페이지에 두 개의 상호작용 지점이 있을 수 있습니다. 즉, 하나는 오퍼에 대한 상호작용 지점, 하나는 그 오퍼와 일치하는 권장사항에 대한 상호작용 지점입니다. 이런 목적을 달성하기 위해, 웹 페이지에서는 Interact을 호출하여 실시간 세그먼트를 통해 최선의 오퍼(예컨대, 모든 소형 어플라이언스 10% 할인)를 판별합니다. 해당 페이지가 Interact에서 오퍼를 받을 때, 그 오퍼에는 범주 ID(이 예제에서는 소형 어플라이언스에 대한 범주 ID)가 포함됩니다. 그러면 페이지에서 API 호출을 사용하여 Intelligent Offer로 소형 어플라이언스에 대한 범주 ID를 전달하고, 인기도를 바탕으로 그 범주에 대한 최선의 제품 권장사항을 응답으로 받게 됩니다.

웹 페이지에서 고객 프로파일과 일치하는 범주(예: 고급 식탁용 나이프)만 찾을 목적으로 Interact을 호출하는 더 간단한 예를 들 수도 있습니다. 이때는 받은 범주 ID를 Intelligent Offer로 전달하고 식탁용 나이프 제품 권장사항을 가져옵니다.

통합 필수 요건

Intelligent Offer - Interact 통합을 사용하려면, 우선 이 섹션에 설명되어 있는 필수 요건을 충족하는지 확인해야 합니다.

다음 필수 요건이 충족되어야 합니다.

- *관리자 가이드*와 온라인 도움말에 문서화되어 있는 Interact API 사용법을 숙지하고 있어야 합니다.
- Intelligent Offer 개발자 설명서에 설명되어 있는 Intelligent Offer REST API를 능숙하게 다룰 수 있어야 합니다.
- HTML, JavaScript, CSS 및 JSON(JavaScript Object Notation)에 대한 기초 지식이 있어야 합니다.

  Intelligent Offer REST API는 사용자가 요청하는 제품 정보를 JSON 형식의 데이터로 반환하기 때문에 JSON을 이해하는 것이 중요합니다.
- (JSP가 필수인 것은 아니지만) Interact과 함께 제공되는 데모 응용 프로그램에서 JSP를 사용하기 때문에, 웹 페이지의 서버 측 코딩에 익숙해야 합니다.

- Interact이 제품 권장사항(사용자가 지정하는 범주에서 가장 잘 팔리거나 인기 있는 제품)을 검색하도록 계획하고 있는 범주 ID 목록과 유효한 Intelligent Offer 계정이 있어야 합니다.
- Intelligent Offer REST API 링크(Intelligent Offer 환경을 위한 URL)가 있어야 합니다.

  자세한 정보는 Interact 설치에 한 예로서 포함되는 샘플 응용 프로그램이나 268 페이지의 『통합 샘플 프로젝트 사용』의 샘플 코드를 참조하십시오.

Intelligent Offer 통합을 위한 오퍼 구성

우선 Intelligent Offer로 전달할 필수 정보로 IBM Unica Interact 오퍼를 구성해야 웹 페이지에서 IBM Coremetrics Intelligent Offer를 호출하여 권장 제품을 검색할 수 있습니다.

Intelligent Offer 링크에 대한 오퍼를 설정하려면, 우선 다음 조건이 충족되는지 확인하십시오.

- Interact 런타임 서버가 설정되어 있고 올바르게 실행 중인지 확인하십시오.
- 방화벽이 표준 웹 연결(포트 80)의 발신 연결 설정을 막지 않는지 확인하는 것을 비롯하여, 런타임 서버가 Intelligent Offer 서버와 연결을 설정할 수 있는지 확인하십시오.

Intelligent Offer와의 통합을 위한 오퍼를 설정하려면 다음 단계를 수행하십시오.

1. Interact에 대한 오퍼를 작성하거나 편집하십시오.

   오퍼 생성 및 수정에 관한 자세한 정보는 *IBM Unica Interact 사용자 가이드*와 IBM Unica Campaign 설명서를 참조하십시오.
2. 오퍼의 다른 설정 외에도, 오퍼에 다음 오퍼 속성이 포함되는지 확인하십시오.
   - 오퍼의 이미지를 링크 대상으로 하는 URL입니다.
   - 오퍼의 랜딩 페이지를 링크 대상으로 하는 URL입니다.
   - 이 오퍼와 연결된 Intelligent Offer 범주 ID입니다.

     Intelligent Offer 구성에서 범주 ID를 수동으로 검색할 수 있습니다. Interact은 범주 ID 값을 직접 검색할 수 없습니다.

   Interact 설치 시 함께 포함되는 데모 웹 응용 프로그램에서는 이런 오퍼 속성을 `ImageURL`, `ClickThruURL` 및 `CategoryID`라고 부릅니다. 웹 응용 프로그램이 오퍼에서 예상하고 있는 값과 일치하는 한, 이름은 사용자에게 의미 있는 어떤 이름이든 될 수 있습니다.

예를 들어, 이런 속성을 포함한 "10PercentOff"라는 오퍼를 정의할 수도 있으며, 여기서 범주 ID(Intelligent Offer 구성에서 검색됨)는 PROD1161127, 오퍼 사용자 클릭의 URL은 http://www.example.com/success, 오퍼에 대해 표시할 이미지의 URL은 http://localhost:7001/sampleIO/img/10PercentOffer.jpg입니다(이 경우에는 Interact 런타임 서버에 대해 로컬인 URL).

3. 이 오퍼를 포함할 대화식 채널에 대한 처리 규칙을 정의하고 대화식 채널을 평소대로 배포하십시오.

이제 오퍼는 Intelligent Offer 통합에 필요한 정보로 정의됩니다. Intelligent Offer가 Interact에 대한 제품 권장사항을 제공할 수 있도록 하기 위해 남은 작업은 적절한 API를 호출하도록 웹 페이지를 구성하는 것입니다.

웹 응용 프로그램이 방문자에게 통합 페이지를 서비스하도록 구성할 때, WEB-INF/lib 디렉토리에 다음 파일이 포함되어 있는지 확인하십시오.

- *Interact_Home*/lib/interact_client.jar - 웹 페이지에서 Interact API로의 호출을 처리하는 데 필요합니다.
- *Interact_Home*/lib/JSON4J_Apache.jar - JSON 형식의 데이터를 반환하는 Intelligent Offer REST API에 대한 호출에서 반환되는 데이터를 처리하는 데 필요합니다.

고객에게 오퍼를 서비스하는 방법에 관한 자세한 정보는 『통합 샘플 프로젝트 사용』을 참조하십시오.

통합 샘플 프로젝트 사용

모든 Interact 런타임 설치에는 Intelligent Offer - Interact 통합 프로세스를 보여주는 샘플 데모 프로젝트가 포함됩니다. 샘플 프로젝트는 웹 페이지의 상호작용 지점에 표시하기 위한 권장 제품 목록을 검색하기 위해 Intelligent Offer로 전달되는 범주 ID를 포함한 오퍼를 호출하는 웹 페이지를 생성하는 프로세스를 처음부터 끝까지 완벽하게 보여주는 데모를 제공합니다.

개요

통합 프로세스를 테스트하려면 포함된 샘플 프로젝트를 제공된 상태 그대로 사용하면 되고, 자체적으로 사용자 정의 페이지를 개발하려면 이 샘플 프로젝트를 시작 위치로 삼으면 됩니다. 샘플 프로젝트는 다음 파일에 있습니다.

Interact_home/samples/IntelligentOfferIntegration/MySampleStore.jsp

이 파일에는 통합 프로세스 전체를 보여주는 작업 예제가 있을 뿐 아니라, Interact에서 설정할 사항, .jsp 파일에서 사용자 정의할 사항, 설치 시 실행할 페이지를 올바로 배포하는 방법을 설명하는 광범위한 설명도 포함되어 있습니다.

MySampleStore.jsp

편의상, 여기에는 MySampleStore.jsp 파일이 표시되어 있습니다. Interact의 후속 릴리스에서 이 샘플은 업데이트될 수 있으므로, 설치 프로그램에 함께 포함된 파일을 필요한 예제의 시작 위치로 삼으십시오.

```
<!--
# ****************************************************************************
#  Licensed Materials - Property of IBM
#  Unica 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 IntelligentOffer.
 *
 * 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 IntelligentOffer 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 : IntelligentOffer 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 Intelligent Offers 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 IO 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
 *   Intelligent Offer
 *****************************************************************************/

 /*****************************************************************************
 * Call IntelligentOffer 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());
   }
%>
```

IBM Unica 기술 지원 담당자에게 문의

문서를 참조해도 문제점을 해결할 수 없는 경우, 회사의 지정된 지원 담당자가 IBM Unica 기술 지원 담당자와의 통화를 기록할 수 있습니다. 이 절의 정보를 사용하여 문제점을 효율적으로 해결하십시오.

회사의 지정된 지원 담당자가 아닌 경우에는 IBM Unica 관리자에게 문의하여 정보를 얻을 수 있습니다.

정보 수집

IBM Unica 기술 지원 담당자에게 문의하기 전에 다음 정보를 수집해야 합니다.

- 문제점의 특성에 대한 간단한 설명
- 해당 문제점이 발생할 때 표시되는 자세한 오류 메시지
- 문제점을 재현할 수 있는 자세한 단계
- 관련 로그 파일, 세션 파일, 구성 파일 및 데이터 파일
- "시스템 정보"에서 설명한 방법에 따라 얻을 수 있는 제품 및 시스템 환경에 대한 정보

시스템 정보

IBM Unica 기술 지원 담당자와 통화할 때 환경 정보를 요청하는 경우가 있습니다.

문제점 때문에 로그인이 불가능한 경우 외에는, 설치된 IBM Unica 응용 프로그램에 대한 정보를 제공하는 제품 정보 페이지에서 이러한 정보 대부분을 얻을 수 있습니다.

도움말 > 제품 정보를 선택하여 제품 정보 페이지에 액세스할 수 있습니다. 제품 정보 페이지에 액세스할 수 없는 경우에는 각 응용 프로그램의 설치 디렉토리 아래에 있는 `version.txt` 파일을 사용하여 모든 IBM Unica 응용 프로그램의 버전 번호를 알 수 있습니다.

IBM Unica 기술 지원 담당자에게 문의

IBM Unica 기술 지원 담당자에게 문의하는 방법은 IBM Unica 제품 기술 지원 웹 사이트(http://www.unica.com/about/product-technical-support.htm)를 참조하십시오.

© Copyright IBM Corp. 2001, 2012

주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 명시적 또는 묵시적인 일체의 보증 없이 이 책을 "현상태대로" 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

© Copyright IBM Corp. 2001, 2012

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함)간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등)하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 단계의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 비IBM 제품을 반드시 테스트하지 않았으므로, 이들 제품과 관련된 성능의 정확성, 호환성 또는 기타 주장에 대해서는 확인할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

여기에 나오는 모든 IBM의 가격은 IBM이 제시하는 현 소매가이며 통지 없이 변경될 수 있습니다. 실제 판매가는 다를 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용 없이 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 본 샘플 프로그램은 일체의 보증 없이 "현상태 대로" 제공됩니다. IBM은 귀하의 샘플 프로그램 사용과 관련되는 손해에 대해 책임을 지지 않습니다

이 정보를 소프트카피로 확인하는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

상표

IBM, IBM 로고 및 ibm.com은 전세계 여러 국가에 등록된 IBM Corp.의 상표 또는 등록상표입니다. 기타 제품 또는 서비스 이름은 IBM 또는 타사의 상표입니다. 현재 IBM 상표 목록은 웹의 『저작권 및 상표 정보』(www.ibm.com/legal/copytrade.shtml)에 있습니다.