Oracle® Real-Time Decisions

スタート・ガイド

ORACLE

リリース 2.2　　2007 年 4 月

Oracle Real-Time Decisions スタート・ガイド, リリース 2.2

部品番号: E05254-01

原本名: Oracle Real-Time Decisions Getting Started with Oracle RTD, Version 2.2

制限付権利の説明

このプログラム（ソフトウェアおよびドキュメントを含む）には、オラクル社およびその関連会社に所有権のある情報が含まれています。このプログラムの使用または開示は、オラクル社およびその関連会社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権と工業所有権に関する法律により保護されています。

独立して作成された他のソフトウェアとの互換性を得るために必要な場合、もしくは法律によって規定される場合を除き、このプログラムのリバース・エンジニアリング、逆アセンブル、逆コンパイル等は禁止されています。

このドキュメントの情報は、予告なしに変更される場合があります。オラクル社およびその関連会社は、このドキュメントに誤りが無いことの保証は致し兼ねます。これらのプログラムのライセンス契約で許諾されている場合を除き、プログラムを形式、手段（電子的または機械的）、目的に関係なく、複製または転用することはできません。

このプログラムが米国政府機関、もしくは米国政府機関に代わってこのプログラムをライセンスまたは使用する者に提供される場合は、次の注意が適用されます。

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

このプログラムは、核、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションへの用途を目的としておりません。このプログラムをかかる目的で使用する際、上述のアプリケーションを安全に使用するために、適切な安全装置、バックアップ、冗長性（redundancy）、その他の対策を講じることは使用者の責任となります。万一かかるプログラムの使用に起因して損害が発生いたしましても、オラクル社およびその関連会社は一切責任を負いかねます。

Oracle、JD Edwards、PeopleSoft、Siebel は米国 Oracle Corporation およびその子会社、関連会社の登録商標です。その他の名称は、他社の商標の可能性があります。

このプログラムは、第三者の Web サイトへリンクし、第三者のコンテンツ、製品、サービスへアクセスすることがあります。オラクル社およびその関連会社は第三者の Web サイトで提供されるコンテンツについては、一切の責任を負いかねます。当該コンテンツの利用は、お客様の責任になります。第三者の製品またはサービスを購入する場合は、第三者と直接の取引となります。オラクル社およびその関連会社は、第三者の製品およびサービスの品質、契約の履行（製品またはサービスの提供、保証義務を含む）に関しては責任を負いかねます。また、第三者との取引により損失や損害が発生いたしましても、オラクル社およびその関連会社は一切の責任を負いかねます。

Oracle Real-Time Decisions スタート・ガイド

はじめに

Oracle Real-Time Decisions（Oracle RTD）は、適応型のエンタープライズ・ソフトウェア・ソリューションの開発を可能にします。こうした適応型のソリューションは、ビジネス・プロセス・トランザクションが実行されるにつれて継続的に自己学習し、ルールと予測モデルに従って、各トランザクションをリアルタイムに最適化します。

このドキュメントについて

このドキュメントには、Oracle RTD の使用開始に役立つ情報および例が記載されています。これら例では、読者が Oracle RTD を Windows システムにインストールしていることを前提としています。

対象読者

このドキュメントは、Oracle RTD のテクニカル・ユーザーが、インライン・サービスの構成に使用する機能、用語、ツールおよび方法について理解を深めることを目的としています。

このドキュメントの構成

このドキュメントの内容は次のとおりです。

第1章「Oracle RTD について」では、Oracle RTD の背景について説明しています。

第2章「インライン・サービスの作成」では、Oracle RTD のインライン・サービスを構築する方法について説明しています。

第3章「インライン・サービスに対する負荷のシミュレート」では、Load Generator を使用してシステムのランタイム操作をシミュレートする方法について説明しています。

第4章「コール・センターのインライン・サービスの機能拡張」では、インライン・サービスの機能を拡張します。

第5章「フィードバック・ループのクローズ」では、さらにインライン・サービスを拡張し、予測メソッドを使用する自己学習モデルを追加してインライン・サービスを更新します。

表記規則

規則	説明
固定幅フォント	ソース・コードおよびプログラム出力を示します。
太字	ラベル、タブ、メニューなどのユーザー・インタフェースを示します。
	タスクの実行に役立つ追加情報を示します。

規則	説明
	その項目に関する追加情報を示します。
	データの損失またはエラーが生じる可能性のあるアクションを示します。

1 Oracle RTD について

Oracle RTD は、企業のビジネス運用プロセスにおける重要な局面でのより適切なリアルタイムの意思決定を可能にする、新世代のエンタープライズ分析ソフトウェア・ソリューションを提供します。

Oracle RTD は、フロント・エンド（CRM アプリケーションなど）およびバック・エンド（エンタープライズ・データストアなど）のどちらでも、エンタープライズ・アプリケーションと容易に統合できます。また Oracle RTD には、負荷テストやデバッグに役立つその他のツールも用意されています。

1.1 用語

Oracle RTD は、次の 5 つのコンポーネントで構成されています。

- Decision Studio
- Real-Time Decision Server
- Decision Center
- Administration（JMX）
- Load Generator

インライン・サービスとは、デプロイされる構成済アプリケーションを表します。

インライン・サービスは、Decision Studio を使用して構成およびデプロイされ、Decision Center を使用して分析および更新されます。実行は Real-Time Decision Server で行われます。

インライン・サービスを使用すると、リアルタイムで継続的にデータを収集し、エンタープライズ・ビジネス・プロセスの特性を分析できます。さらに、そのデータと分析を利用した意思決定や、重要なビジネス・プロセスへのフィードバックが可能になります。

要素は、次のオブジェクト・タイプのいずれかになります。

1. アプリケーション: モデルに対するアプリケーション・レベルの設定や、インライン・サービスに必要なパラメータを指定します。
2. パフォーマンス目標: インライン・サービスにおいて追跡および最適化するキー・パフォーマンス・インディケータ（KPI）を指定します。
3. 選択肢: インライン・サービスで提示されるオファー、または自己学習モデルで追跡される属性を表します。
4. ルール: ルールはグラフィカルに構成され、母集団のセグメントを定めたり、適切な選択肢を決定したり、特定の選択肢のスコアリングを行ったりするために使用されます。
5. デシジョン: 適切な選択肢のスコアリングを行って比較検討し、パフォーマンス目標が最適化される 1 つの選択肢を示します。

6. 選択関数: カスタムな方法で選択を実行するために、デシジョンで使用できます。

7. エンティティ: システム内のアクターを表します。

8. データソース: テーブルまたはストアド・プロシージャからデータを取得します。

9. 統合点: インライン・サービスが外部システムに接し、データの入力やアドバイスの出力を実行する場所を表します。統合点には、インフォーマントとアドバイザの 2 つのクラスがあります。インフォーマントは外部システムからデータを受信します。一方、アドバイザは外部システムからデータを受信して、外部システムにアドバイスを返します。

10. モデル: 自己学習する予測モデルによってデシジョンが最適化され、データが分析されます。

11. 統計コレクタ: エンティティに関する統計を追跡する特殊なモデルです。

12. カテゴリ: Decision Center に表示するデータの分割に使用されます。

1.2 Decision Studio について

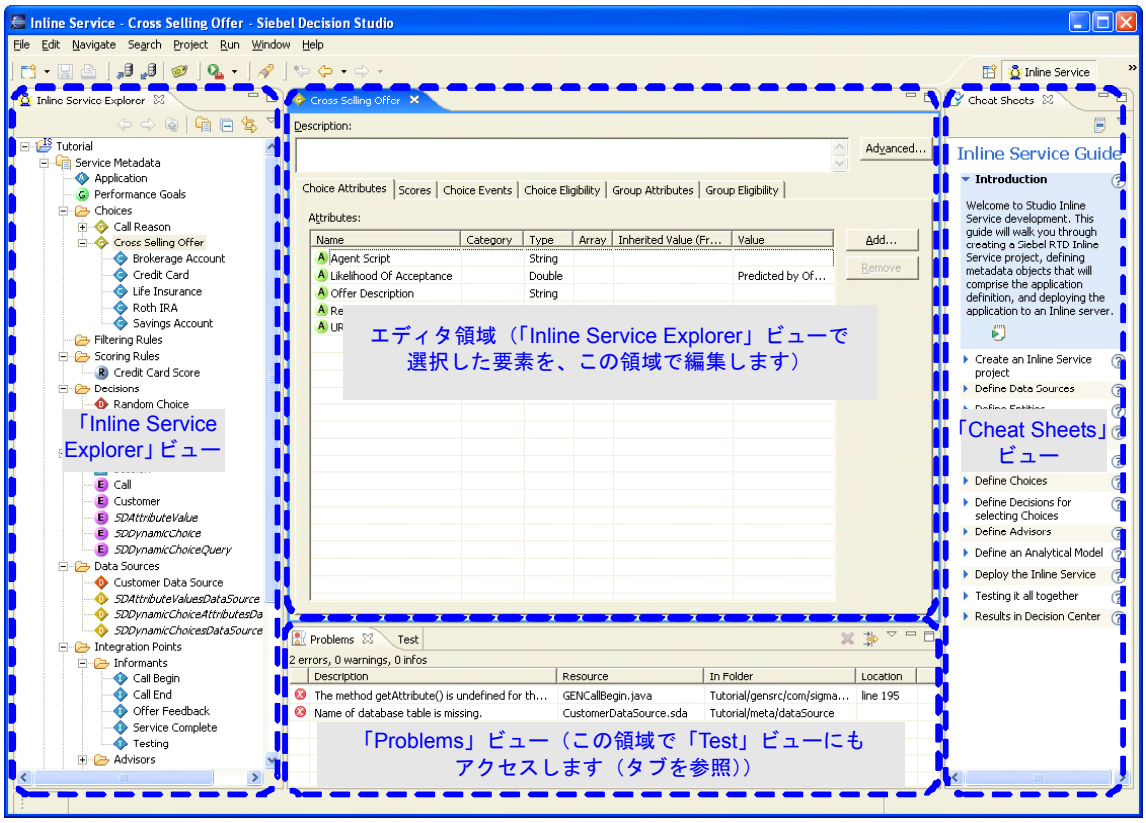
Decision Studio は、インライン・サービスを構成するためのグラフィカルな開発ツールです。インライン・サービスは、アクティビティの監視、統計の収集および提案の作成を可能にします。

Decision Studio は、Eclipse Foundation 社により開発されたオープン・ソースの Java IDE である Eclipse と完全に統合されています。Decision Studio は Eclipse 環境に対する標準プラグインです。Eclipse を使用している場合は、この環境を利用して、追加の開発機能や高度な機能を使用できます。Eclipse に精通していない場合でも、Decision Studio を完全に透過的に使用できます。Eclipse および Decision Studio では、ともにヘルプ・メニューからオンライン・ヘルプを使用できます。

Decision Studio では、いくつかのパースペクティブからインライン・サービスを操作できます。パースペクティブは、そのパースペクティブのビューやエディタの初期設定およびレイアウトを定義します。パースペクティブはそれぞれ、特定のタイプのタスクを実行するための一連の機能を提供し、特定のタイプのリソースを操作します。パースペクティブは、特定のメニューおよびツールバーの表示を制御します。

パースペクティブ（インライン・サービス、Java、リソースなど）を選択したり、別のパースペクティブに変更したりするには、Decision Studio で「**Window**」メニューをクリックし、「**Open Perspective**」を選択して、使用可能なパースペクティブのリストから選択します。最初に Decision Studio を起動したときのデフォルトのパースペクティブは「**Inline Service**」です。このチュートリアルでは、このパースペクティブを使用します。通常、インライン・サービスの開発にはこのパースペクティブを使用します。

デフォルトの「**Inline Service**」パースペクティブには、次の 4 つのビューおよびエディタ領域があります。



1.2.1 「Inline Service Explorer」ビュー

「**Inline Service Explorer**」ビューはプロジェクト構造全体のビューを示します。新しいインライン・サービスを開始すると、選択したインライン・サービスのすべての要素がこのビューに移入されます。

1.2.2 「Problems」ビュー

「**Problems**」ビューには、インライン・サービスを構築する際の検証エラー（.sda）およびコンパイル・エラー（.java）が表示されます。検証エラーをダブルクリックすると、「**Problems**」ビューに、メタデータ/要素エディタがエラー・ポイントで開きます。コンパイル・エラーをダブルクリックすると、「**Problems**」ビューに、生成されたソース・コード（.java ファイル）がエラー・ポイントで開きます。生成されたソース・コード・ファイルは直接編集せず、関連するメタデータまたは要素の問題を修正してください。これにより、ソース・コードが再生成および再コンパイルされます。

1.2.3 「Test」ビュー

「**Test**」ビューでは、構築したとおりに、サーバーに対してインライン・サービスをテストできます。

1.2.4 「Cheat Sheets」ビュー

「**Cheat Sheets**」ビューには、一般的なタスクの手順が表示されます。インストール後、このビューはウィンドウの右側に配置されます。

ヒント:「**Cheat Sheets**」ビューを閉じてエディタ領域を拡大することもできます。このチュートリアルでは「Cheat Sheets」は使用しません。

1.2.5 エディタ領域

「**Inline Service**」パースペクティブの中央の領域がエディタ領域です。ここには、プロジェクト・ツリーで選択したノードに固有のエディタが表示されます。新しいエディタに変更するには、編集する「**Inline Service Explorer**」ビューの要素をダブルクリックします。

1.2.6 ビューの配置とエディタのサイズ変更

エディタのタブには、現在編集のために開かれているリソース名が表示されます。アスタリスク（*）は、変更がまだ保存されていないことを示します。タブには、機能を提供するツールバーがある場合もあります。

パースペクティブのビューおよびエディタは、画面上の任意の領域にドラッグできます。.ビューおよびエディタのサイズは、配置される領域に合せて自動的に変更されます。（主な作業場所である）エディタまたはビューの一部が、他のビューに覆われたり、使用するには不便な領域にサイズ変更される場合があります。エディタまたはビューのサイズを変更するには、開いている別のビューを閉じて残りが自動的にサイズ変更されるようにするか、エディタ・タブをダブルクリックすることによりエディタまたはビューを最大化します。

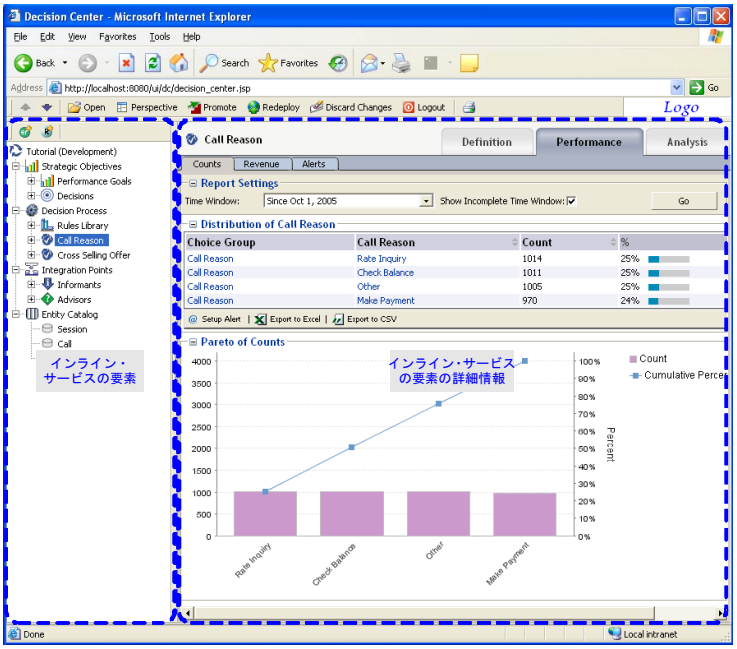
エディタとビューはどちらも、タブをダブルクリックするか、右クリックして表示されるメニュー・アイテムを使用して、最大化と最小化を切り替えることができます。

さらにビューを表示するか、閉じたビューを開くには、Decision Studio の「**Window**」メニューをクリックし、「**Show View**」を選択して使用可能なビューのリストから選択します。

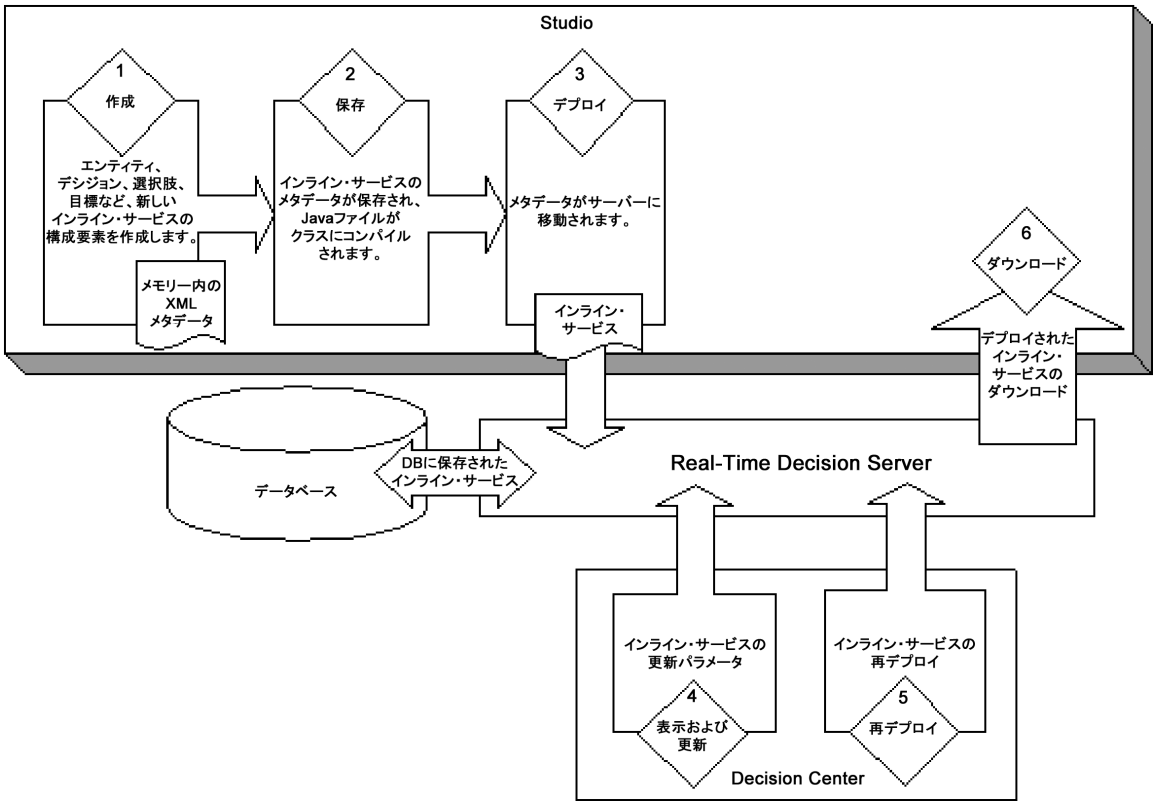
1.3 Decision Center について

Decision Center は、ビジネス・アナリストによる、デプロイされたインライン・サービスの監視および最適化を可能にする Web ベースのアプリケーションです。Decision Center では、意思決定方法の調整だけでなく、モデルから収集された統計の表示や、抱合せ販売などのキャンペーンの微調整を実行できます。

Decision Center のユーザー・インタフェースでは、インライン・サービスを 2 つのペインで表示します。左側のペインにはインライン・サービスの要素のリストが表示され、右側のペインには選択された要素に関連する詳細情報が表示されます。



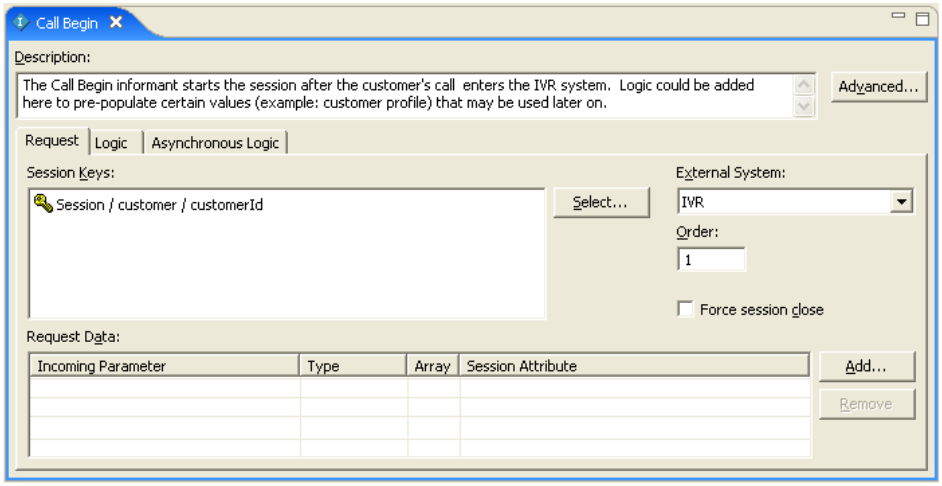
1.4 インライン・サービスのライフサイクルの概要



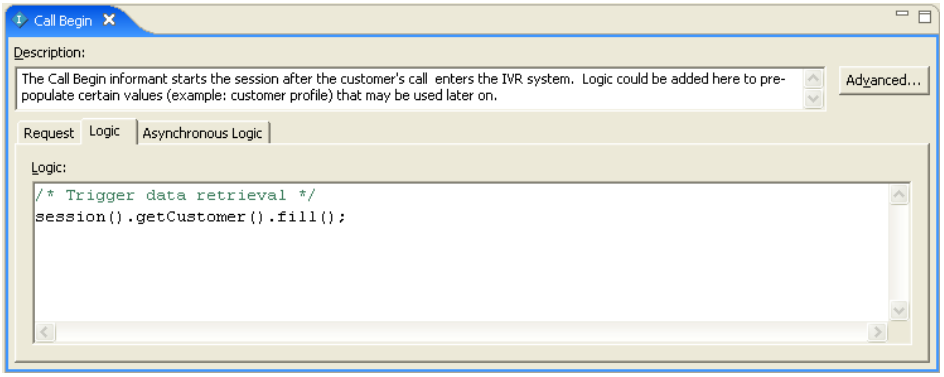
インライン・サービスは Decision Studio を使用して作成されます。次の手順では、インライン・サービスを作成、デプロイおよびダウンロードするプロセス全体の概要を示します。

1. **作成:** Decision Studio を使用して要素を作成および構成します。多くの場合、構成では、ビジネス状況に適用される属性または設定をチェックします。要素の例には、選択肢グループ、パフォーマンス目標、デシジョン、インフォーマント、アドバイザ、エンティティおよびデータソースなどがあります。

   一部の要素では、「Logic」属性および「Asynchronous Logic」属性に Java スクリプトレットを使用できます。例として、次にインフォーマント要素を示します。この要素の名前は Call Begin です。「Description」および「Advanced」ボタンの他に 3 つのタブがあり、それぞれにインフォーマントの属性のセットがあります。



   この Call Begin インフォーマントの「**Logic**」タブで、特定のタスクを実行するオプションの Java コードを記述できます。



   要素が作成され保存されると、オブジェクトを記述する XML メタデータがメモリーに作成されます。

2. **保存**: Decision Studio でインライン・サービスを保存すると、ローカル・ファイル・システム上のインライン・サービス・ディレクトリで、拡張子***.sda** の XML ファイルにメタデータが記述されます。インフォーマント Call Begin のメタデータは `CallBegin.sda` というファイルに保存され、その内容は次のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<sda:RTAPType xmlns:sda="http://www.sigmadynamics.com/schema/sda"
internalName="CallBegin" lastModifiedTime="1133228616435" name="Call
Begin" schemaVersion="20050818" forcesSessionClose="false"
order="1.0">
  <sda:description>The Call Begin Informant starts the session after the
customer's call  enters the IVR system.  Logic could be added here to
pre-populate certain values (example: customer profile) that may be used
later on.</sda:description>
  <sda:system ref="Ivr"/>
  <sda:sessionKey path="customer.customerId" relativeTo="session"/>
  <sda:requestMapper internalName="RequestMapper">
    <sda:entity type="ApplicationSession" var="session"/>
    <sda:dataSource type="RequestDataSource" var="result">
      <sda:arg>
        <sda:path localVarName="session" path="request"
relativeTo="local"/>
      </sda:arg>
    </sda:dataSource>
  </sda:requestMapper>
  <sda:requestData internalName="RequestDataSource">
    <sda:param internalName="message" dataType="object"
objectType="com.sigmadynamics.client.wp.SDRequest"/>
    <sda:request>
      <sda:resultSet/>
    </sda:request>
  </sda:requestData>
  <sda:body>
    <sda:java order="0">/* Trigger data retrieval
*/&#xD;&#xA;session().getCustomer().fill(); </sda:java>
  </sda:body>
  <sda:postOutputBody/>
</sda:RTAPType>
```

Decision Studio で要素に割り当てられた**セッション・キー**および**外部システム**などの属性がここに示されます。Java スクリプトレットも XML ファイルのボディに挿入されていることに注意してください。

インライン・サービスの要素が追加、構成および保存されると、Decision Studio は自動的に必要な Java コードを生成し、これらを Java クラス・ファイルにコンパイルします。2 つのクラスの Java コードが生成されます。最初のセットは、インライン・サービスで使用される基本の Java ファイルです。これらのファイル名は、要素 ID の前に GEN を付けた名前となります。たとえば、CallBegin 要素は `GENCallBegin.java` というファイルを生成します。

2 番目の Java ファイルのセットは、生成されたコードを上書きできるように作成されます。これらのファイル名は要素 ID と同じ名前になります。たとえば、CallBegin 要素は `CallBegin.java` というファイルを生成します。デフォルトでは、Java クラス `CallBegin` は、クラス `GENCallBegin` を単純に拡張します。

インライン・サービスのコンパイル時には、上書きコードを使用する特別な指示がないかぎり、生成されたコードが使用されます。上書きコードを使用するには、上書き Java ファイル（CallBegin.java など）を更新し、次の生成されたソース・ファイルのフォルダ

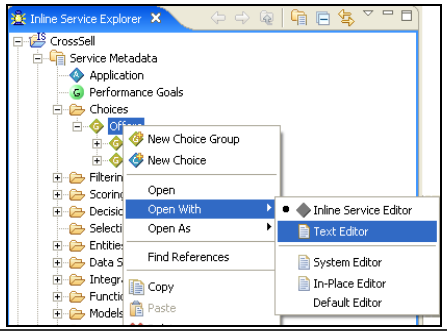
```
[InlineServiceProjectRootFolder]¥gensrc¥com¥sigmadynamics¥sdo¥
```

から、次の上書きソース・ファイルのフォルダに移動します。

```
[InlineServiceProjectRootFolder]¥src¥com¥sigmadynamics¥sdo¥
```

これにより、Decision Studio では、生成された Java ファイルではなく、上書き Java ファイルを使用してコンパイルが実行されます。

ヒント: インライン・サービス・オブジェクトの XML は、Decision Studio の組込みテキスト・エディタで表示できます。「**Inline Service Explorer**」ビューでインライン・サービス・オブジェクトを右クリックし、「**Open With**」メニューから「**Text Editor**」を選択します。通常のエディタ形式に戻すには、「**Inline Service Editor**」オプションを選択します。**注意**: インライン・サービス・オブジェクトを変更する場合は、XML（*.sda）ファイルを直接編集しないでください。変更するには、対応するインライン・サービス・エディタを使用してください。



3. **デプロイ**: インライン・サービスは、Decision Studio を使用して Real-Time Decision Server にデプロイされます。このサーバーの管理サービスでは、メタデータおよびコンパイルされたインライン・サービス・ファイルの受信、インライン・サービスのデータベースへの格納、およびインライン・サービスのメモリへのロードが行われます。これにより、インライン・サービスを利用して、リクエストの処理やレポートの表示などを実行できます。

4. **表示および更新**: レポートおよび学習は、ブラウザベースの Decision Center インタフェースを介して表示されます。インライン・サービスの選択された要素およびパラメータは、Decision Center から更新できます。更新されたインライン・サービスは、再デプロイされるまでランタイムで使用できません。

5. **再デプロイ**: Decision Center でインライン・サービスを更新した場合、Decision Center でインライン・サービスを再デプロイすることによって、この変更を使用できるようになります。管理サービスでは、必要なすべてのメタデータと Java ファイルの再生成、インライン・サービスの再コンパイル、インライン・サービスのデータベースへの格納、およびメモリーへのロードが行われます。

6. **ダウンロード**: Decision Studio を使用して、デプロイされたインライン・サービスをサーバーからダウンロードできます。ダウンロードでは、データベースにあるインライン・サービスをコピーして、メタデータ、Java およびクラス・ファイルをハード・ドライブ上の Decision Studio プロジェクトに配置します。これは、インライン・サービスの元の開発者ではないためにメタデータ・ファイルがない場合に便利です。ビジネス・プロセスで他のユーザーに Decision Center を介したインライン・サービスの変更および再デプロイを許可している場合は、インライン・サービスを最初に Decision Studio で開発およびデプロイしたユーザーであっても、Decision Studio でそのインライン・サービスをさらに変更するには、まずサーバーから最新バージョンをダウンロードする必要があります。

2　インライン・サービスの作成

この章では、オブザーバとして機能するインライン・サービスを構築する方法について説明します。オブザーバ・インライン・サービスは、リアルタイムで継続的に対象プロセスの特性を分析することを目的としています。オブサーバ・インライン・サービスは、ビジネス・ユーザーが、様々なビジネス・イベントおよび時間の経過に伴うビジネス・イベントの変化を分析する際の指針となります。

このチュートリアルのインライン・サービスは、クレジット・カード会社のコール・センターに関するものです。このインライン・サービスでは、顧客およびコール・センターの業務系システムに関するデータを収集し、その通話内容と解決方法についての情報を分析します。

このインライン・サービスの目標は、通話のパターン、通話理由および顧客を分析することです。後述の各項では、このインライン・サービスの機能を拡張して、抱合せ販売のオファーについてCRM システムに提案を行い、さらにその提案の結果についてサービスにフィードバックを加えます。

2.1　チュートリアルの概要

インライン・サービスは Studio グラフィカル開発ツールを使用して作成されます。一般に、インライン・サービスは次のように作成されます。

- Decision Studio でプロジェクトが開始されます。
- 要素がそのプロジェクトに追加され、ビジネス・ニーズに合せて構成されます。
- 操作を実行する特定の要素に、Java スクリプトレットの形式でロジックが追加されます。
- インライン・サービスが、実行される Real-Time Decision Server にデプロイされます。
- インライン・サービスの結果が Decision Center に表示されます。

このチュートリアルでは、次の要素が追加および構成されます。

1. アプリケーション: 必要なアプリケーション・レベルの設定を行い、インライン・サービスのセキュリティを定義します。アプリケーション要素は、各インライン・サービスに対して自動的に作成されます。

2. パフォーマンス目標: スコアリングを使用して最適化されるメトリックで構成される組織目標を表します。たとえば、収益や通話継続時間はパフォーマンス・メトリックです。組織目標は、通話継続時間を最小化して収益を最大化すること、などになります

3. データソース: 外部データソースからのデータのプロバイダとして機能します。データソースのデータの構造および形式は様々に異なります。たとえば、次のような場合があります。

    - RDBMS テーブルの行と列

    - ストアド・プロシージャからの出力値と結果行セット

データソースは、エンティティ要素にマップできるデータのプロバイダで、エンティティ要素にデータを提供します。

たとえば、このチュートリアルでは、データベース内のテーブルに接続するデータソースを追加します。このテーブルには顧客データが含まれます。

4. エンティティ: 1 つ以上のデータソースから構築されるデータの論理的表現です。エンティティは次の目的を果たします。

    - 組織、分析およびモデル化のためのオブジェクトにデータを編成する。
    - 様々なソースのデータの Java コードから、比較的簡単でわかりやすいアクセスを可能にする。
    - データ取得の詳細を隠し、ロジックを変更せずにこれらの詳細を変更できるようにする。
    - データ取得のメカニズムを隠し、ユーザーがデータの取得に使用される API を処理せずに済むようにする。
    - オブジェクトを共有する必要がある場合に、オブジェクトの共有をサポートする。たとえば、サービス・エージェントを表すオブジェクトは複数のセッションで使用される場合があります。

    エンティティの属性にはキー値があります。エンティティ・キーは、エンティティのインスタンスの特定に使用されます。

    たとえば、このチュートリアルでは、顧客を表すエンティティを作成します。そのエンティティの属性は、値のデータソースにマップされます。顧客 ID がその顧客エンティティのキーとして選択されます。

    後で、通話を表すエンティティも作成します。

5. セッション・エンティティ: インライン・サービスの特別なエンティティです。セッション・エンティティは、定義された特定のセッションに固有の属性のコンテナを表します。セッション・キーにより、そのセッションの開始と終了が識別されます。

    セッション・エンティティの属性として設定することにより、定義されたエンティティをセッションに関連付けることができます。セッション属性であるエンティティのみが、そのキーをセッション・キーとしてマーク付けできます。

    たとえば、このチュートリアルでは、顧客エンティティを属性としてセッション・エンティティに追加し、セッション・キーとして顧客キー値である顧客 ID を選択します。

6. インフォーマント: 業務上の相互作用の発生を識別して、データ内の関連する統計パターンを継続的に識別するビジネス・ロジックをトリガーするインライン・サービス内の統合点です。インフォーマントはプロセスを監視しますが、プロセスとのやりとりは行いません。

    このチュートリアルでは、まずテスト用インフォーマントを作成し、その後 CRM システムからすべてのサービス・データを収集するインフォーマントを作成します。

このチュートリアルの後半では、モデルの予測の成功または失敗について、インライン・サービスにフィードバックを行うインフォーマントを作成します。

7. 選択肢グループ: 選択肢グループは選択肢の編成に便利です。選択肢グループは、収集および分析される監視結果の編成方法、またはアドバイザ統合点を経由してビジネス・プロセスに提供するフィードバックの編成方法のいずれかを提供します。

   たとえば、このチュートリアルでは、まず通話理由を編成する選択肢グループを作成します。インライン・サービスを拡張してアドバイザを含める際に、選択肢グループを使用して、サービス・センター・エージェントに提案される抱合せ販売のオファーを編成します。

8. モデル: 組込み分析モデルによる自己学習および自動分析が可能です。モデルは、単にデータを分析する場合にも、ビジネス・プロセスへの提案を行う場合にも使用できます。

   このチュートリアルでは、通話理由を分析するモデルを作成し、その後、顧客に最も受け入れられる可能性のある抱合せ販売のオファーの決定に役立つモデルを作成します。

9. デシジョン: アドバイザによって、適切な選択肢の決定、これらの選択肢の動的なスコアリング、母集団のセグメントに対するスコアリングの比較検討、および最適な選択肢の提示に使用されます。

10. アドバイザ: ビジネス・プロセスに情報を戻すことを可能にすることで、インライン・サービスの機能を拡張します。アドバイザは選択肢グループおよびルールと密接に結び付いています。

    このチュートリアルでは、クレジット・カード・サービス・センターへの通話者に対して行うオファーの選択肢グループを作成します。アドバイザはデシジョンを呼び出し、通話および通話者に関する情報に基づいて、通話者にとって最適なオファーを決定します。アドバイザは、その抱合せ販売の提案をフロント・エンド・アプリケーションに渡します。これによって、コール・センター・エージェントが抱合せ販売を提案できるようになります。

2.2 命名および説明に関する注意

要素の名前および説明は、ビジネス・ユーザーのユーザー・インタフェースである Decision Center で広く使用されます。したがって、要素を作成する際は、時間をかけてわかりやすい要素名を付け、すべての要素に適切な説明を記述することが非常に重要です。

2.2.1 始める前に

始める前に、Real-Time Decision Server が起動されていることを確認します。Real-Time Decision Server の起動方法の詳細は、『Oracle Real-Time Decisions インストレーションおよび管理ガイド』を参照してください。

2.2.2 アプリケーション要素の構成方法

1 *RTD_HOME*¥eclipse¥eclipse.exe を実行して Decision Studio を起動します。Decision Studio が起動したら、「**File**」→「**New**」→「**Inline Service Project**」を選択して新しいプロジェクトを開始します。

注意: このチュートリアルでは、新規インストールを元のプリファレンス設定で使用すると仮定します。すでに Decision Studio または Eclipse を使用している場合は、新しいワークスペースに切り替えることもできます。新しいワークスペースに切り替えるには、「**File**」→「**Switch Workspace**」を選択し、新しいワークスペース・フォルダを選択します。

2 プロジェクト名として Tutorial を指定し、「**Basic Application**」テンプレートを選択します。「**Finish**」をクリックします。インライン・サービスのアップグレードについて尋ねられたら、「**Yes**」を選択します。「**Inline Service Explorer**」にインライン・サービス・プロジェクトが作成されます。デフォルトでは、インライン・サービス・プロジェクトのファイルは、Decision Studio のワークスペースのプロジェクトと同じ名前のフォルダに保存されます（C:¥Documents and Settings¥Win_User¥Oracle RTD Studio ¥Tutorial など）。

3 Studio で、「**Tutorial**」および「**Service Metadata**」フォルダを展開します。「**Application**」要素をダブルクリックし、要素エディタを表示します。要素エディタに、Tutorial インライン・サービスの説明を入力します。

4 「**Permissions**」タブで「**Add**」を使用し、アプリケーション権限にユーザーを追加します。Real-Time Decision Server を選択するには、「**Select Server**」をクリックします（デフォルトでは localhost:8080）。「Connect to a Server」ウィンドウが表示されます。サーバーの「**Host Name**」および「**Port Number**」を入力し、Oracle RTD の管理権限（権限 0）を持つユーザーの「**User Name**」および「**Password**」を入力します。Oracle RTD の認証およびユーザーは、Oracle RTD のインストールおよび構成時に設定されています。詳細は、『Oracle Real-Time Decisions インストレーションおよび管理ガイド』を参照してください。

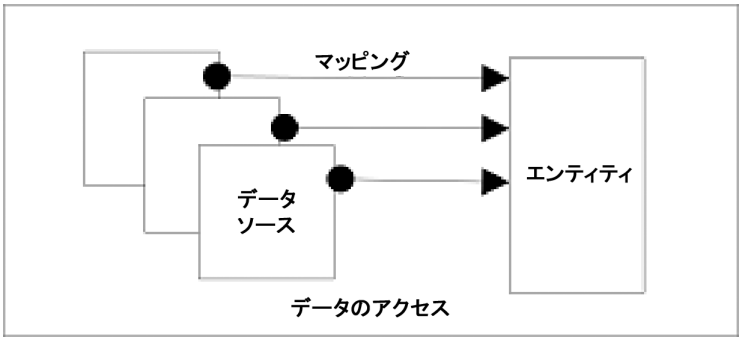
5 「**Connect**」をクリックします。サーバーにログオンしたら、「**Show users**」ボックスを選択し、「**Get Names**」を使用してサーバーのユーザー・リストを表示します。ユーザー名がリスト表示されます。ユーザー名を選択し、「**OK**」をクリックします。

6 ユーザーを選択し、「**Granted**」の下でクリックして権限を付与します。一部の権限には他の権限も含まれます。たとえば、「**Deploy Service from Studio**」を選択すると、自動的に Decision Center の権限が開き、読み込まれ、デプロイされます。すべての権限を選択します。

2.3 データのアクセス

組織データにアクセスするには、次の 2 つの要素を構成します。

- **データソース**: データベース内のデータ構造を表す要素です。
- **エンティティ**: 1 つ以上のデータソースによって移入可能なデータ、またはインフォーマントによって取得されるコンテキスト・データの論理的表現です。



2.4 データソースの作成

1 「Inline Service Explorer」で「Data Sources」を選択して右クリックします。「**New SQL Data Source**」を選択します。データソース名として `Customer Data Source` を指定し、「**OK**」をクリックします。データソース・エディタが表示されます。

2 「**Description**」で、データソースの説明に「`Customer data from a database table`」と入力します。

ヒント: 適切な説明であることが非常に重要です。これらの説明は Decision Center で使用され、ビジネス・ユーザーがレポートおよび分析のコンポーネントを識別するために必要不可欠です。

注意: いくつかの別のデータソースがすでに定義されています。これらはインライン・サービスのフレームワークの一部であり、このチュートリアルでは使用しません。

2.4.1 データソースの出力のインポート

データソースの出力は、データベースから取得された列になります。出力にテーブル内のすべての列が含まれる必要はありません。

1. 「**Import**」をクリックします。「**Import Database Table**」が表示されます。「**Server**」の横に各自のサーバーが表示されます。「**Next**」をクリックし、サーバーに接続します。「**Select Table or View**」が表示されます。
2. 「**Data Source**」で SDDS、「**Table Name**」で `CrossSellCustomers` を選択します。このテーブルは、デフォルトの標準インストールにより作成および移入されています。
3. 「**Finish**」をクリックします。
4. テーブルのすべての列が「**Output**」テーブルにインポートされます。
5. データソースの入力列を設定します。入力は、データ・レコードを取得する際に一致させる列です。ここでは、「**Output**」列テーブルで列名 ID を選択し、右矢印（→）をクリックして、ID を「**Input**」列テーブルに移動できます。データ型は「**Integer**」です。
6. データソースの出力列を設定します。「**Output**」列テーブルで次の列を除くすべてを選択し、「**Remove**」を使用して削除します。

名前	型
Age	Integer
HasCreditProtection	String
Language	String
LastStatementBalance	Double
MaritalStatus	String
NumberOfChildren	Integer
Occupation	String

7. 「**File**」→「**Save All**」を選択して作業を保存します。作業にエラーがある場合は、「**Problems**」ビューに通知が表示されます。

注意: 「**Import**」を使用して、データソースの出力に列名およびデータ型をインポートできます。使用しない列は、「**Remove**」で削除します。

2.5 エンティティの作成

データソースを定義したので、対応するエンティティの定義に進みます。エンティティは、構成内の他の要素によって使用されるオブジェクトです。エンティティは、データソースやインフォーマントなどのデータのソースからの抽象レベルを提供します。1 つのエンティティは、多くのデータソースからのデータ、あるいは計算値を持つことができます。今回は、データソースの構造に直接マップされる簡単なエンティティを作成します。

1 「Inline Service Explorer」で、「**Entities**」グループを探します。右クリックし、メニュー・アイテム「**New Entity**」を選択します。名前として `Customer` を指定し、「**OK**」をクリックします。エンティティ・エディタが表示されます。「**Description**」に「`Customer entity`」と入力します。

注意: オブジェクト ID は、Java のネーミング規則（変数は最初の文字が小文字で大文字と小文字が混合されていること、クラスは最初の文字が大文字で大文字と小文字が混合されていること）に準拠して自動的に作成されます。ラベル名に空白が含まれている場合、オブジェクト ID を作成する際に空白は削除されます。

「Inline Service Explorer」のタスク・バーにある  アイコンを使用すると、オブジェクトのラベルとそのオブジェクト ID を切り替えることができます。

ヒント: エンティティ属性に対して適切な説明を記述することは特に重要です。それぞれのエンティティに適切な説明を追加してください。

2 「**Import**」を使用して、Customer Data Source から属性名とデータ型をインポートします。オプション「**Build data mappings for the selected data source**」は選択したままにします。

3 「**Definition**」タブの「**Default Value**」列で、クリックして挿入ポイントを取得し、各属性のデフォルト値を追加します。String データ型の値は、自動的に二重引用符で囲まれます。

名前	型	デフォルト
age	Integer	35
hasCreditProtection	String	NO
language	String	English
lastStatementBalance	Double	1000

名前	型	デフォルト
maritalStatus	String	Single
numberOfChildren	Integer	0
occupation	String	Student

2.5.1 その他のエンティティ・プロパティについて

エンティティの属性に関するその他の設定も変更できます。たとえば、より複雑なインライン・サービスでは、属性のカテゴリを定義できます。これを定義するには、カテゴリ要素を作成し、属性の「**Properties**」で「**Category**」を使用してこれを割り当てます。属性のプロパティを表示するには、「**Definition**」タブの属性を選択し、右クリックしてメニューから「**Properties**」を選択します。

学習について属性を使用しないように指定することもできます。たとえば、顧客の電話番号がある場合、その番号の分析には意味がありません。そのような場合は、「**Use for Analysis**」を選択解除します。

Decision Center に属性を表示するかどうか制御するには、「**Show in Decision Center**」オプションを使用します。これは、属性が技術的な意味を持つだけで、ビジネス上の直接的な意味または関心を引くような意味がない場合に便利です。

2.5.2 エンティティ・キーの追加

エンティティ・オブジェクトを完全にデータソースにマップするには、エンティティ属性をデータソースのキー値にマップして、マッピングを完了する必要があります。

1. Customer エンティティの「**Definition**」タブで、「**Add Key**」を使用してキー属性を追加します。「**Add Key**」が表示されます。「`customerId`」と入力し、キー値の説明を追加して、データ型を「`Integer`」に変更し、「**OK**」をクリックします。
2. 「**File**」→「**Save All**」を選択して作業を保存します。「**Problems**」ビューにエラーが表示される場合があります。これは、データソースに対する Customer エンティティ属性のマッピング定義が完了していないための予期されるエラーです。次の項に進んでマッピング定義を完了してください。

2.6 セッション・エンティティについて

セッションは、プロセスのユニットのランタイム・データの基礎になります。セッションの期間中、データはメモリーに格納されます。エンティティに関するデータを追跡するには、これを Oracle RTD フレームワークの一部であるセッション・エンティティに関連付けます。エンティティをセッションに関連付けるには、これをセッション・エンティティの属性にします。セッションに対するキーが選択されます。そのキーの固有のインスタンスが検出されると、セッションが開始されます。

たとえば、Oracle RTD で追跡されるコール・センター・プロセスを考えます。セッションには、通話者とエージェントを表すエンティティが含まれます。セッション（この場合は通話）の期間中、これらのエンティティで定義されたデータおよびそれらの相互作用はメモリーに格納され、分析および意思決定に使用できます。

2.6.1 セッション・エンティティへの属性の追加

1 「Inline Service Explorer」で、「**Entities**」の下の「**Session**」をダブルクリックします。

2 「**Definition**」タブで「**Add Attribute**」を使用します。属性名として `customer` を指定し、「**Description**」を入力します。最初のデータ型は String です。これは、次の手順で変更します。

3 「**Data type**」で「**Other**」までスクロールし、これを選択します。「**Type**」選択ダイアログが表示されます。「**Entity Types**」で「`Customer`」を選択します。「**OK**」をクリックします。

2.6.2 セッション・キーの作成

1 「**Session Keys from Dependent Entity**」で「**Select**」をクリックします。

2 ツリーを展開し、セッションに関連付けられたすべてのエンティティのキーを表示します。「**customer**」を展開し、ボックスを選択することにより「**customerId**」をセッション・キーとして選択します。「**OK**」をクリックします。

ヒント: Oracle RTD では、異なるシステムが同一のインライン・サービスにインフォーマントとアドバイザを送信する場合にセッションを追跡できるよう、複数のセッション・キーをサポートしています。このチュートリアルおよび実際の多くのインストールでは、必要なセッション・キーは 1 つのみです。

2.6.3 データソースへのエンティティのマッピング

エンティティ・オブジェクト・エディタで定義したマッピングにより、Customer エンティティを Customer Data Source に関連付けます。Customer エンティティの属性が Customer Data Source からインポートされると、これらの属性が Customer Data Source の出力列に自動的にマップされます。第2.5項を参照してください。ここで、Customer エンティティの Customer Data Source の入力列の値をマップする必要があります。

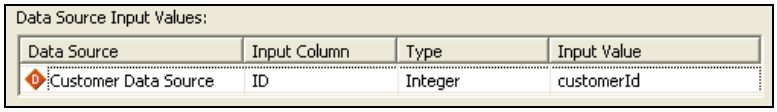
Customer エンティティを開き、「**Mapping**」タブを選択します。エンティティ・エディタは、E アイコン 

で識別されます。

1 インポートを使用したので、Customer Data Source 属性はすでに Customer エンティティ属性にマップされています。属性 Age の場合、「**Source**」の値は `Customer Data Source / Age`（「Show Object ID」アイコンが選択されている場合は `Customer Data Source.Age`）のようになります。インポート後に属性を追加した場合は、「**Source**」の下の  をクリックしてデータソース属性を配置することにより、これらがマップされます。

2 「**Attributes**」テーブルで、各データソース（この場合は、Customer Data Source）の入力列の値を特定する必要があります。データソースの入力列は、レコードを取得する際の識別子（SQL select 文の where 句）です。Customer Data Source では入力列は ID のみであるため、この「**Mapping**」タブでは、「**Attributes**」テーブルにある「**Data Source Input Values**」テーブルにエントリが 1 つのみ表示されます。このエントリの「**Input Value**」セルで…をクリックし、「Edit Value」ダイアログを表示します。

3 このインライン・サービスでは、Customer エンティティのキーを選択します。「**Attribute or Variable**」を選択します。「**Customer**」を展開し、「**customerId**」を選択して「**OK**」をクリックします。



4 「**File**」→「**Save All**」を選択し、インライン・サービスを保存します。

2.7 インフォーマントの作成

インフォーマントは、Real-Time Decision Server にメッセージを送信できる統合点の一種で、プロセス内の特定のユニットに関する情報を含みます。

この段階でインライン・サービスをテストするために、顧客の年齢を表示するテスト・インフォーマントを作成します。実際の命令文を表示して確認するには、インライン・サービスを Real-Time Decision Server にデプロイし、インフォーマントをコールする必要があります。

数字を戻すと、エンティティ、マッピングおよびデータソースが機能していることがわかります。

2.7.1 インフォーマントの追加

1 「Inline Service Explorer」で、「**Integration Points**」に移動し、「**Informants**」を選択します。右クリックし、メニューから「**New Informant**」を選択します。オブジェクト名として `Testing` を指定し、「**OK**」をクリックします。

2 **Testing** インフォーマント・エディタで、「**Description**」に説明を追加します。

3 「Description」の横にある「**Advanced**」をクリックします。「**Show in Decision Center**」を選択解除します。これにより、このインフォーマントが Decision Center のビジネス・ユーザーに対して非表示になります。「**OK**」をクリックします。

2.7.2 インフォーマントへのテスト・ロジックの追加

Testing インフォーマント・エディタで、「**Request**」タブを選択します。インフォーマントは、I アイコン 

で識別されます。

1 セッション・キーを追加するには、「Session Keys」の下の「**Select**」をクリックします。「Customer」から「**customerId**」を選択します。「**OK**」をクリックします。

注意: Decision Studio でエンティティを構成すると、クラスが生成されます。生成されたクラスには、各属性のプロパティ、getter および setter があります。

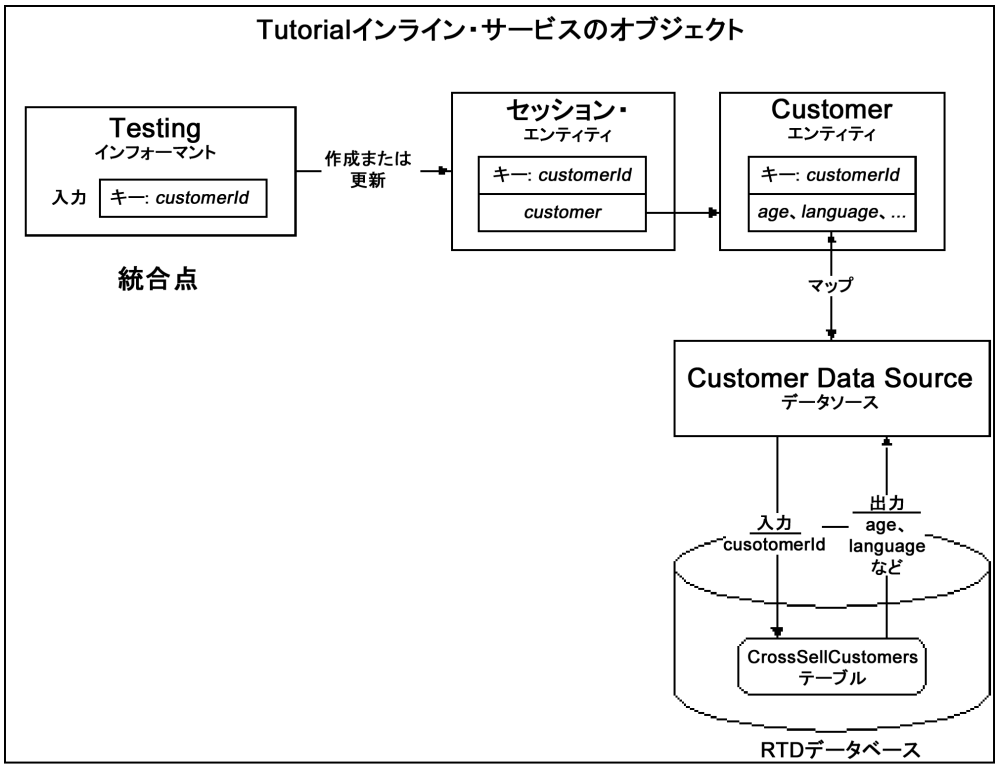
2 「**Logic**」タブを選択します。「**Logic**」の下に次のスクリプトレットを追加します。

```
logInfo("Customer age = " + session().getCustomer().getAge() );
```

logInfo をコールすることにより、情報を「**Test**」ビューの「**Log**」サブタブ、およびサーバーのログ・ファイル（通常は、*RTD_HOME*¥log）に出力できます。セッションを使用して Customer オブジェクトにアクセスし、年齢属性の内容を取得します。

3 これでデプロイの準備が整いました。「**File**」→「**Save All**」を選択し、構成を保存します。

4 次の図は、Testing インフォーマントがコールされてセッションが作成される際、このインフォーマントがどのように顧客データにアクセスするのかを示します。



2.8 インライン・サービスのテスト

インライン・サービスをテストするには、これをデプロイしてテスト・データを持つインフォーマントをコールし、「**Test**」ビューを使用して結果を確認します。インフォーマントはコール元に値を返さないため、結果は「**Test**」ビューの「**Log**」タブに表示されます。

2.8.1 テスト用インライン・サービスのデプロイ

1 タスクバーの「**Deploy**」アイコン  をクリックして、インライン・サービスをデプロイします。

> **注意:** メニュー・アイテム「**Project**」→「**Deploy**」を選択してインライン・サービスをデプロイすることもできます。

2 「**Select**」ボタンを使用して、デプロイするサーバーを選択します。Real-Time Decision Server の場所にデプロイします。これはデフォルトで、インストールのデフォルト構成と同様、localhost になります。ドロップダウン・リストを使用して、デプロイの状態「`Development`」を選択します。「**Terminate Active Sessions (used for testing)**」を選択します。「**Deploy**」をクリックします。

デプロイには 10 秒から数分かかります。デプロイが完了すると、「Inline Service Explorer」の下に「Tutorial deployed successfully」というメッセージが表示されます。

> **注意:** アクティブなセッションを終了するのは、最後にデプロイされたインライン・サービスに対してテストを実行するためです。アクティブなセッションがあり、その同じセッション ID（この場合は customerId）を使用する場合、インフォーマントのテストは、以前にデプロイされたインライン・サービスに対して実行されます。現在アクティブなセッションを終了すると、使用されるセッション ID に関係なく、最後にデプロイされたインライン・サービスに対してテストが確実に実行されます。

3 画面下部の「**Test**」ビューで、テストする統合点として「**Testing**」を選択します。「**customerId**」フィールドに「`7`」と入力します。「**Send**」  をクリックします。

4 「**Test**」ビューの「**Log**」タブを選択し、結果を表示します。`logInfo` コマンドによるすべての出力がタイムスタンプ付きで表示されます。

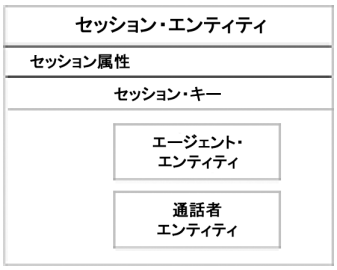
結果は次のようになります。

```
11:53:54,102 Customer age = 38
```

2.9 機能の追加

次に、通話に固有の情報を保持するエンティティを作成します。これは、顧客との相互作用の特性に関するコンテキスト情報です。このエンティティのデータは、インフォーマントから、あるいは計算によって取得され、データベースから取得されることはありません。

最初に通話を表すエンティティを作成し、次に通話からデータを収集するインフォーマントを作成します。通話の分析の対象として選択肢が作成されます。ここでは、通話理由の分析に焦点を当てます。



このエンティティを使用して、通話理由ごとの通話の長さ、これらの通話を行う顧客の特徴など、通話理由に関連するファクタを調査します。インフォーマントにより収集された通話理由を収集および分析するため、自己学習分析モデルが追加され、レポートが Decision Center に表示されます。

2.9.1 通話エンティティの作成

1 「Inline Service Explorer」で、「**Entities**」グループを選択します。右クリックし、メニュー「**New Entity**」を選択します。オブジェクト名として `Call` を指定し、「**OK**」をクリックします。

2 次の表にリストする属性ごとに、次の操作を行います。

- **エンティティ・エディタ**の「**Definition**」タブで、「**Add Attribute**」をクリックします。「**Add Attribute**」が表示されます。表の値を入力し、「**OK**」をクリックします。
- 「**Type**」をクリックします。プルダウンを使用して、属性ごとに適切なデータ型を選択します。

名前	型
agent	String
length	Integer
reason code	Integer

3 「Inline Service Explorer」で、「**Entities**」の下の「**Session**」をダブルクリックします。

4 「**Definition**」タブで「**Add Attribute**」をクリックします。オブジェクト名として `call` を指定します。デフォルトの型は String です。デフォルトの型は次の手順で変更します。

5 「**Data type**」で「**Other**」にスクロールします。「**Other**」から「**Entity types**」を選択し、タイプとして「**Call**」を選択します。call の「**Description**」を追加します。「**OK**」をクリックします。

6 「**File**」→「**Save All**」を選択し、インライン・サービスの変更を保存します。

2.9.2 Call Begin インフォーマントの作成

CRM アプリケーションによってコールされる 3 つのインフォーマント（Call Begin、Service Complete および Call End）を作成します。1 番目の Call Begin はセッションを開始します。このインフォーマントでは、後でより迅速にアクセスできるよう、オプションで特定のセッション属性値を事前にロードおよびキャッシュできます。たとえば、後で使用する顧客のプロファイル情報があり、データベースのコールまたはその他の制約によってこの情報のロードが遅くなることが予測される場合、この顧客のプロファイルを事前にロードしておくことができます。

セッションの属性値は必要に応じて自動的にロードされるため、事前にロードする必要はありません。たとえば、前述の項（2.7.2）の Testing インフォーマントのように顧客の年齢を表示する場合、Real-Time Decision Server は自動的にセッションの Customer エンティティ属性全体を移入し、Age 値を返します。この Tutorial インライン・サービスでは、Call Begin インフォーマントはセッションを開始するだけで、セッションの属性値を事前に移入することはありません。

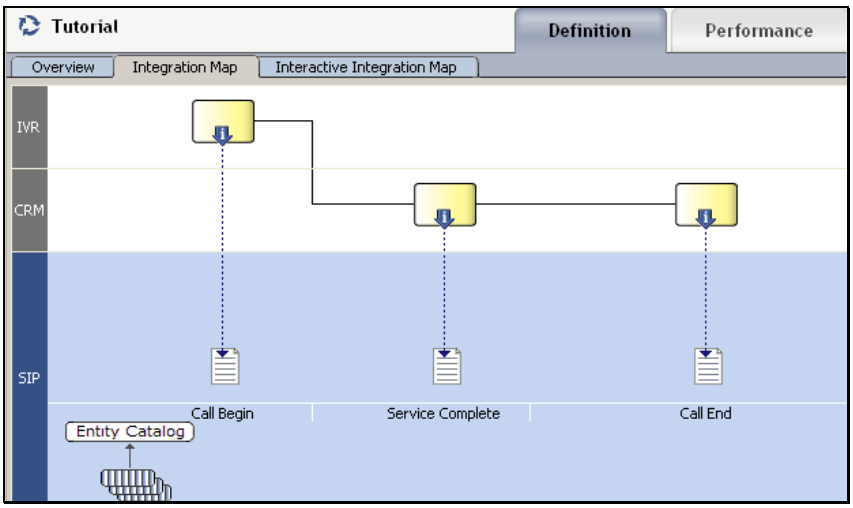
1 「Inline Service Explorer」で、「**Integration Points**」の下の「**External Systems**」グループを選択します。右クリックし、メニュー「**New External System**」を選択します。「**Object Name**」が表示されます。システム名として `IVR` を指定し、「**OK**」をクリックします。要素の説明を入力します。このオブジェクトを保存します。

2 「Inline Service Explorer」で、「**Informants**」グループを選択します。右クリックし、メニュー「**New Informant**」を選択します。「**Object Name**」が表示されます。インフォーマント名として `Call Begin` を指定し、「**OK**」をクリックします。

3 インフォーマント・エディタを使用して、Call Begin の説明を入力します。

4 セッション・キーを Call Begin インフォーマントに追加するには、「**Request**」タブの「**Session Keys**」の横にある「**Select**」をクリックします。「**customerId**」を選択します。「**OK**」をクリックします。

5 さらに「**Request**」タブで、「**External System**」ドロップダウン・リストから「`IVR`」を選択し、「**Order**」ボックスに「`1`」を入力します。「**Force session close**」は選択しないでください。「**External System**」および「**Order**」は、Decision Center の統合マップ（第4.1.7項を参照）の表示レイアウトと順序を決定します。3 つのインフォーマントの定義を完了し、インライン・サービスをデプロイすると、Decision Center の統合マップは次の図のようになります。



6 「**Logic**」タブで、次のコードを入力します。

```
/*

Prepopulate customer data during start of call even though the information may
not be used until much later.  This is not explicitly necessary since the server
will automatically retrieve the information whenever logic in the Inline Service
needs it.

*/

//session().getCustomer().fill();

int CustomerID = session().getCustomer().getCustomerId();

logInfo("Integration Point - CallBegin: Start Session for customerID = " +
CustomerID);
```

7 「**File**」→「**Save All**」を選択し、インライン・サービスの変更を保存します。

2.9.3 Service Complete インフォーマントの作成

2 番目のインフォーマントは、通話を処理したエージェント、通話の長さ、顧客の通話理由など、通話情報についてレポートします。このインフォーマントは、コール・センター・エージェントが顧客のニーズに応えたとき、すなわちサービスが完了したときに CRM アプリケーションによってコールされます。このインフォーマントによって収集されるデータは、Call エンティティに移入されます。

1 「Inline Service Explorer」で、「**Integration Points**」の下の「**External Systems**」グループを選択します。右クリックし、メニュー「**New External System**」を選択します。「**Object Name**」が表示されます。システム名として `CRM` を指定し、「**OK**」をクリックします。要素の説明を入力します。このオブジェクトを保存します。

2 「Inline Service Explorer」で、「**Informants**」グループを選択します。右クリックし、メニュー「**New Informant**」を選択します。「**Object Name**」が表示されます。インフォーマント名として `Service Complete` を指定し、「**OK**」をクリックします。

3 インフォーマント・エディタを使用して、Service Complete の説明を入力します。

4 セッション・キーを Service Complete インフォーマントに追加するには、「**Request**」タブの「**Session Keys**」の横にある「**Select**」をクリックします。「**Customer**」を展開し、「**customerId**」を選択します。「**OK**」をクリックします。

5 さらに「**Request**」タブで、「**External System**」ドロップダウン・リストから「CRM」を選択し、「**Order**」ボックスに「2」を入力します。「**Force session close**」は選択しないでください。

6 インフォーマントにデータを追加するには、次の表にリストされた入力パラメータごとに、次の操作を行います。

- インフォーマント・エディタの「**Request**」タブで、「**Add**」ボタンを使用します。名前を入力し、ドロップダウン・リストを使用してデータ型を選択します。「**OK**」をクリックします。
- 「**Session Attribute**」の下で省略記号  をクリックし、「**Assignment**」を使用します。ドロップダウン・リストから「call」を選択し、さらに入力項目に一致する通話属性を選択します。

入力パラメータ名	型	セッション属性
agent	String	call / agent（「Show Object ID」アイコンが選択されている場合は call.agent）
length	Integer	call / length（「Show Object ID」アイコンが選択されている場合は call.length）
reason code	Integer	call / reason code（「Show Object ID」が選択されている場合は call.reason code）

7 「**Logic**」タブで、次のコードを入力します。

```
logInfo("Integration Point - Service Complete");

logInfo("  Reason Code: " + session().getCall().getReasonCode());

logInfo("  Agent: " + session().getCall().getAgent());

logInfo("  Call Length: " + session().getCall().getLength());
```

8 「**File**」→「**Save All**」を選択し、インライン・サービスの変更を保存します。

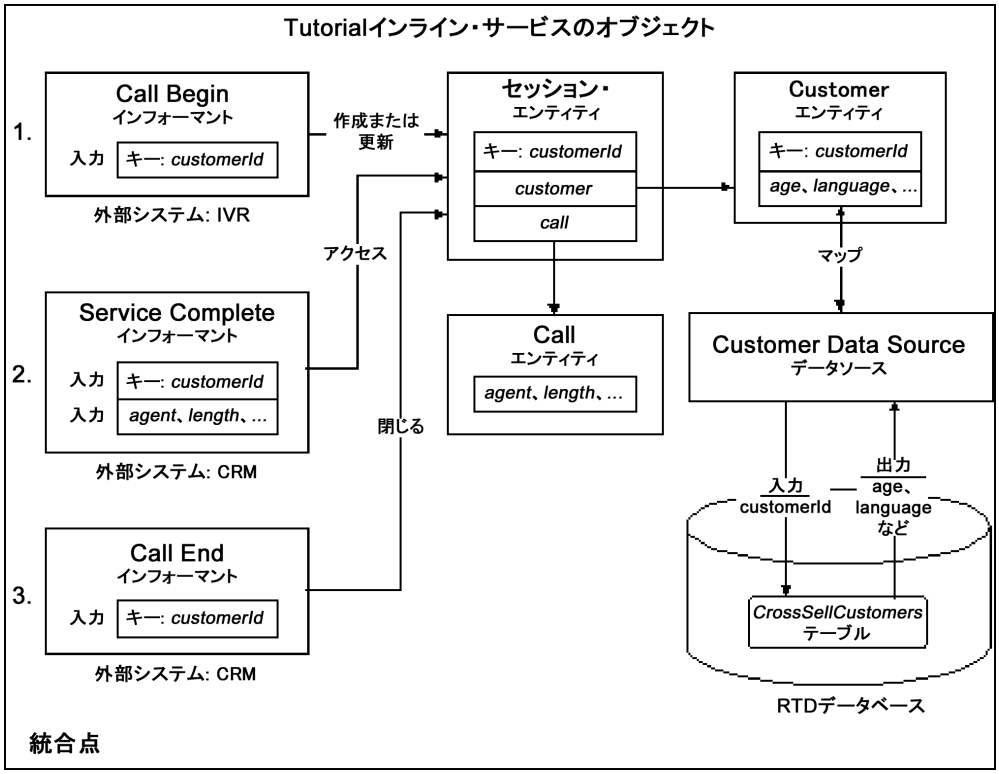
2.9.4 Call End インフォーマントの作成

3 番目のインフォーマントではセッションを閉じます。これは、CRM アプリケーションからコールされる最後のインフォーマントとなります。この Tutorial インライン・サービスでは、このインフォーマントをセッションを閉じるためだけに使用しますが、実際のシステムでは、さらに処理を実行したり、モデルの学習をトリガーする場合もあります。

1. 「Inline Service Explorer」で、「**Informants**」グループを選択します。右クリックし、メニュー「**New Informant**」を選択します。「**Object Name**」が表示されます。インフォーマント名として `Call End` を指定し、「**OK**」をクリックします。
2. インフォーマント・エディタを使用して、Call End の説明を入力します。
3. セッション・キーを Call End インフォーマントに追加するには、「**Request**」タブの「**Session Keys**」の横にある「**Select**」をクリックします。「**Customer**」を展開し、「**customerId**」を選択します。「**OK**」をクリックします。
4. さらに「**Request**」タブで、「**External System**」ドロップダウン・リストから「`CRM`」を選択し、「**Order**」ボックスに「`5`」を入力します。「**Order**」を 5 に設定するのは、このチュートリアルの後半で、さらに 2 つの統合点（アドバイザともう 1 つのインフォーマント）を追加するためです。
5. オプション「**Force session close**」が選択されていることを確認します。このオプションを選択している場合は、インフォーマントがコールされ、そのロジックが処理されると、明示的にそのセッションが閉じられます。明示的にセッションを閉じない場合は、一定の期間が経過した後、自動的に閉じられます。この期間はデフォルトで 30 分で、JConsole を使用して変更できます。
6. 「**Logic**」タブで、次のコードを入力します。

```
logInfo("Integration Point - CallEnd");

logInfo("***************************");
```

7. 「**File**」→「**Save All**」を選択し、インライン・サービスの変更を保存します。
8. 次の図は、3 つのインフォーマントが同じセッションにアクセスし、それを更新する方法を示しています。



2.9.5 インフォーマントのテスト

ここで、作成した 3 つのインフォーマントをコールする簡単なシナリオをテストします。それぞれ、1）コールの開始、2）サービスの完了、3）コールの終了に対応します。「**Test**」ビューを使用してインフォーマントをコールし、インフォーマントのロジック部分に配置したログ・メッセージを表示します。

1 **サーバーにデプロイします。**タスクバーの「Deploy」  を使用して、Tutorial インライン・サービスをデプロイします。「**Terminate Active Sessions (used for testing)**」を必ず選択してください。

2 Decision Studio の下部にある「**Test**」ビューの「**Integration Point**」で「`Call Begin`」を選択します。リクエスト入力「`customerId`」に、たとえば「`7`」などの整数値を入力します。「**Send**」  をクリックし、リクエストをサーバーに送信します。「**Test**」ビュー内の「**Log**」タブに、Call Begin 統合点がコールされたことを示すメッセージが表示されます。

   他の 2 つの統合点（Service Complete および Call End）についても、順に次の表に示す入力値でこの処理を繰り返します。この表には、「**Log**」タブの表示例も示します。

統合点	リクエスト入力	「Log」タブ
Call Begin	customerId: 7	`09:15:41,753 Integration Point - CallBegin: Start Session for customerID = 7`

統合点	リクエスト入力	「Log」タブ
Service Complete	customerId: 7 agent: John length: 21 reason code: 18	09:17:51,845 Integration Point - Service Complete 09:17:51,845 Reason Code: 18 09:17:51,845 Agent: John 09:17:51,845 Call Length: 21
Call End	customerId: 7	09:20:17,342 Integration Point - CallEnd 09:20:17,342 **********

3 インフォーマントは任意の順序でコールできます。セッションの開始に Call Begin インフォーマントは必要なく、セッションの終了に Call End は必要ありません。Service Complete インフォーマントのみをコールした場合、セッションは正常に開始され、応答も同様に返されますが、セッションは開いたままとなります。このチュートリアルで操作する 3 つのインフォーマントは、Real-Time Decision Server に対してコールを実行できる、コール・センター・プロセスの異なる場所を示しています。

ヒント: トラブルシューティング

コンパイルでエラーが生じた場合は、Decision Studio のダイアログにより、「**Problems**」ビューにこれらのエラーが表示されます。エラーをダブルクリックすると、エラーのある要素のエディタが開きます。

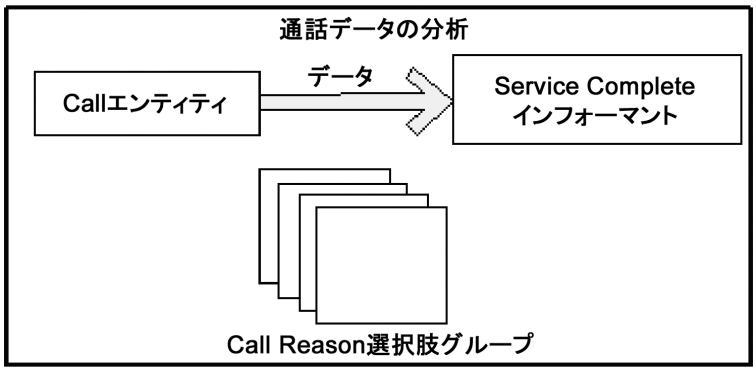
通信しているサーバーが localhost であることを確認してください。Decision Studio では、以前入力した値がドロップダウンに保持されます。そのため、デフォルトが localhost ではない場合があります。

2.10 通話理由の分析

前述の項では 3 つのインフォーマントを作成しましたが、その 2 番目の Service Complete では、通話情報を CRM アプリケーションから Real-Time Decision Server に送信します。通話情報の 1 つは通話理由、すなわち顧客がなぜ通話したのかということです。この項では、選択肢およびモデルを使用して登録された通話理由を分析します。通話理由に関する基本的なレポート（記録された理由の数、それぞれの通話理由とセッション属性の間に相関関係があるのか、それはどのような相関関係か）を表示できるようにすることが目的です。

2.10.1 分析のための選択肢の使用について

選択肢は、分析の対象の作成に使用されます。ここでは最初に通話理由の分析に焦点を当てます。まず、通話理由の選択肢グループを作成します。次に、この選択肢グループ code の属性を定義します。このグループ内で作成される個々の選択肢は、値はそれぞれ異なりますが、この属性定義を継承します。



2.10.2 選択肢グループの追加

1 「Inline Service Explorer」で、「**Service Metadata**」内の「**Choices**」フォルダを選択します。右クリックし、アイテム「**New Choice Group**」を選択します。グループ名として `Call Reason` を指定し、「**OK**」をクリックします。説明を追加します。

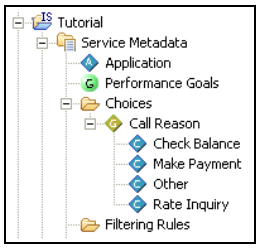
2 Call Reason 選択肢グループ・エディタの「**Choice Attributes**」タブで、「**Attributes**」テーブルの横にある「**Add**」をクリックします。名前として `code` を指定します。データ型に `Integer` を選択します。「**Overridable**」を選択します。「`Choice codes`」という説明を追加します。

> **注意:** グループ属性に対して、これを選択肢属性としました。この 2 つの違いは、選択肢属性では階層の各選択肢に対して値が与えられるのに対し、グループ属性は現在のグループに対してのみ与えられるという点です。

3 グループの下に選択肢を作成するには、「Inline Service Explorer」で選択肢グループ「Call Reason」を右クリックし、「**New Choice**」を選択します。選択肢に「`Check Balance`」を追加します。

「`Make Payment`」、「`Rate Inquiry`」、「`Other`」の各選択肢に対してこれを繰り返します。それぞれの説明を追加します。

4 「Inline Service Explorer」で、「**Choices**」の下の「**Call Reason**」グループを展開し、各選択肢を表示します。



5 4 つの選択肢それぞれについて、次の操作を行います。

「Inline Service Explorer」で、「Choices」を選択します。選択肢エディタの「**Attribute Values**」タブで、属性 code に対して、次の表に示すとおりに「**Attribute Value**」を設定します。

選択肢	属性値
Check Balance	17
Make Payment	18
Other	20
Rate Inquiry	19

6 「**File**」→「**Save All**」を選択し、インライン・サービスの変更を保存します。

2.10.3 分析モデルについて

自己学習分析モデルが作成され、通話理由の自動分析が行われます。このモデルは各通話理由を追跡し、すべてのセッション属性をこれらの結果と関連付けます。Decision Center では、このモデルを使用してレポートを作成し、アラートを送信します。

2.10.4 分析モデルの追加

1 「Inline Service Explorer」で、「**Models**」グループを選択します。右クリックし、メニュー「**New Choice Model**」を選択します。モデル名として `Reason Analysis` を指定し、「**OK**」をクリックします。「**Choice Event Model**」ではなく、「**Choice Model**」を使用してください。

2 「**Use for prediction**」を選択解除します。

3 分析の対象が choice モデル属性であることを示すには、「**Choice**」タブを選択し、「**Choice Group**」ドロップダウン・リストから「`Call Reason`」を選択します。

4 エディタの「**Learn Location**」タブで、「**On Integration Point**」を選択します。

5 「**Select**」を使用して、リストから「Service Complete」を選択します。

6 「**File**」→「**Save All**」を選択し、インライン・サービスの変更を保存します。

2.10.5 選択肢の選択のためのロジックの追加

Service Complete インフォーマントを受け取ったら、該当する通話理由を示す選択肢を選択する必要があります。これを行うには、選択肢モデルのメソッド `addToChoice` を使用して、モデルの選択肢配列に理由を追加します。

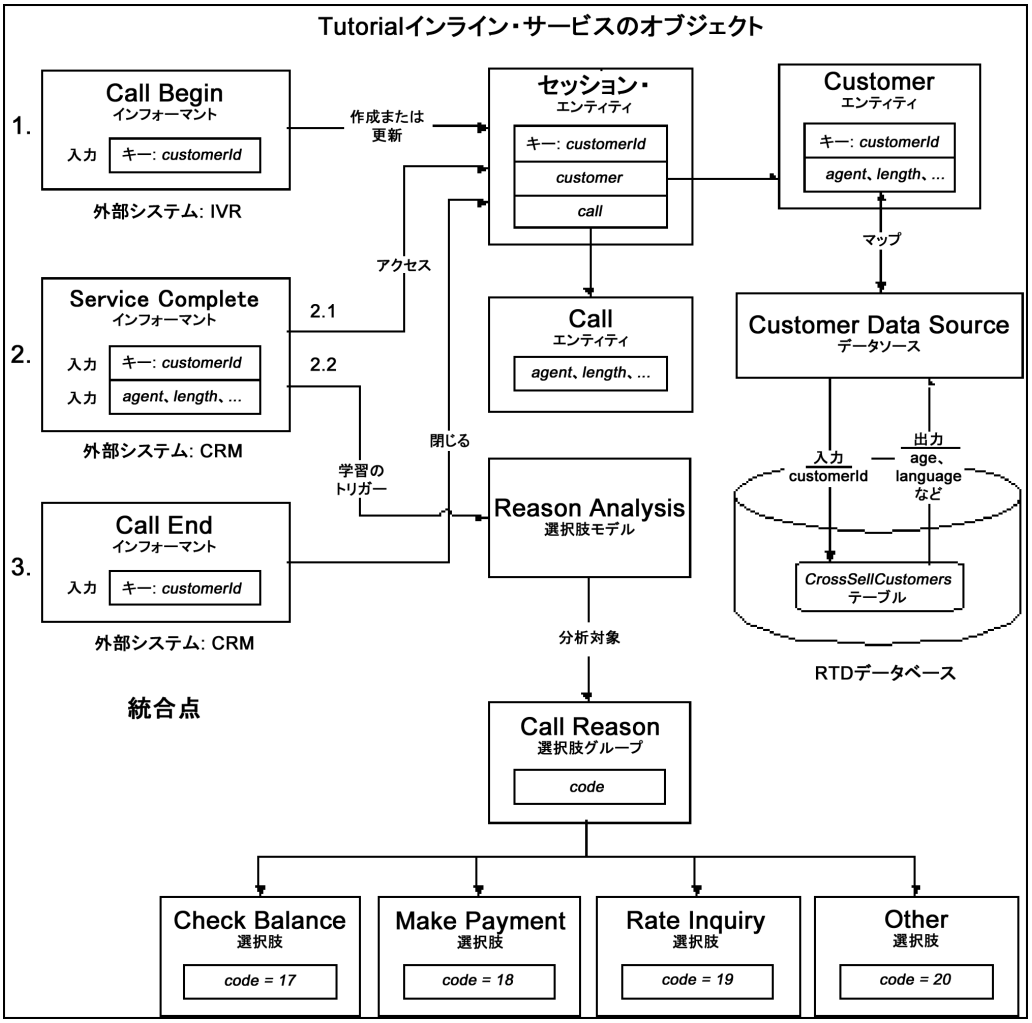
1 「Inline Service Explorer」で、「**Integration Points**」を展開します。「**Informants**」で「Service Complete」インフォーマントを選択します。

2 「**Logic**」タブを選択し、次のロジックを入力します。これにより、通話理由を示す選択肢のオブジェクト ID がモデルに追加されます。

```
logInfo ("Integration Point - Service Complete. ");
logInfo ("  Reason Code: " + session().getCall().getReasonCode());

logInfo ("  Agent: " + session().getCall().getAgent());

logInfo ("  Length: " + session().getCall().getLength());


int code=session().getCall().getReasonCode();
switch (code) {
     case 17:
            ReasonAnalysis.addToChoice("CheckBalance");
            logInfo ("  CheckBalance was added to the model");
        break;
     case 18:
            ReasonAnalysis.addToChoice("MakePayment");
            logInfo ("  MakePayment was added to the model");
            break;
     case 19:
            ReasonAnalysis.addToChoice("RateInquiry");
            logInfo ("  RateInquiry was added to the model");
        break;
      default:
            ReasonAnalysis.addToChoice("Other");
            logInfo ("  Other was added to the model");
            break;
}
```

3 「**File**」→「**Save All**」を選択し、構成を保存します。

4 次の図は、インフォーマント Service Complete がコールされると、モデル Reason Analysis がどのように更新されるかを示しています。



2.10.6 全体のテスト

1 構成をサーバーにデプロイします。デプロイまたはコンパイルにエラーがないことを確認します。

2 「**Test**」ビューを使用して統合点をテストします。「Service Complete」を選択し、各引数の値を customerId=7、agent=John、length=21、reasonCode=18 と設定します。

「**Send**」をクリックします。次のような結果が表示されます。

```
13:57:29,794 Integration Point - Service Complete

13:57:29,794   Reason Code: 18

13:57:29,794   Agent: John

13:57:29,794   Call Length: 21

13:57:29,794   MakePayment was added to the ReasonAnalysis model
```

表示された入力値でインフォーマントがコールされると、そのセッションの属性である Call エンティティに、エージェント、通話長および理由コードについての情報が移入されます。次にインフォーマント・ロジックは、理由コードが 18 なので、選択肢 Make Payment が Reason Analysis モデルに追加されると判断します。つまり、選択肢 Make Payment のカウントが 1 増加します。カウントとともに、モデルはすべてのセッション属性および選択肢との相関関係も追跡します。

値を変更し、数回テストして、その他の理由コードの場合でも正しいオブジェクト ID がモデルに追加されることを確認します。

3　インライン・サービスに対する負荷のシミュレート

この項のチュートリアルでは、Load Generator を使用してシステムのランタイム操作をシミュレートする手順について説明します。一般的に Load Generator は、次の 3 つの目的で使用されます。

1. **パフォーマンスのチェック:** 負荷がかかったときのシステムのパフォーマンスを検証します。レスポンス時間を測定したり、バックエンド・データの入力時などにおいてシステム全体が想定負荷に対応できることを確認します。

2. **初期化:** 重要なデータを使用して、デモ目的で学習機能および統計機能を初期化します。

3. **データ処理:** オフラインのバッチ・ソースを処理します。たとえば、プロジェクトの初期段階に、以前に収集したデータの分析を行います。

3.1　負荷下のパフォーマンス

負荷がかかったときのパフォーマンスを評価するために、インライン・サービスに定義されている統合点（Begin Call、Service Complete および Call End）をコールする負荷シミュレーション・スクリプトを作成します。スクリプトでは、顧客 ID、通話理由コード、エージェント名および通話の長さを毎回変えて、これら 3 つの連続する統合点を複数回コールします。この反復実行ごとに Reason Analysis モデルで学習が行われ、その分析結果を Decision Center レポートにおいて参照できます。

このチュートリアルでは、**Load Generator** を CRM アプリケーションのシミュレーション・プログラムとして扱い、Real-Time Decision Server に対して統合点のコールを反復実行します。作成する Load Generator スクリプト（xml ファイルとして保存）には、このシミュレーションの定義が記述されます。

注意: Load Generator スクリプトを定義するときは、インライン・サービス・オブジェクトへのすべての参照を、ラベルではなく、オブジェクト ID の形式で記述する必要があります。オブジェクト ID を Studio において表示するには、「Inline Service Explorer」タスクバーにあるオブジェクト ID 切替えアイコン  を使用します。たとえば、Service Complete インフォーマントの ID は ServiceComplete で、reason code インフォーマント・パラメータの ID は reasonCode です。他も同様です。

3.1.1　Load Generator スクリプトの作成

1　*RTD_HOME*¥scripts¥loadgen.cmd を実行して、Load Generator を起動します。「**Create a new Load Generator script**」をクリックします。

2　このツールのオンライン・ヘルプや、このチュートリアルに説明されていないパラメータの説明を参照する場合は、[F1]を押します。

3 「**General**」タブを選択し、次のパラメータを入力します。

パラメータ	**説明**	**値**
Client ConfigurationFile	サーバーとの通信に使用するプロトコル、接続先サーバーおよび使用するポートを指定するプロパティ・ファイル。デフォルトでは、HTTPを使用して、ローカル・サーバーとポート8080で通信します。今回の要件には、デフォルト・ファイルが適しています。	*RTD_HOME*¥client¥clientHttpEndPoints.properties
Graph refresh Interval in Seconds	ユーザー・インタフェースにのみ影響します。このパラメータにより、UI上のグラフおよびカウンタのリフレッシュ間隔を指定します。デフォルトは2秒間隔のリフレッシュです。	2
インライン・サービス	前の項で作成したインライン・サービスに割り当てた名前。	Tutorial
Random Number Generator Seed	ランダムな数値の生成に使用するシード。デフォルトは-1です。	-1
Think Time	複数のトランザクション間における時間。セッション指向の負荷シミュレーションでは、必要に応じて異なる値を指定します。このチュートリアルでは、可能なかぎり多くのセッションを送信し、最大スループットを調べます。このパラメータ値には、固定値または範囲値を指定できます。	グローバルな思考時間を指定する固定値
Constant	複数のトランザクション間における思考時間に対する固定の定数。	0
Number of concurrent scripts to run	並行して実行することで、同時にアクティブになるセッションの数。このシミュレーションでは、一度に1セッションのみを実行します。	1
Maximum number of scripts to run	作成されるセッションの合計数。この数値に達した時点で、イベントの送信が停止します。	2000

パラメータ	説明	値
Enable Logging	loadgen カウンタ・ログの有効化と無効化を切り替えるためのチェック・ボックス。このログには、loadgen のパフォーマンス・データ（Load Generator によって送信されたリクエストの数、エラーの数、リクエストの平均レスポンス時間およびピーク時のレスポンス時間など）の履歴が記録されます。このチェック・ボックスが選択されていない場合は、ログ関連の他の 3 つのパラメータ（「Append to Existing File」、「Log File」および「Logging Interval in Seconds」）は無視されます。	選択解除
Append to Existing File	loadgen スクリプトを実行するたびに、既存のログ・ファイルを上書きするか、追加書込みするかを指定するチェック・ボックス。	選択解除
Log File	ascii ファイルへのファイル・パス。Load Generator のログが、タブ区切りの書式で書き込まれる場所です。	*RTD_HOME*¥log¥loadgen.csv
Logging Interval in Seconds	Load Generator のログにのみ影響します。ログへの書込み間隔を指定します。デフォルトは 10 秒です。	10

パス名において *RTD_HOME* は、Real-Time Decision Server がインストールされている場所（たとえば、C:¥OracleBI¥RTD）に置き換えてください。

セッションに関連するパラメータは、実行中に変更できないことに注意してください。正確に言うと変更は可能ですが、Load Generator スクリプトが停止して再起動されるまで、変更は実行に影響しません。

4 構成を保存します。通常、Load Generator スクリプトは、インライン・サービスのプロジェクト・フォルダ内にあるフォルダ（フォルダ名は etc）に保存します。**Tutorial** インライン・サービスがデフォルト・ワークスペースに作成されている場合は、このパスは C:¥Documents and Settings¥Win_User¥Oracle RTD Studio¥Tutorial¥etc のようになります。スクリプト（xml ファイル）に任意の名前を付けます（たとえば、TutorialLoadgen.xml）。

5 統合点に対するパラメータの値を定義するには、「**Variables**」タブをクリックします。「**Variables**」タブにより、統合点の各パラメータ値を、それぞれ異なるソースから取得できます。

注意: すべてのツリーが左側に表示されていない場合があります。すべてのツリーを表示するには、バーをドラッグして、2 つの領域に分割します。

6 「**Script**」を右クリックし、「**Add Variable**」を選択します。名前を `var_customerId` にします。「**Contents**」で、「**Integer Range**」を選択し、1 から 2000 までの順次実行を指定します。この定義によって、各セッションにつき一度処理され、1 から 2000 まで順次実行される変数が作成されます。この例では、最初のセッションが var_customerId = 1 となり、最後のセッションが var_customerId = 2000 となります。「**Script**」を右クリックし、「**Add Variable**」をさらに 3 回選択し、次の 4 つのパラメータを定義します。

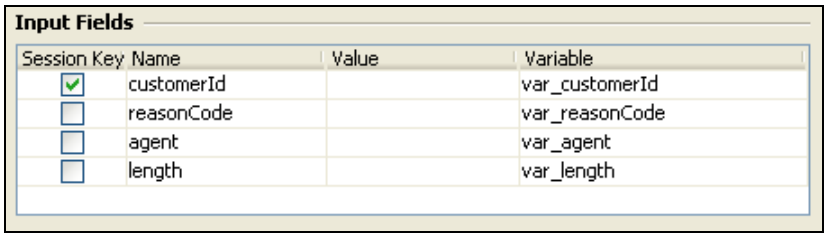
パラメータ	**コンテンツ・タイプ**	**変数値**
var_customerId	整数値の範囲	最小 = 1、最大 = 2000、アクセス・タイプ = 順次
var_reasonCode	整数値の範囲	最小 = 17、最大 = 20、アクセス・タイプ = ランダム
var_agent	文字列配列	文字列を配列に追加するには、テーブル領域を右クリックし、「**Add Item**」を選択します。次に、新しく作成された行を選択（ダブルクリック）してカーソルを取得し、使用する名前を入力します。[Enter]キーを押して、新しい行の値を登録します。エージェント名として、いくつかのサンプル値（John、Peter、Mary、Sara など）を追加します。
var_length	整数値の範囲	最小 = 75、最大 = 567、アクセス・タイプ = 順次 この値は秒単位で、通話の長さとして使用されます。

7 「**Edit Script**」タブを選択し、左側の領域を右クリックして「**Add Action**」を選択します。それぞれが 1 つの統合点に対応する 3 つのアクションを追加します。アクションは適切な順序（つまり CallBegin、ServiceComplete、CallEnd の順）にする必要があります。

8 最初のアクションに対して、タイプを「`Message`」に、**統合点**の名前を `CallBegin` にそれぞれ設定します。「**Input Fields**」で右クリックして「**Add item**」を選択し、入力フィールドを追加します。「**Name**」の下のスペースをダブルクリックして値 `customerId` を入力し、[**Enter**]を押して新しい値を登録します。**customerId** に対応する「Variable」列で、ドロップダウン・リストから「`var_customerId`」を選択します。「**Session Key**」を選択し、このフィールドをセッション・キーとして指定します。

9 「**Edit Script**」タブで、左側の領域を右クリックして「**Add Action**」を選択します。タイプを「`Message`」に、**統合点**の名前を `ServiceComplete` にそれぞれ設定します。「**Input Fields**」で右クリックして「**Add item**」を選択し、入力フィールドを追加します。「**Name**」を `customerId` に、「**Variable**」を `var_customerId` にそれぞれ設定し、「**Session Key**」を選択します。

10 ServiceComplete アクションには、3 つのフィールド（**reasonCode**、**agent** および **length**）を追加し、次の図に示すようにその名前と変数値を設定します。



フィールド名は、Decision Studio に表示される ServiceComplete インフォーマントの入力パラメータ ID と完全に一致する必要があります。Decision Studio の「Inline Service Explorer」タスクバーにある  アイコンを使用すると、表示形式をオブジェクト・ラベルとオブジェクト ID 間で切り替えることができます。

11 「**Edit Script**」タブで、左側の領域を右クリックして「**Add Action**」を選択します。タイプを「Message」に、**統合点**の名前を CallEnd にそれぞれ設定します。「**Input Fields**」で右クリックして「**Add item**」を選択し、入力フィールドを追加します。「**Name**」を customerId に、「**Variable**」を var_customerId にそれぞれ設定し、「**Session Key**」を選択します。

12 再び Load Generator 構成スクリプトを保存します。この時点で、Load Generator スクリプトに、3 つの統合点へのコールが定義されます。「**Edit Script**」タブにおけるアクションの順序が正しいこと（つまり、CallBegin、ServiceComplete、CallEnd の順になっていること）を確認します。この順序でない場合は、各アクションを右クリックして、上下に移動します。スクリプトを再び保存します。

13 「**Run**」タブに移動し、「**Play**」  ボタンを押します。Load Generator においてスクリプトの実行が完了するまで待ちます。

注意:「**Pause**」ボタンと「**Stop**」ボタンがあります。両者の違いは、「**Pause**」では順次実行が記憶され一時停止した箇所から再開するのに対して、「**Stop**」では完全にリセットされる点にあります。

ヒント:

トラブルシューティング

「**Run**」タブの「**Total Errors**」を調べます。この値が 0 よりも大きい場合は、サーバー出力ウィンドウを調べます。問題を示す情報が存在する場合があります。一般的な間違いは次のとおりです。

1. インライン・サービスがデプロイされていない。

2. インライン・サービスや統合点の名前に、スペルの間違いや大文字と小文字の間違いがある。

3. サーバーが実行していない。

「**Total Number of Requests**」の値が 1 から増えていない場合は、loadgen スクリプトの定義に間違いがある可能性があります。次の点を調べてください。

1. 「**Integer Range**」の変数で、最小値が最大値よりも小さいこと。

2. メッセージで送信される変数への値のマッピングが正しいこと。たとえば、変数名を変更した場合は、そのマッピングをやりなおす必要があります。

3. クライアント構成ファイルが正しいこと。

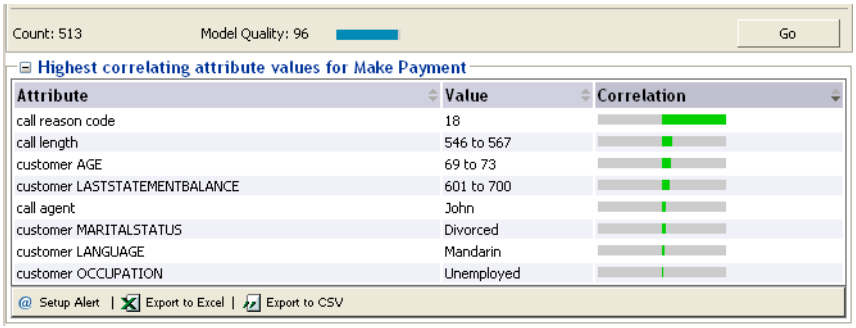
3.1.2 Decision Center における分析結果の表示

Load Generator の実行後に、Decision Center を使用すると、モデルの学習結果を参照できます。

1 Web ブラウザを起動して URL に `http://server_name:8080/ui` を指定して、Decision Center を起動します。管理権限（権限 0）を持つユーザーとしてログインします。

2 「**Open Inline Services**」をクリックします。「Select Inline Service」ウィンドウが表示されます。「**Tutorial**」を選択し、「**Call Reason**」を展開して「**Make Payment**」などの選択肢を 1 つ選択します。右側のペインで、「**Analysis**」タブ→「**Best-fit**」サブタブにナビゲートします。このレポートには、この通話理由が該当した回数と、このコール理由と各属性値間の相関関係に関する要約がそれぞれ記載されます。

3 興味深いデータが表示されます。**call reason code** と **Make Payment** との間に、予想外に強い相関関係があります。



通話理由コード（call reason code）のデータは Load Generator 変数によってランダムに生成されているため、強い相関関係は想定範囲にありません。ただし、ここでは、通話理由コード（ServiceComplete インフォーマントによって送信される）が、この通話理由に大きく関係しています（第2.10.5項に記載されているロジックを参照）。

この誘発的な相関関係を解消するには、この属性がモデルへの入力として使用されないように除外します。モデルから除外する属性の別のタイプとして、顧客の電話番号があります。個々の電話番号と通話理由との間には、特定の相関関係が存在する可能性は低いです。一方、顧客の地域コードと通話理由との間には相関関係が存在する場合があるため、この属性はモデルから除外しません。次の項では、通話理由コードの属性をモデルから除外し、Load Generator スクリプトを再び実行します。

3.1.3 属性の除外

1. Decision Studio で、**Tutorial** プロジェクトを開きます。
2. 「Inline Service Explorer」から「**Service Metadata**」→「**Models**」を開き、「**Reason Analysis**」を選択します。
3. 「**Attributes**」タブに移動します。下側にあるテーブル（テーブルのタイトルは「Excluded Attributes」）で、「**Select**」をクリックし除外する属性を選択します。「**Session**」ノード→「**call**」エンティティを開き、「**reason code**」を選択します。
4. すべてを保存し、ローカル・ホスト・サーバーに再デプロイします。
5. Load Generator スクリプトを再び実行できます。

再実行後に Decision Center を使用してカウントの値を調べると、Load Generator を 2 回実行したときのイベントが含まれています。これは、Load Generator スクリプトを属性除外前に実行してから属性除外後に実行する前にモデルをリセットしていないためです。

3.2 モデルの学習内容のリセット

JConsole 管理ツールを使用すると、モデルの学習内容がリセットされます。

1. OC4J または WebLogic を使用している場合は、*JAVA_HOME*¥bin¥jconsole.exe を実行して JConsole を起動します。WebSphere を使用している場合は、JConsole の構成中に作成したバッチ・スクリプトを実行します。JConsole へのアクセス方法の詳細は、『Oracle Real-Time Decisions インストレーションおよび管理ガイド』を参照してください。
2. 「**Remote**」タブを選択します。インストール中に作成した管理者資格証明と適切なポート番号（通常は 12345）を入力し、「**Connect**」をクリックします。
3. 「**MBean**」タブをクリックしてから、「**OracleRTD**」→「**InlineServiceManager**」→「**Tutorial**」→「**Development**」→「**Loadable**」MBean に移動します。
4. 「**Operations**」タブをクリックしてから、**deleteAllOperationalData()**操作を使用して、このインライン・サービスの操作データをすべて（結果も含めて）削除します。
5. 新しい分析結果を Decision Center で参照するために、Load Generator スクリプトを再び実行します。

3.2.1 インライン・サービスの要約

これまで、単純ですが完全な機能を持つインライン・サービスを作成しました。それには、データ環境、データソースおよび顧客エンティティの定義から始めて、現在の通話データのエンティティを定義しました。基本的な機能をテストした後、分析実行用の統合点とモデルを作成しました。顧客の通話理由を調べて通話理由の発生回数を記録するロジックを、モデルに分析目的で追加しました。次に、Load Generator を使用して、Real-Time Decision Server および **Tutorial** インライン・サービスに対するリクエストをシミュレートしました。最後に、結果を Decision Center に表示しました。

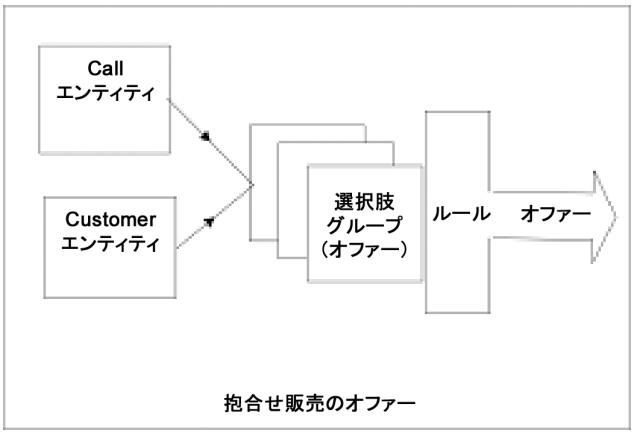
4　コール・センターのインライン・サービスの機能拡張

第 2 項では、クレジット・カード会社のコール・センターにおける通話理由に関連する入力データを追跡して分析するインライン・サービスを作成しました。第 3 項では、Load Generator を使用して、インライン・サービスへのクライアント・リクエスト（インフォーマントのコールを介したリクエスト）をシミュレートしました。

この項では、Tutorial インライン・サービスの機能を拡張し、抱合せ販売のアドバイス機能を CRM アプリケーションに対して追加します。これにより、機能が次のように拡張されます。エージェントによって顧客からの通話が通常の方法（Service Complete インフォーマントのコール）で処理された後に、抱合せ販売を顧客に対して提案する機会を設定します。第 5 項では、抱合せ販売に対する顧客のレスポンスを追跡し、レスポンスから学習した内容を別の顧客への抱合せ販売に反映します。

4.1　選択肢グループの使用と抱合せ販売へのスコアリングについて

サービス・センターに通話を行う顧客に対して提案可能な抱合せ販売の選択肢グループを作成します。コストを最小化するパフォーマンス・メトリックをサポートできるように、選択肢に適用するスコアはコスト・ベースにします。次に、アドバイザを作成して抱合せ販売提案を CRM アプリケーションに渡します。これによって、コール・センター・エージェントが抱合せ販売を提案できるようになります。



4.1.1　選択肢グループによるオファー（抱合せ販売）インベントリの作成

1　「Inline Service Explorer」で、「**Choices**」グループを選択します。右クリックし、「**New Choice Group**」を選択します。グループ名に `Cross Selling Offer` を指定し、「**OK**」をクリックします。

2　「**Choices**」を開き、新しく作成したグループを選択して開きます。説明を追加します。

3 「**Choice Attributes**」タブで、「**Attributes**」テーブルの横にある「**Add**」をクリックします。次の属性を追加し、「**Send to client**」と「**Overridable**」を選択します。

属性名	データ型	Send to client	Overridable
Offer Description	String	選択	選択
URL	String	選択	選択
Agent Script	String	選択	選択

注意: これらの属性はクライアント（コール・センター・エージェント）でのオファーの提案に必要なため、クライアントに送信されます。

属性の値は実際のオファーごとに異なるため、これらの属性は上書き可能である必要があります。抱合せ販売のオファー内容は、この選択肢グループ内の選択肢によって表されます。

実際のインライン・サービスでは、具体的なオファーが提示される前に、レベル分けされた選択肢グループが表示される場合があります。各選択肢グループはオファーの論理グループを表し、グループ内のオファーに一様に適用される属性やビジネス・ルールが存在する場合があります。

4 「Inline Service Explorer」で、「**Choices**」にある「**Cross Selling Offer**」選択肢グループを選択し、次の表に示す属性を持つ 5 つの選択肢を追加します。

各選択肢を追加する手順は次のとおりです。

- 「Inline Service Explorer」で「**Cross Selling Offer**」を右クリックし、「**New Choice**」を選択します。**Credit Card**、**Savings Account**、**Life Insurance**、**Roth IRA** および **Brokerage Account** の 5 つの選択肢を追加します。
- 「Inline Service Explorer」で、「**Choices**」にある「**Cross Selling Offer**」グループを開き、各選択肢を表示します。
- 5 つの各選択肢に対して、次の手順を実行します。
    1. 「Inline Service Explorer」で、「Choices」を選択します。選択肢用のエディタで、説明を追加します。

2. 「**Attribute Values**」タブに、3 つの属性「Agent Script」、「Offer Description」および「URL」が表示されます。「**Attribute Value**」を使用して、次の表に示す属性値を追加します。

選択肢名	**Agent Script**	**Offer Description**	**URL**
Brokerage Account	Would you like to try our new brokerage account?	Brokerage Account offer	http://www.offer.com/offer1.html
Credit Card	Would you like to try our new credit card?	Credit Card offer	http://www.offer.com/offer2.html
Life Insurance	Would you like to try our new life insurance?	Life Insurance offer	http://www.offer.com/offer3.html
Roth IRA	Would you like to try our new Roth IRA?	Roth IRA offer	http://www.offer.com/offer4.html
Savings Account	Would you like to try our new savings account?	Savings Account offer	http://www.offer.com/offer5.html

5 「**File**」→「**Save All**」を選択し、構成を保存します。

4.1.2 パフォーマンス目標の構成

1 「Inline Service Explorer」で、「**Performance Goals**」要素をダブルクリックし、エディタを起動します。「**Add**」ボタンを使用してパフォーマンス・メトリックを追加します。メトリック名を `Cost` にします。「**OK**」をクリックします。

2 「**Optimization**」で、「`Minimize`」を選択し、メトリックを「**Required**」にします。

注意: 複数のパフォーマンス・メトリックがある場合は、「Normalization Factor」を使用して値を正規化する必要があります。たとえば、あるメトリック（Minimize hold time という名前）が秒単位で測定される場合、この正規化ファクタでは、1 ドルの売上を実現するのに要する最小時間が秒単位で測定されます。

3 「**File**」→「**Save All**」を選択し、構成を保存します。

4.1.3 選択肢のスコアリング

各製品には、その維持に必要な平均コスト（年ベース）があります。ドル単位のコストを、その製品のスコアに使用します。

1. 「Inline Service Explorer」で、「**Choices**」にある「**Cross Selling Offer**」選択肢グループを選択し開きます。「**Scores**」タブで、「**Select Metrics**」をクリックし、パフォーマンス・メトリックの「**Cost**」を選択します。これによって、選択肢グループにコスト・スコアが設定されます。実際のスコア値は、選択肢ごとに設定されます。

2. スコア値は定数である必要はありません。多くの場合、顧客のタイプが異なるとスコア値が大きく異なります。こうした相違は、式（つまり、**スコアリング・ルール**）を使用して表現できます。たとえば、クレジット・カード・アカウントの維持に必要なコストは、40 歳以下の顧客では低くなる場合があります。このロジックを**スコアリング・ルール**に定義してから、このルールを Credit Card オファーのコスト・スコアに割り当てます。

3. 「Inline Service Explorer」で、「**Scoring Rules**」フォルダを右クリックし、「**New Scoring Rule**」を選択します。スコアリング・ルールの名前を `Credit Card Score` にします。この新しいスコアリング・ルール用のエディタが起動します。

4. 「**Add conditional value**」ボタン  をクリックし、デフォルト以外のルール条件を設定します。これによって、新しい行が表示されます。その左側のセルに比較式によるルールを指定し、右側のセルにスコア値を指定します。このロジックは次のようになります。左側のセルが真の場合は、右側のセルの値が返されます。それ以外の場合は、2 行目の値が使用されます。ルールの左側をクリックし、省略記号  をクリックします。「**Edit Value**」ダイアログが表示されます。「**Attribute**」を選択してから「**session attributes**」→「**customer**」を開き、「**Age**」を選択してから「**OK**」をクリックします。条件演算子  をクリックし、その右下にある三角形をクリックして、以下を表す記号（**<=**）を選択します。ルールの右側をクリックし、数値の `40` を入力します。Then セルに、数値の `130` を入力します。`2` 行目では、右側のセルを選択し、値として数値の `147` を入力します。入力したルールは次のようになります。

Condition	Value
If All of the following 1. session / customer / AGE <= 40	Then 130.0
Otherwise...	The value is: 147.0

作成した Credit Card Score スコアリング・ルールを保存します。他のオファーについては、定数値をコスト・スコアに設定します。

5 「**Cross Selling Offer**」選択肢グループの選択肢ごとに、「**Scores**」タブを開きます。Cost メトリックの「Score」列に、次の表に示す値を入力します。クレジット・カード選択肢にコスト・スコアを設定するには、「**Score**」列の省略記号をクリックし、**値ソース**として「**Function or rule call**」を選択します。「**Function to Call**」ドロップダウン・リストで、「Credit Card Score」を選択します。

選択肢	コスト・スコア
Brokerage Account	150
Credit Card	Credit Card Score スコアリング・ルール: 40 歳以下は 130、それ以外は 147
Life Insurance	140
Roth IRA	145
Savings Account	135

今回のパフォーマンス目標はコストの最小化にあるため、顧客の年齢が 40 歳以下の場合は Credit Card オファー（スコアは 130）が選択され、それ以外の場合は Savings Account オファー（スコアは 135）が選択されます。このチュートリアルの後述の項では、別のパフォーマンス目標（売上最大化メトリック）を追加し、これらの競合する 2 つのパフォーマンス・メトリックがプラットフォームによって最適化される方法について説明します。

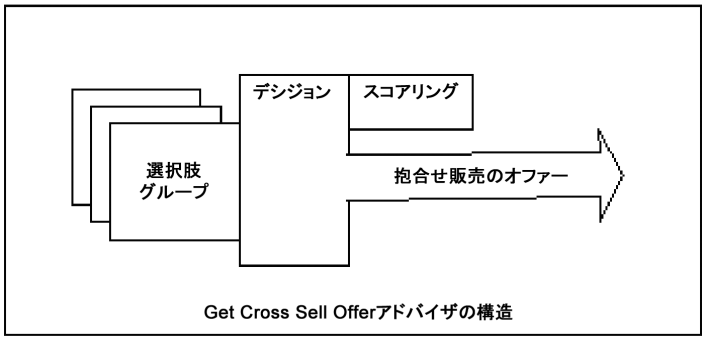
6 「**File**」→「**Save All**」を選択し、構成を保存します。

4.1.4 アドバイザについて

外部システムで意思決定を行う必要があるとき、アドバイザがコールされます。この項では、特定の顧客用に選択されたオファーを CRM アプリケーションに返すアドバイザを作成します。

アドバイザの内部構造には、1 つ以上の選択肢グループに関連付けるデシジョンが含まれます。これらの選択肢グループには、作成されるオファーが含まれます。デシジョンの結果は、アドバイザに送信される結果です。

アドバイザには 2 種類のデシジョンがあり、1 つは通常の処理用で、もう 1 つは制御グループ用です。制御グループは、Oracle RTD によって実現されたパフォーマンス結果を示すベースラインとして機能します。



Get Cross Sell Offerアドバイザの構造

4.1.5 デシジョンの作成

1 「Inline Service Explorer」で、「**Decisions**」グループを選択します。右クリックし、「**New Decision**」を選択します。デシジョン名として `Select Offer` を指定し、「**OK**」をクリックします。

2 Select Offer デシジョンの説明を追加します。「Decision」エディタの「**Selection Criteria**」タブで、「**Select Choices from**」を探します。「**Select**」を使用して「**Cross Selling Offer**」をリストから選択し、「**OK**」をクリックします。

3 制御グループには、オファーをランダムに選択するデシジョンを作成します。新しいデシジョンを作成し、その名前を `Random Choice` にします。

4 Random Choice デシジョンの説明を追加します。「Decision」エディタの「**Selection Criteria**」タブで、「**Select Choices from**」を探します。「**Select**」を使用して「**Cross Selling Offer**」をリストから選択し、「**OK**」をクリックします。

5 「**Select at random**」チェック・ボックスを選択します。

> **注意:** 制御グループは、予測モデルの結果を既存のビジネス・プロセスと比較可能にするベースラインとして機能します。制御グループのデシジョンは、Oracle RTD がインストールされていない場合にデシジョンが適切に反映されるように、適切に定義することが重要です。たとえば、コール・センター用の抱合せ販売アプリケーションで、Oracle RTD の導入前にランダムにオファーが選択されていた場合は、制御グループのデシジョンでランダムな選択が返されるように定義する必要があります。

6 「**File**」→「**Save All**」を選択し、構成を保存します。

4.1.6 アドバイザの作成

1 「Inline Service Explorer」で、「**Integration Points**」の「**Advisors**」グループを選択します。右クリックし、「**New Advisor**」を選択します。アドバイザ名に `Get Cross Sell Offer` を指定し、「**OK**」をクリックします。

2 セッション・キーを Get Cross Sell Offer アドバイザに追加するには、エディタで「**Session Keys**」の「**Select**」を使用して、「**Customer**」から「**customerId**」を選択します。「**OK**」をクリックします。

3 「**External System**」で「**CRM**」を選択します。「**Order**」に「3」を入力します。

Service Complete インフォーマントの「**Order**」を 2 に設定したことを思い出してください。Get Cross Sell Offer アドバイザはこのインフォーマントの後にコールされるため、順序は 3 になります。この「**Order**」は、アプリケーション・プロセスを図示する目的で Decision Center の統合マップでのみ使用され、統合点を特定の順序で実行するものでないことに注意してください。

4 「**Response**」タブで、通常処理と制御グループの両方に対して**デシジョン**を選択します。「**Decision**」には `Select Offer` デシジョンを選択し、「**Control Group Decision**」には `Random Choice` デシジョンを選択します。

5 「**Default Choices**」セクションで、「**Select**」を使用してリストから「**Life Insurance**」を選択し、「**OK**」をクリックします。これによって、選択されたオファーがこのアドバイザのデフォルト・レスポンスになります。このデフォルトは、処理に問題（タイムアウトなど）が発生したときに使用されます。

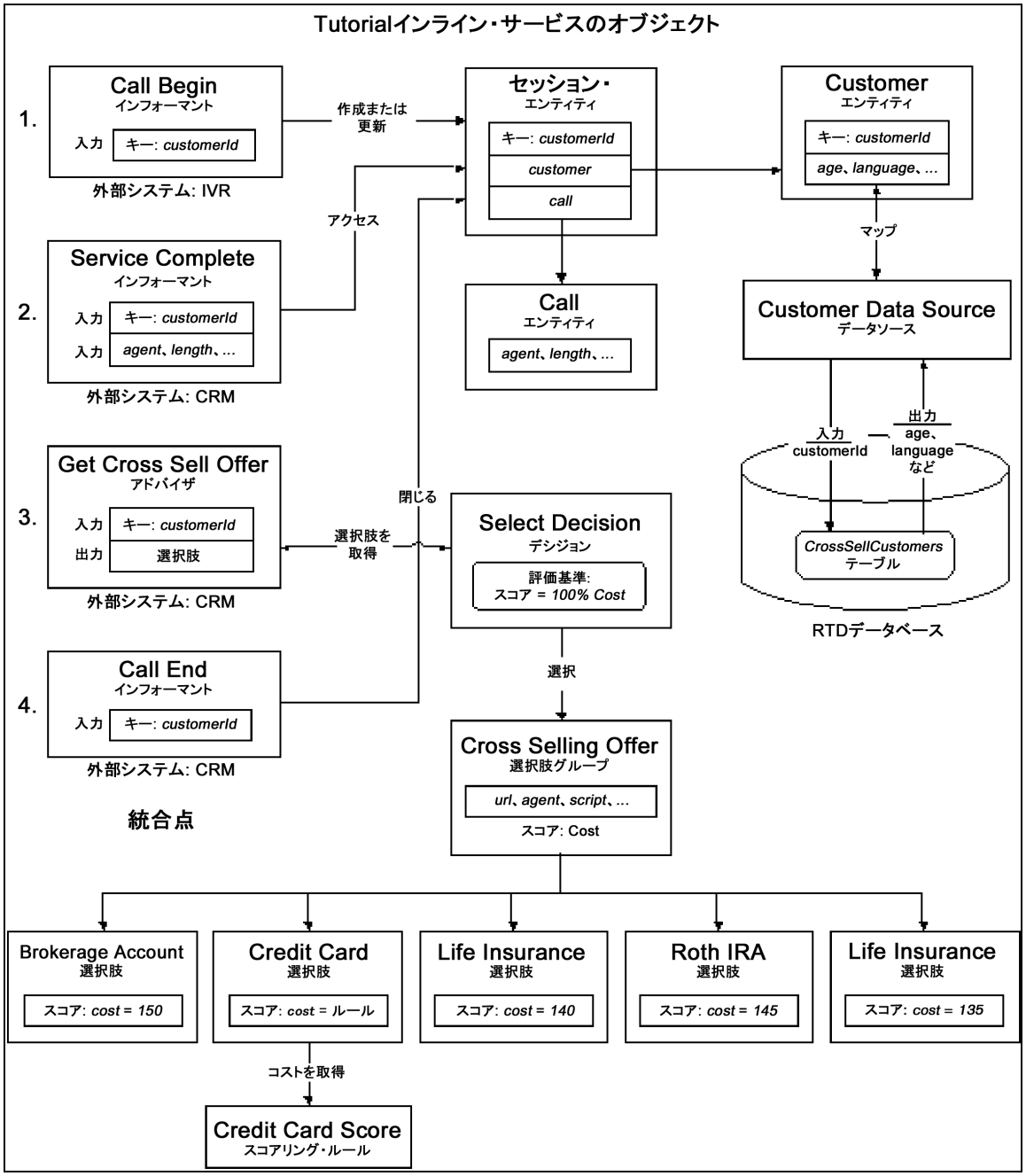
6 「**Asynchronous Logic**」タブで、次のコードを入力します。

```
logInfo("Integration Point - Get Cross Sell Offer");

logInfo("  Customer age = " + session().getCustomer().getAge() );
// 'choices' is array returned by the 'Select Offer' decision

if (choices.size() > 0) {

  //Get the first offer from array
  Choice offer = choices.get(0);

  logInfo("  Offer presented: '" + offer.getSDOLabel() + "'");

}
```

コードを「**Logic**」タブに入力すると、返されるオファーがデシジョンによって決定される前にコードが実行されるため、返されるオファー名を出力できなくなります。前述のコードでは、顧客の年齢と提示されたオファー名が出力されます。コストの最小化を目標としているため、顧客の年齢に基づいて、Savings Account オファーおよび Credit Card オファーのみが提示されることを思い出してください。

7 インライン・サービスを保存します。「**Deploy**」ボタンをクリックします。「**Terminate Active Sessions (used for testing)**」を選択して、その時点でアクティブなすべてのセッションを終了し、デプロイします。

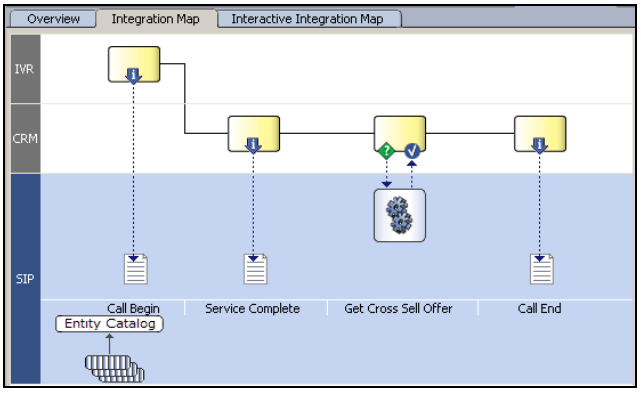
8 次の図に、パフォーマンス目標の Cost に基づいて Get Cross Sell Offer アドバイザが Cross Selling Offer 選択肢グループからオファーを取得するフローを示します。



4.1.7 統合マップの表示

1. Web ブラウザを起動して URL に `http://server_name:8080/ui` を指定して、Decision Center を起動します。インストール時に作成したデフォルト管理者資格証明を使用してログインします。Decision Center を実行するには、Real-Time Decision Server が起動されている必要があります。
2. 「**Open an Inline Service**」をクリックします。
3. **Tutorial** インライン・サービスを選択します。

4 左側のツリーで、ルート・ノードの「**Tutorial**」をクリックします。右側のペインで、「**Definition**」タブの「**Integration Map**」サブタブをクリックして表示します。次のようなマップが表示されます。



5 統合マップでは次の記号を使用して、統合点、データ処理、エンティティおよび情報フローが示されます。

記号	意味
	Real-Time Decision Server でのデータ処理
	アドバイザ・コール
	Real-Time Decision Server に渡す情報（データ）
	インフォーマント・コール

4.1.8 アドバイザのテスト

1 Decision Studio で、「**Test**」ビューを使用して、統合点リクエストを送信します。Service Complete インフォーマントを選択し、パラメータ値を入力します。たとえば、顧客 ID に 7、エージェント名に John、長さに 21、理由コードに 18（または 17、19、20）をそれぞれ入力します。

2 「**Send**」  をクリックし、「**Log**」サブタブでメッセージが送信されたことを確認します。このインフォーマント・コールによって、新しいセッションが顧客 ID に基づいて作成され、顧客の通話理由、エージェント名およびコールの長さが登録されます。

3 **Get Cross Sell Offer** アドバイザを選択します。同一セッションで作業を続行するため、「**customerId**」はそのままにします。「**Send**」をクリックします。

選択されたオファーとその属性が返され、「**Test**」ビューの「**Response**」ペインに表示されます。

「**Test**」ビューの「**Log**」サブタブで顧客 ID が 7 のエントリには、次のようなレスポンスが記録されます。

```
00:24:40,764 Integration Point - Get Cross Sell Offer

00:24:40,764   Customer age = 38

00:24:40,764   Offer presented: 'Credit Card'
```

4 「**customerId**」と他のパラメータで値をそれぞれ異なる値に変えて、手順 1 から 3 までを繰り返します。顧客の年齢が 40 歳以下のときは Credit Card オファーが返され、40 歳を超えるときは Savings Account オファーが返されることを確認してください。このような結果となる理由は、アドバイザが Cost パフォーマンス・メトリックに対してのみ最適化されており、最小コストのオファーが顧客の年齢に応じて Savings Account または Credit Card のどちらかになるためです（第4.1.3項の Credit Card Score スコアリング・ルールを参照）。

5 「**Test**」ビューの「**Trace**」サブタブには、オファーが選択されるまでの一連のプロセス（つまり、各オファーのスコアが計算され最適なオファーが決定されるプロセスから、最終的にパフォーマンス目標（コストの最小化）を満足したオファーが選択されるまでのプロセス）の説明が記録されます。

5　フィードバック・ループのクローズ

前の項では、コール・センター・エージェントが顧客にオファーを提示できるように、オファーを CRM アプリケーションに返すアドバイザを追加しました。顧客にオファーが提示された後は、顧客がそのオファーを受け入れたかどうかを追跡してから、オファーの提示と受入れのプロセスに関するループをクローズする必要があります。フィードバックのループは、様々な方法によって様々なタイミングでクローズされる場合があります。デシジョンやオファーが実行されてから数日から数週間後に結果が判明することもあります。それだけの時間が経過した後でも、肯定的な結果のみが判明し、否定的な結果は不明な場合が多くあります。フィードバックの直接的な取得元は顧客だけでなく、通話を処理するエージェントに加えて、サービス、契約および請求を処理する業務システムやバッチ・プロセスに及ぶ場合があります。

インライン・サービスに対するフィードバック・ループは、インフォーマントの使用を介して Real-Time Decision Server に通知することでクローズされます。

5.1　成功を追跡するイベントの使用について

一般的にオファーの存続期間において、成功を追跡する観点から見て有意義な**イベント**が複数あります。たとえば、Credit Card オファーの存続期間では、次のようなイベントが存在する場合があります。

- オファーの提示
- 顧客による興味関心
- カード発行の申込み
- カードの受領
- カードの使用

これらのイベントは、顧客がクレジット・カードを使用したときが本当の成功とみなす考えに基づいています。最終的な目標は、顧客による興味関心、カード発行の申込みおよびカードの受領だけではなく、多くの顧客がカードを使用することにあります。カードが使用されることで企業に売上がもたらされるからです。

一般的に、オファーの提示直後のイベントは追跡が容易です。たとえば、コール・センターにおいてエージェントがオファーを提示した時点で、顧客が示す興味関心の程度を測定できます。オファーが Web サイトに提示されている場合は、クリック・イベントが興味関心のインジケータになります。

オファーの存続期間の後半にあるイベントほど、オファーを追跡して妥当性を評価することが難しくなる場合があります。そのため、プロジェクトを開始して即座に結果の出るフィードバック・ループのみをクローズし、システムが熟成するにつれて下流のイベントを追加するのが一般的です。Oracle RTD では、開始直後のフィードバックのみでも、マーケティングの意思決定に大きく貢献できます。

5.1.1 選択肢グループにおけるイベントの定義について

イベントは、**選択肢グループ**のレベルで定義します。イベントはグループ階層の任意のレベルで定義できますが、ルートに近い最上位レベルで定義するのが一般的です。

ここではイベントを 2 つ作成します。1 つはオファーが顧客に**提示された**事実を示すイベントで、もう 1 つはオファーが**受け入れられた**事実を示すイベントです。このチュートリアルでは、アドバイザの結果として選択されたすべてのオファーが提示され、オファーの受入れがただちに判明することを前提にしています。

5.1.2 選択肢グループにおけるイベントの定義

1 「Inline Service Explorer」で、「**Cross Selling Offer**」選択肢グループを選択します。

2 「**Choice Events**」タブを選択します。「**Add**」を使用して、イベントを 2 つ追加します。最初のイベント名は `Presented`、2 番目のイベント名は `Accepted` にします。これらのイベント名は単なるラベルで、オファーの内部状態には対応しないことに注意してください。これらのイベントは選択肢イベント・モデル（次の項で説明）で使用され、これらのイベント名はそこで意味を持ちます。

3 イベントごとに、ドロップダウン・リストを使用して、「**Statistic Collector**」を「`Choice Event Statistic Collector`」に設定します。これはデフォルトの統計コレクタです。このコレクタによって、各イベントに関する統計が収集されます。

4 「**Event History (days)**」が「`Session Duration`」に設定されていることを確認します。

この設定により、セッション期間中にのみイベントが記憶されます。数日から数週間にわたるオファー・イベントの記憶が必要な場合は、ここで設定します。

5 「**Value Attribute**」は空欄のままにします。

これは、イベントの自動処理に使用されます。このチュートリアルでは、フィードバック・インフォーマントのロジックによりイベントを記録します。

6 すべてを保存します。

5.1.3 選択肢イベント・モデルについて

ここまでで、イベントの定義が完了し、統計を追跡する準備が整いました。イベントでは、統計の追跡に加えて、様々な要因（顧客の特性、通話とエージェント、およびオファーの成功や失敗）間における相関関係に関する学習が自己学習モデルにおいて行われます。この知識は次の 2 つの目的で役立ちます。

- マーケティング担当者や営業担当者の洞察力と把握力の向上
- 様々な状況で最適なオファーを提示する自動予測

このチュートリアルでは、これらの両方に対する使用方法について説明します。

5.1.4 選択肢イベント・モデルの定義

1 「Inline Service Explorer」で、「**Models**」を選択してから右クリックし、「**New Choice Event Model**」を選択します。新しいモデルの名前を `Offer Acceptance Predictor` に設定し、「**OK**」をクリックします。

2 モデルのエディタで、「**Default time window**」の選択を解除し、1 週間に設定します。

3 「**Choice Group**」で「`Cross Selling Offer`」を選択します。

このグループは選択肢階層の最上位グループで、このモデルを使用してオファーの受入れを追跡します。

4 「**Base Event**」で「`Presented`」を選択します。第5.1.2項で、選択肢グループのこれらのイベント名を定義したことを思い出してください。

このイベントが、成功を測定する対象イベントになります。オファーが受け入れられたかどうかは、オファーの提示後に追跡します。

5 「**Positive Outcome Events**」で、「**Select**」を使用してリストから「**Accepted**」を選択し、「**OK**」をクリックします。このチュートリアルでは、これが唯一の肯定的な結果です。複数のイベントを追跡する場合は、ここで追加します。

6 必要に応じて、イベントのラベルをオファー指向の名前に変更できます。

5.1.5 その他のモデル設定

選択肢イベント・モデルには、他にも役に立つ設定があります。「**Attributes**」タブには重要な設定が 2 つあります。1 つはパーティション化属性でもう 1 つは除外属性です。

5.1.5.1 パーティション化属性

パーティション化属性は、大きく異なる結果をもたらす区分に従ってモデルを分割する目的で使用されます。たとえば、同じオファーでも、Web 提示とコール・センター提示では、その受入れ特性が大きく異なる場合があります。その場合は、提示チャネルをパーティション化属性として設定できます。

パーティション化属性は複数設定できますが、メモリー使用量に影響する場合があることに注意してください。各パーティション化属性では、属性値の数だけモデル数が倍増します。たとえば、モデルに 3 つの値を持つパーティション化属性と 4 つの値を持つパーティション化属性がある場合、メモリーの使用量はパーティション化でないモデルの 12 倍になります。それでもパーティション化が重要な場合は、それによってモデルの予測性能と説明性能が大きく向上するため、必ずパーティション化属性を使用してください。

5.1.5.2 除外属性

モデルへの入力として意味がない属性があります。たとえば、Reason Analysis モデル（第3.1.2項を参照）では、入力として理由コードを持つことで、理由コードと通話理由の選択肢との間に相関関係が検出されました。この関係は、第2.10.5項で説明したロジックの結果と想定されます。この相関関係は人為的であり有意義な洞察でないため、理由コードをモデルから除外しました。

他のモデルでは理由コードが重要な要素となる場合があるため、除外が適切でない場合があることに注意してください。たとえば、Offer Acceptance Predictor モデルでは、オファーの受入れと理由コードとの間における相関関係に注目する必然性があります。

5.1.5.3 学習の実行場所

「**Learn Location**」タブで、モデルの学習をプロセス内のどのタイミングで実行するかを設定します。ほとんどの場合は、デフォルトの「**On session close**」が適しています。セッションにおいて複数の状態から学習を行う必要があるときは、特定の統合点において学習を設定すると有効な場合があります。

5.1.6 ループのクローズについて

選択肢イベント・モデルの定義が完了し、使用する準備が整いました。モデルに適切な情報を入力するには、ループをクローズするロジックを完成する必要があります。

提案されたオファーをモデルで使用可能にするには、オファーID をセッションで記憶する必要があります。オファーID はフロントエンド・クライアント側で記憶される場合もあるため、この手順は必須ではありませんが、ここではフロントエンドの機能を前提にしません。オファーの記憶には単純な文字列属性を使用するだけですが、複雑な場合は配列を使用して複数の選択肢を記憶します。

5.1.7 提案されたオファーの記憶

1. 「Inline Service Explorer」で、「**Entities**」の「**Session**」エンティティを選択します。
2. 「**Add Attribute**」を使用して属性を追加し、`Offer Extended` と命名します。
3. 属性の説明を入力します。「**Show in Decision Center**」および「**Use for Analysis**」の選択を解除します。「**OK**」をクリックします。

   この操作を行う理由は、現時点でこの属性を内部変数として扱い、ビジネス・ユーザーには表示しないためです。

4. 「Inline Service Explorer」で、「**Integration Points**」の「**Advisors**」にある「**Get Cross Sell Offer**」を選択します。エディタに移動します。

5 「**Asynchronous Logic**」タブで、既存コードを更新します。**提示された**イベントを記録して OfferExtended セッション属性に選択肢 ID の値を設定する行を既存コードに追加します。更新後のコードは次のようになります。

```
logInfo("Integration Point - Get Cross Sell Offer");

logInfo("  Customer age = " + session().getCustomer().getAge() );

// 'choices' is array returned by the 'Select Offer' decision

if (choices.size() > 0) {

  //Get the first offer from array
  Choice offer = choices.get(0);

  //For the selected offer, record that it has been 'presented'
  offer.recordEvent("presented");

  //Set the session attribute 'OfferExtended' with the offer's ID.
  session().setOfferExtended(offer.getSDOId());

  logInfo("  Offer presented: '" + offer.getSDOLabel() + "'");

}
```

これによって、選択された選択肢の SDOId がセッション・エンティティの OfferExtended 属性に割り当てられます。SDOId は一意の識別子です。Oracle RTD 構成のすべてのオブジェクトは一意な SDOId があります。オファー・オブジェクトには、選択されたオファーの Presented イベントも記録されます。イベント名は小文字表記で、Presented の選択肢イベント ID に対応することに注意してください。この ID を表示するには、「Inline Service Explorer」で、「**Choices**」を開いてから「**Cross Selling Offer**」をダブルクリックし、「**Choice Events**」タブ→ラベル/ID 切替ボタン  をクリックします。

デシジョンの時点で、どのオファーが選択されコール・センター・エージェントによって（Get Cross Sell Offer アドバイザを介して）顧客に提示されたかがセッションにおいて記憶されます。顧客からのレスポンスは判明していません。レスポンスは、次の項で説明するフィードバック・インフォーマントを介して送信されます。

5.1.8 フィードバック・インフォーマントの作成

このインフォーマントにより Oracle RTD に対して、オファー選択デシジョンの結果評価に必要な情報が提供されます。

1 「Inline Service Explorer」で、「**Integration Points**」を開き「**Informants**」を選択します。右クリックしてメニューから「**New Informant**」を選択します。インフォーマント名を `Offer Feedback` にします。

2 エディタで、インフォーマントの説明を入力します。「**External System**」で「**CRM**」を選択します。「**Order**」に「4」を入力します。

3 セッション・キーを Offer Feedback インフォーマントに追加するには、「**Session Keys**」の「**Select**」を使用して、「**Customer**」から「**customerId**」を選択します。「**OK**」をクリックします。

4 「**Add**」を使用して入力パラメータを追加します。パラメータ名を `Positive` にします。

5 データ型が「**String**」になっていない場合は選択します。

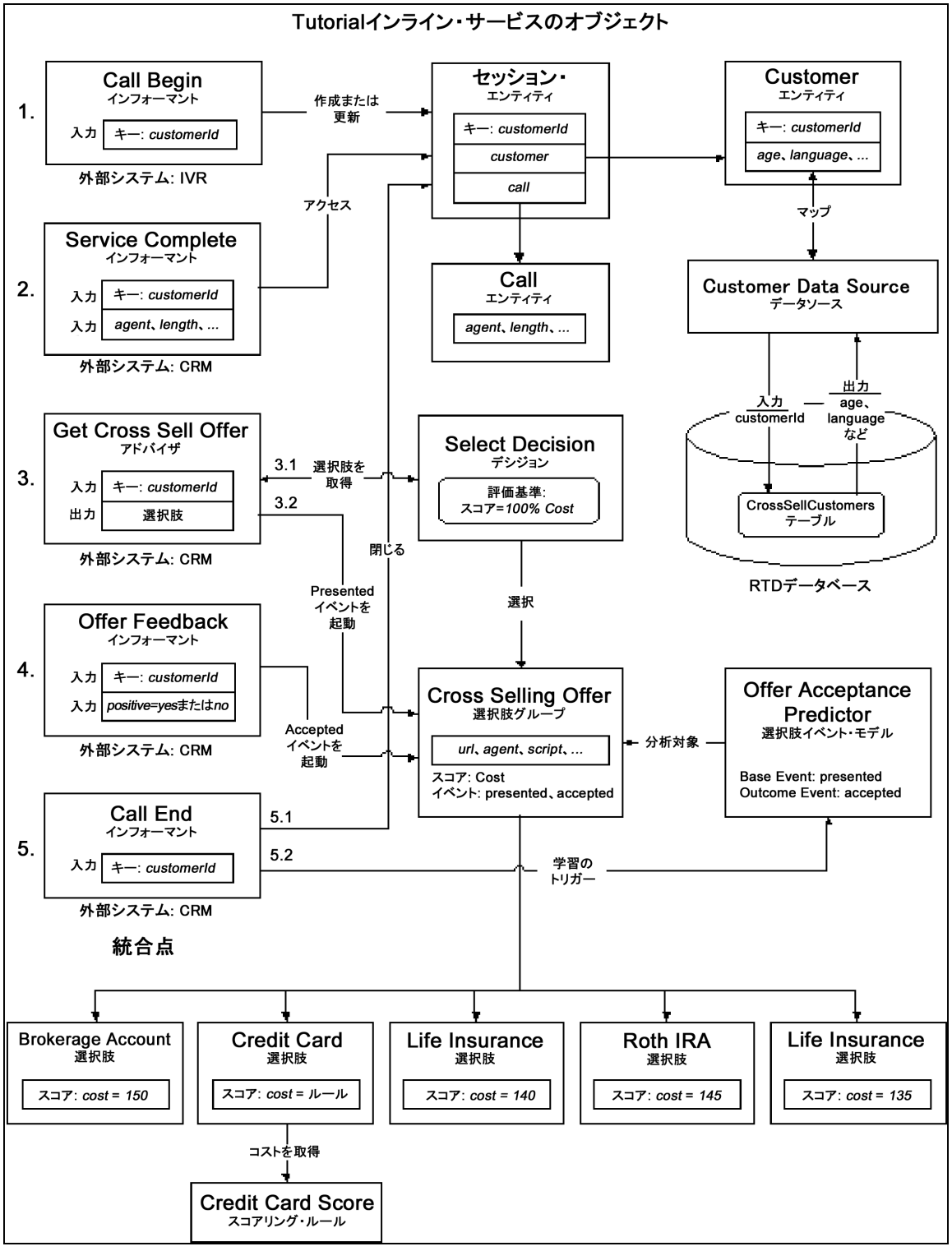
パラメータはマップされないままにします。このパラメータにセッション属性へのマッピングが不要なのは、この引数がオファーの受入れの成否を即時に判定する目的で使用されるためです。オファーの受入れは、yes 値によって示されます。

「**Logic**」タブの「**Logic**」に次のコードを入力し、受入れイベントの発生時に記録されるようにします。

```
logInfo("Integration Point - Offer Feedback");

//"yes" or "no" to accept offer.
String      positive = request.getPositive();
positive = positive.toLowerCase();


//Get the offer id from session attribute 'OfferExtended'
String extendedOfferID = session().getOfferExtended();

if (extendedOfferID != null) {
   //Get the offer from choice group 'Cross Selling Offer'
   Choice offer = CrossSellingOffer.getChoice(extendedOfferID);

   if (offer != null){
        String offerId = offer.getSDOId();

        //If response is "yes", then record the offer as accepted.
        if (positive.equals("yes")) {
            offer.recordEvent ("accepted");
            logInfo("  Offer '" + offer.getSDOLabel() + "' accepted");
        }

   }

}
```

6 すべてを保存し、インライン・サービスを再デプロイします。「**Deploy**」ダイアログで、「**Terminate Active Sessions (used for testing)**」を選択します。

7 次の図に、Get Cross Sell Offer アドバイザがオファーを取得および提示してから、Offer Feedback インフォーマントがオファーを受け入れるか拒否するまでのフローを示します。Call End インフォーマントによってセッションがクローズされると、Offer Acceptance Predictor モデルがオファーの Presented イベントまたは Accepted イベントによって更新されます。


Tutorialインライン・サービスのオブジェクト
1.
Call Begin
インフォーマント
入力
キー: customerId
外部システム: IVR
作成または
更新
セッション・
エンティティ
キー: customerId
customer
call
Customer
エンティティ
キー: customerId
age、language、...
マップ
アクセス
2.
Service Complete
インフォーマント
入力
キー: customerId
入力
agent、length、...
外部システム: CRM
Call
エンティティ
agent、length、...
Customer Data Source
データソース
入力
customerId
出力
age、
language
など
CrossSellCustomers
テーブル
RTDデータベース
3.
Get Cross Sell Offer
アドバイザ
入力
キー: customerId
出力
選択肢
外部システム: CRM
3.1
選択肢を
取得
3.2
Select Decision
デシジョン
評価基準:
スコア=100% Cost
閉じる
Presented
イベントを
起動
選択
4.
Offer Feedback
インフォーマント
入力
キー: customerId
入力
positive=yesまたはno
外部システム: CRM
Accepted
イベントを
起動
Cross Selling Offer
選択肢グループ
url、agent、script、...
スコア: Cost
イベント: presented、accepted
分析対象
Offer Acceptance
Predictor
選択肢イベント・モデル
Base Event: presented
Outcome Event: accepted
5.
Call End
インフォーマント
入力
キー: customerId
外部システム: CRM
5.1
5.2
学習の
トリガー
統合点
Brokerage Account
選択肢
スコア: cost = 150
Credit Card
選択肢
スコア: cost = ルール
Life Insurance
選択肢
スコア: cost = 140
Roth IRA
選択肢
スコア: cost = 145
Life Insurance
選択肢
スコア: cost = 135
コストを取得
Credit Card Score
スコアリング・ルール

5.1.9 フィードバック・インフォーマントのテスト

Offer Feedback インフォーマントをテストするには、最初に Get Cross Sell Offer をコールしてオファーを取得して提示する必要があります。

1 「**Test**」ビューで、Get Cross Sell Offer **統合点**を選択します。「**customerId**」に `10` などの値を入力します。

2 「**Send**」  をクリックし、「**Response**」サブタブでオファーが取得されたことを確認します。「**Log**」サブタブには、次のような結果が表示されます。

```
00:45:28,466 Integration Point - Get Cross Sell Offer

00:45:28,466   Customer age = 38

00:45:28,466   Offer presented: 'Credit Card'
```

他の値を「**customerId**」に入力しても、提示されるオファーは必ず Savings Account か Credit Card のどちらかになります。これは、この段階ではパフォーマンス目標をコストの最小化に対してのみ構成しており、顧客の年齢に応じて Savings Account または Credit Card が最小コストとなるためです。

3 「**Integration Point**」ドロップダウン・リストから、Offer Feedback インフォーマントを選択します。同一セッションで手順を続行するため、「**customerId**」の値はそのままにします。入力の「**Positive**」に、`yes` などの値を入力します。

4 「**Send**」をクリックし、Get Cross Sell Offer アドバイザによって取得されたオファーが受け入れられたことを「**Log**」サブタブで確認します。次のような結果が表示されます。

```
00:46:01,418 Integration Point - Offer Feedback

00:46:01,418   Offer 'Credit Card' accepted
```

5 入力の「**Positive**」値を `no` に変更し、Offer Feedback インフォーマントを再**送信**します。「**Log**」サブタブの内容は次のようになります。

```
00:47:31,494 Integration Point - Offer Feedback
```

5.1.10 Load Generator スクリプトの更新

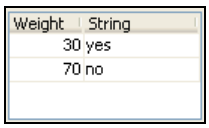
Get Cross Sell Offer アドバイザおよび Offer Feedback インフォーマントに対するコールが含まれるように、Load Generator スクリプトを更新します。これらの統合点コールは、ServiceComplete インフォーマントの後でしかも CallEnd インフォーマント（セッションをクローズするインフォーマント）の前に実行される必要があることに注意してください。ロジックは次のようになります。最初にコールを開始します。次に、標準サービス（通話理由の記録と分析を Reason Analysis モデルを使用して行う）を実行します。その後で、コスト最小化目標に基づいてエージェントが抱合せ販売オファーを顧客に提示します。オファー提示後、顧客のオファー受入れ可否を記録します。最後にコールまたはセッションの終了処理（オファーの提示や受入れに関する学習を Offer Acceptance Predictor モデルにより行う）を行います。

Get Cross Sell Offer アドバイザを Load Generator スクリプトに追加します。

1. *RTD_HOME*¥scripts¥loadgen.cmd を実行して、Load Generator を起動します。そして、以前のスクリプトを開きます。
2. 「**Edit Script**」タブを選択し、左側の領域を右クリックして「**Add Action**」を選択します。アクションのタイプを「**Message**」に、統合点の名前を GetCrossSellOffer にそれぞれ設定します。
3. 「**Input Fields**」で右クリックして「**Add item**」を選択し、入力フィールドを追加します。「**Name**」の下のスペースをクリックし、customerId を追加します。
4. 入力フィールドの「**Variable**」をクリックし、ドロップダウン・リストを使用して、対応する変数の var_customerId（第3.1.1項を参照）を選択します。「**Session Key**」を選択して、**customerId** をセッション・キーとしてマークします。
5. このアクションをスクリプトに追加すると、アクション・リストの下部に追加されます。GetCrossSellOffer が ServiceComplete の後にコールされるように、順序を調整する必要があります。「**Edit Script**」タブの左側で「**GetCrossSellOffer**」を右クリックしてから、「**Move Up**」または「**Move Down**」を使用して、順序が CallBegin、ServiceComplete、GetCrossSellOffer、CallEnd になるようにします。
6. Load Generator スクリプトを保存します。

Offer Feedback インフォーマントを Load Generator スクリプトに追加します。

1. Offer Feedback へのコールを「**Edit Script**」タブで追加する前に、新しい変数を「**Variable**」タブで作成する必要があります。Offer Feedback インフォーマントの定義を思い出してください。**positive** パラメータを使用してオファーの受入れを示します。Load Generator では、このパラメータ値はランダムに、**yes** が 30%、**no** が 70%になるように設定します。この設定は、重み付けされた文字列配列を使用して行います。
2. 「**Variables**」タブの左側で、「**Script**」フォルダを右クリックし、「**Add Variable**」を選択します。「**Variable name**」に var_positive と入力し、「**Contents**」タイプを「**Weighted String Array**」に設定します。2 つのアイテムを配列に追加します（コンテンツ・タイプの下のスペースを右クリックし、「**Add Item**」を選択）。最初のアイテムでは、「**Weight**」セルをダブルクリックして編集可能にして値として 30 を入力し、対応する「**String**」セルに値として yes を入力します。2 番目のアイテムでは、重み付け値を 70 にし、文字列値を no にします。重み付け値は合計を 100 にする必要はなく、自動的に正規化されます。この場合は、重み付け値を 6 と 14 にしても、同じ結果となります。

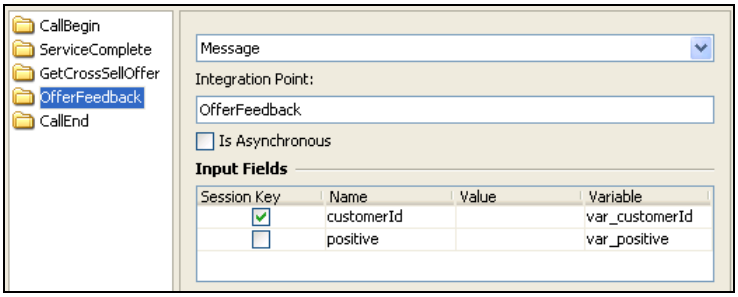


3 「**Edit Script**」タブを選択し、左側の領域を右クリックして「**Add Action**」を選択します。アクションのタイプを「Message」に、統合点の名前を `OfferFeedback` にします。

4 「**Input Fields**」で右クリックして「**Add item**」を選択し、入力フィールドを追加します。「**Name**」の下のスペースをクリックし、`customerId` を追加します。「**Variable**」列で、対応する変数の `var_customerId`（第3.1.1項を参照）を選択します。「**Session Key**」を選択して、**customerId** をセッション・キーとしてマークします。

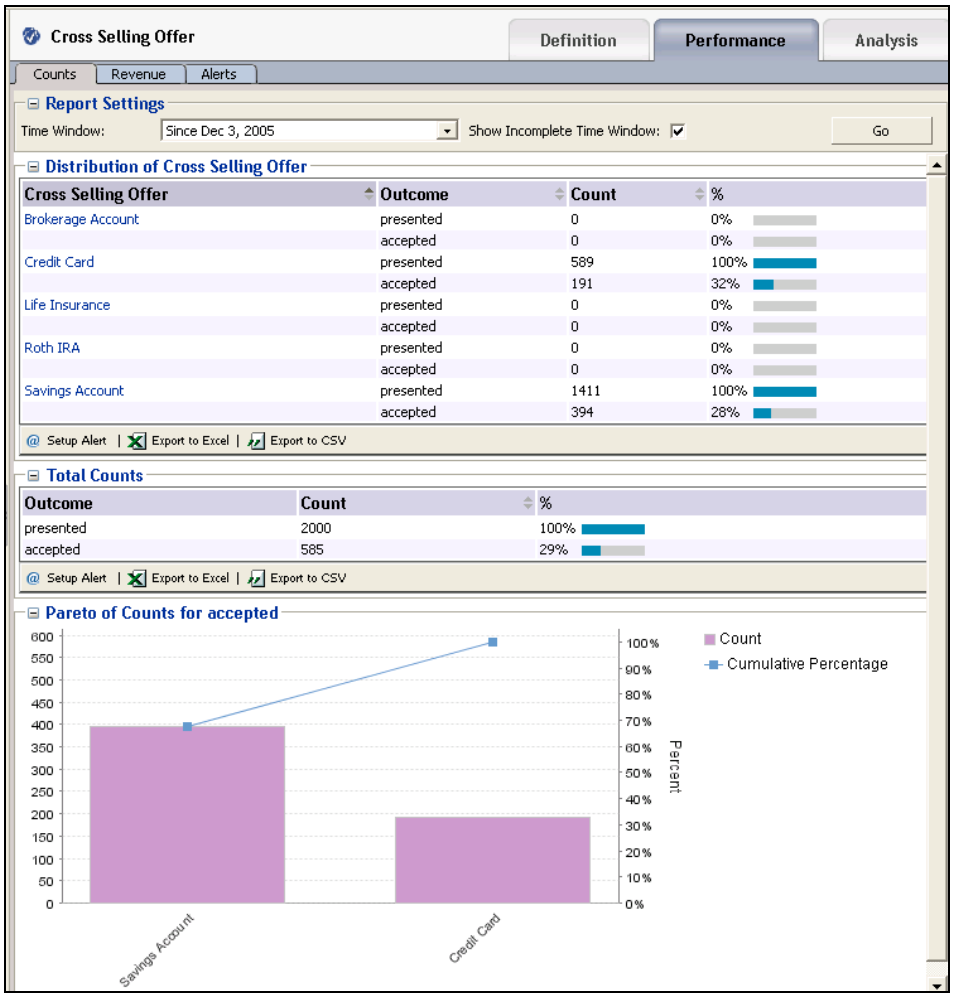
5 再び「**Input Fields**」で右クリックして「**Add item**」を選択し、入力フィールドを追加します。「**Name**」の下のスペースをクリックし、`positive` を追加します。「Variable」列で、対応する変数の `var_positive` を選択します。

6 このアクションをスクリプトに追加すると、アクション・リストの下部に追加されます。OfferFeedback が GetCrossSellOffer の後にコールされるように、順序を調整する必要があります。「**Edit Script**」タブの左側で「**OfferFeedback**」を右クリックし、「**Move Up**」または「**Move Down**」を使用して、順序が CallBegin、ServiceComplete、GetCrossSellOffer、OfferFeedback、CallEnd になるようにします。



7 Load Generator スクリプトを保存します。

Load Generator スクリプトをこの時点で実行できます。実行結果に古いデータが混在しないように、スクリプトを実行する前に既存データを再び削除することをお薦めします。削除方法の詳細は、第3.2項を参照してください。Load Generator スクリプトを実行した場合は、その結果を Decision Center で参照できます。Decision Center にログインし、「**Cross Selling Offer**」選択肢グループをクリックし、Offer Acceptance Predictor モデルの結果を表示します。「**Performance**」タブ→「**Counts**」サブタブをクリックします。オファーの分布とパレート図が次のように表示されます。



Credit Card と Savings Account の 2 つのオファーのみが提示され、それぞれの受入れ率が約 30%になっていることに注意してください。これは、これまでに設定したロジックの当然の結果です。設定ロジックでは、まずコスト最小化というパフォーマンス目標のみを適用し、顧客の年齢に応じて Savings Account または Credit Card が最小コストとなるように設定します（第4.1.3項を参照）。次に Load Generator スクリプトで、OfferFeedback インフォーマントを介して登録されるオファーへの肯定的なレスポンスを 30%に指定します。個々のオファーの分析レポートにドリルダウンすると、オファーの受入れとセッション属性との間における密接な相関関係はありません。これは、ランダムな顧客プロファイル・データを使用し、顧客またはその他の属性（通話の長さ、コール・エージェント名、通話理由など）に関係なく、受入れ率が 30%になるように強制適用しているためです。

これまで、パフォーマンス目標を使用して提示オファーを決定する方法と、選択肢イベント・モデルを使用して提示オファーの受入れ率を追跡する方法についてそれぞれ説明しました。これまでに使用したのは、**分析**に関するモデルのみです。次の項では、2 番目のパフォーマンス目標（売上の最大化）を追加し、そのモデルの学習内容を使用して、提示するオファーに**影響を与える**方法について説明します。また、2 人以上の子供を持つ顧客に Life Insurance オファーが提示された場合に、その受入れ確率を高める人為的なバイアスを導入します。これによって、このバイアスがモデルの結果にどのように影響するかを調べます。

5.2 モデルの予測機能の使用

これまでに作成したモデルでは、顧客特性、通話特性および抱合せ販売結果との間における相関関係の学習が行われます。このモデルは、オファー受入れの確率を予測する予測計算にも使用できます。この確率情報を使用することで、提示するオファーを決定する際にオファー値を調整できます。たとえば、オファーA の受入れ確率がオファーB の 2 倍と予測される場合は、提示するオファーの選択時に、オファーA を優先して選択するのが妥当です。この項では、2 番目のパフォーマンス目標として売上の最大化を導入します。売上の最大化では、その値やスコアの**受入れ確率**と**売上**単価の積として計算します。

たとえば、Brokerage Account オファーの売上単価が$300 で受入れ確率が 30%（0.3）と仮定した場合、その売上最大化スコアは$300 x 0.3 = $90 になります。Life Insurance オファーの売上単価は$185 で受入れ確率が 60%（0.6）と仮定した場合、売上最大化スコアは$185 x 0.6 = $111 になります。Brokerage Account のほうが売上単価の値が高くても、売上最大化スコアが高くなる Life Insurance オファーを優先して選択するほうが有利になります。

ここでは、コストと売上最大化の**両方**に基づいて、提示するオファーを選択することに注意してください。前述の例で、コストと売上最大化の重み付けされた合計において Brokerage Account が Life Insurance よりも高い場合は、Brokerage Account が選択されます。

この項では、最初に売上単価を追加し、その後で 2 番目のパフォーマンス目標である売上最大化を追加します。次に、売上最大化の目標のスコアとして、売上単価と受入れ確率との積の値を設定します。設定が完了したら、提示するオファーの選択時にコストと売上最大化の両方の目標が適用されるように、Select Offer デシジョンを更新します。最後に、Offer Feedback に、Life Insurance オファーの提示先であり特定のプロファイルを持つ顧客に対して、オファー受入れバイアスを導入するロジックを追加します。

5.2.1 売上単価選択肢属性の追加

1 「Inline Service Explorer」で、「**Cross Selling Offer**」選択肢グループを選択します。「**Choice Attributes**」タブで、「**Add**」ボタンをクリックします。

2 この属性の名前を `Revenue` に、データ型を `Integer` にそれぞれ設定します。オファーごとに異なる値を割り当てるため、「**Overridable**」オプションが選択されていることを確認します。

3 「**Cross Selling Offer**」選択肢グループの各選択肢に、次の表に示す「**Revenue**」属性の値を設定します。

選択肢	売上単価の値
Brokerage Account	300
Credit Card	205
Life Insurance	185
Roth IRA	190
Savings Account	175

5.2.2 2番目のパフォーマンス目標である売上最大化の追加

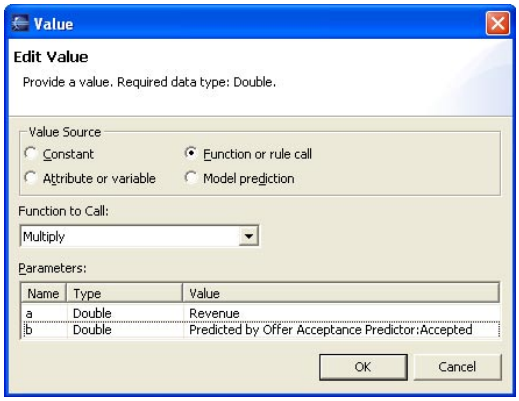
このチュートリアルにおける前述の項で、Cost パフォーマンス目標を定義しました。ここでは、2番目のパフォーマンス目標として Maximize Revenue を追加します。この新しいパフォーマンス・メトリックのスコアの計算では、選択肢の受入れ確率と売上単価を使用します。その計算式は、売上単価×受入れ確率＝潜在売上スコアとなります。

1. 「Inline Service Explorer」で、「**Performance Goals**」をダブルクリックし、エディタを起動します。「**Add**」ボタンを使用してパフォーマンス・メトリックを追加します。メトリック名を `Maximize Revenue` とし、「**OK**」をクリックします。
2. 「**Optimization**」で、「**Maximize**」を選択し、メトリックを「**Required**」にします。コストの$1 と売上の$1 は等しいため、正規化ファクタは調整不要です。
3. このメトリックを「**Cross Selling Offer**」選択肢グループに追加します。「**Inline Service Explorer**」で、「**Cross Selling Offer**」選択肢グループを選択します。「**Scores**」タブで、「**Add**」をクリックします。「**Select**」ダイアログで、「**Maximize Revenue**」を選択し、「**OK**」をクリックします。

5.2.3 パフォーマンス目標である Maximize Revenue のスコア値の計算

Maximize Revenue 目標のスコア値を計算するには、Offer Acceptance Predictor 選択肢イベント・モデルによって決定される受入れ確率値と売上単価が必要です。この値は、「Edit Value」ダイアログで値スコアを「**Model prediction**」に変更することで簡単に取得できます。

1. 「**Inline Service Explorer**」で、「**Cross Selling Offer**」選択肢グループを選択します。「**Scores**」タブで、Maximize Revenue メトリックの「**Scores**」列→省略記号 ... をクリックし、「**Edit Value**」ダイアログを起動します。
2. 「**Value Source**」で、「**Function or rule call**」を選択します。「**Function to Call**」で、「**Multiply**」関数を選択します。乗算対象パラメータで、パラメータ **a** の「**Value**」セルをクリックします。省略記号 ... をクリックして「**Attribute or variable**」を選択し、「**Choice**」フォルダを開きます。そして、「**Revenue**」を選択して「**OK**」をクリックします。乗算対象パラメータである「Parameters」で、パラメータ **b** の「**Value**」セルをクリックします。省略記号 ... をクリックして「**Model prediction**」を選択します。Offer Acceptance Predictor モデルと Accepted イベントによって予測される受入れ確率値を選択して「**OK**」をクリックします。



実際の受入れ確率値は 0 から 1 までの間になり、1 は受入れ確率が 100%を意味します。この値は NaN（Not a number）になる場合もあり、これは受入れ確率値の計算に十分なデータがモデルにないことを意味します。この場合、Maximize Revenue のスコア値は計算不能で、Select Offer デシジョンによるオファー選択は、組込みのスコア計算ロジックに基づきます（スコアが必須かどうかに応じて処理される）。

3 Maximize Revenue のスコアを選択肢グループ・レベルで定義することで、グループ内のすべての選択肢においてその定義が継承され、選択肢固有の売上単価値と受入れ確率値が実行時に適用されます。

5.2.4 2 番目のパフォーマンス目標の挿入による Select Offer デシジョンの更新

これまで、新しいパフォーマンス・メトリックとその値の計算方法を定義しました。次に、提示するオファーの選択時に両方のパフォーマンス・メトリックが適用されるように、Select Offer デシジョンを更新します。

1 「**Inline Service Explorer**」で、「**Decisions**」フォルダを開き、「**Select Offer**」を選択します。

2 「**Selection Criteria**」タブの「Priorities for "Default" Segment」テーブルには、**パフォーマンス目標**の Cost のみが 100%の**重み付け**値で表示されます。「**Goals**」をクリックしてパフォーマンス目標の「**Maximize Revenue**」を選択し、「**OK**」をクリックします。

3 優先度のテーブルに、2 つのパフォーマンス目標が、それぞれ**重み付け**値 50%で表示されます。デフォルトでは、重み付けは、選択されたすべてのメトリック間で均等に分配されます。Maximize Revenue パフォーマンス目標を Cost よりも優先する場合は、このパーセント値を調整して重み付けを高くします。このチュートリアルでは、デフォルトの**重み付け**の 50%を使用します。

4 次の表に、Select Offer デシジョンにより特定のオファーに対する合計スコアを計算する例を示します。この例では、オファーの Cost スコアを 150、Maximize Revenue スコアを 215 とします。

パフォーマンス目標	スコア	重み付け	「Maximize」/「Minimize」	正規化	重み付けされたスコア
Cost	150	50%	Minimize	1	-75
Maximize Revenue	215	50%	Maximize	1	107.5
				合計スコア	**32.5**

オファーの重み付けされたスコアの合計は 32.5 になります。Cost の重み付けされたスコアは、「Optimization」が「Minimize」のため負の値になります。オファーの合計スコアは、2 つの重み付けされたスコアの加算値です。合計スコアはオファーごとに計算され、最高値を持つオファーが選択されます。

5.2.5 受入れ確率を表示するための選択肢属性の追加

受入れ確率の値を表示するには、対応する選択肢属性を追加し、**logInfo** を介して表示するか、「**Test**」ビューの「Response」タブに表示します。

1. 「Inline Service Explorer」で、「**Cross Selling Offer**」選択肢グループを選択します。「**Choice Attributes**」タブで、「**Add**」をクリックして属性を追加します。プロパティ・ダイアログで、「**Display Label**」を `Likelihood Of Acceptance` に設定します。「**Data Type**」を `Double` に設定します。
2. この選択肢グループ内のすべての選択肢によってこの属性に同じ定義が使用されるため、「**Overridable**」オプションの選択を解除します。「**Send to client**」オプションを選択し、「**OK**」をクリックします。
3. Likelihood Of Acceptance 属性の「**Value**」列で、省略記号 ... をクリックし、その値を設定します。「**Edit Value**」ダイアログで、「**Value Source**」を「**Model prediction**」に設定します。**Offer Acceptance Predictor** モデルと **Accepted** イベントを選択し、「**OK**」をクリックします。
4. インライン・サービスに対するすべての変更を保存します。

5.2.6 受入れ確率値のチェック

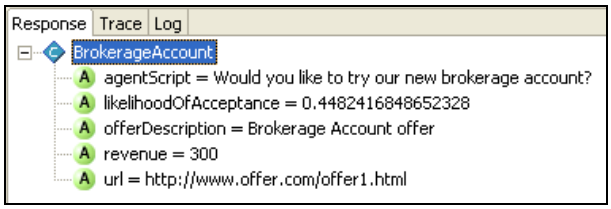
受入れ確率値を表示するには、**logInfo** 文を Get Cross Sell Offer アドバイザに追加します。

1. 「Inline Service Explorer」で、「**Integration Points**」の「**Advisors**」にある「**Get Cross Sell Offer**」を選択します。エディタに移動します。
2. 「**Asynchronous Logic**」タブで、受入れ確率の値を出力する行を追加して、既存コードを更新します。更新後のコードは次のようになります。

```
logInfo("Integration Point - Get Cross Sell Offer");

logInfo("  Customer age = " + session().getCustomer().getAge() );

// 'choices' is array returned by the 'Select Offer' decision.  The
// name 'choices' was set (and can be changed) in the 'Choice Array'
// text box in the 'Select Offer' decision's 'Pre/Post Selection
// Logic' tab.
if (choices.size() > 0) {

  //Get the first offer from array
  Choice offer = choices.get(0);

  //For the selected offer, record that it has been 'presented'
  offer.recordEvent("presented");

  //Set the session attribute 'OfferExtended' with the offer's ID.
  session().setOfferExtended(offer.getSDOId());

  logInfo("  Offer presented: '" + offer.getSDOLabel() + "'");

  //Cast selected offer to type CrossSellingOfficeChoice -
  //the base Choice type of choice group 'Cross Selling Offer'
  CrossSellingOfferChoice cso = (CrossSellingOfferChoice) offer;
  logInfo("  Likelihood of Acceptance = " + cso.getLikelihoodOfAcceptance());

}
```

3 アドバイザの変更結果を確認するには、すべてを保存し、インライン・サービスをデプロイします。

4 「**Test**」ビューで、Get Cross Sell Offer 統合点を選択し、「**customerId**」に 8 などの値を入力します。「**Send**」をクリックします。「**Test**」ビューの「**Response**」サブタブに、次のような結果が表示されます。



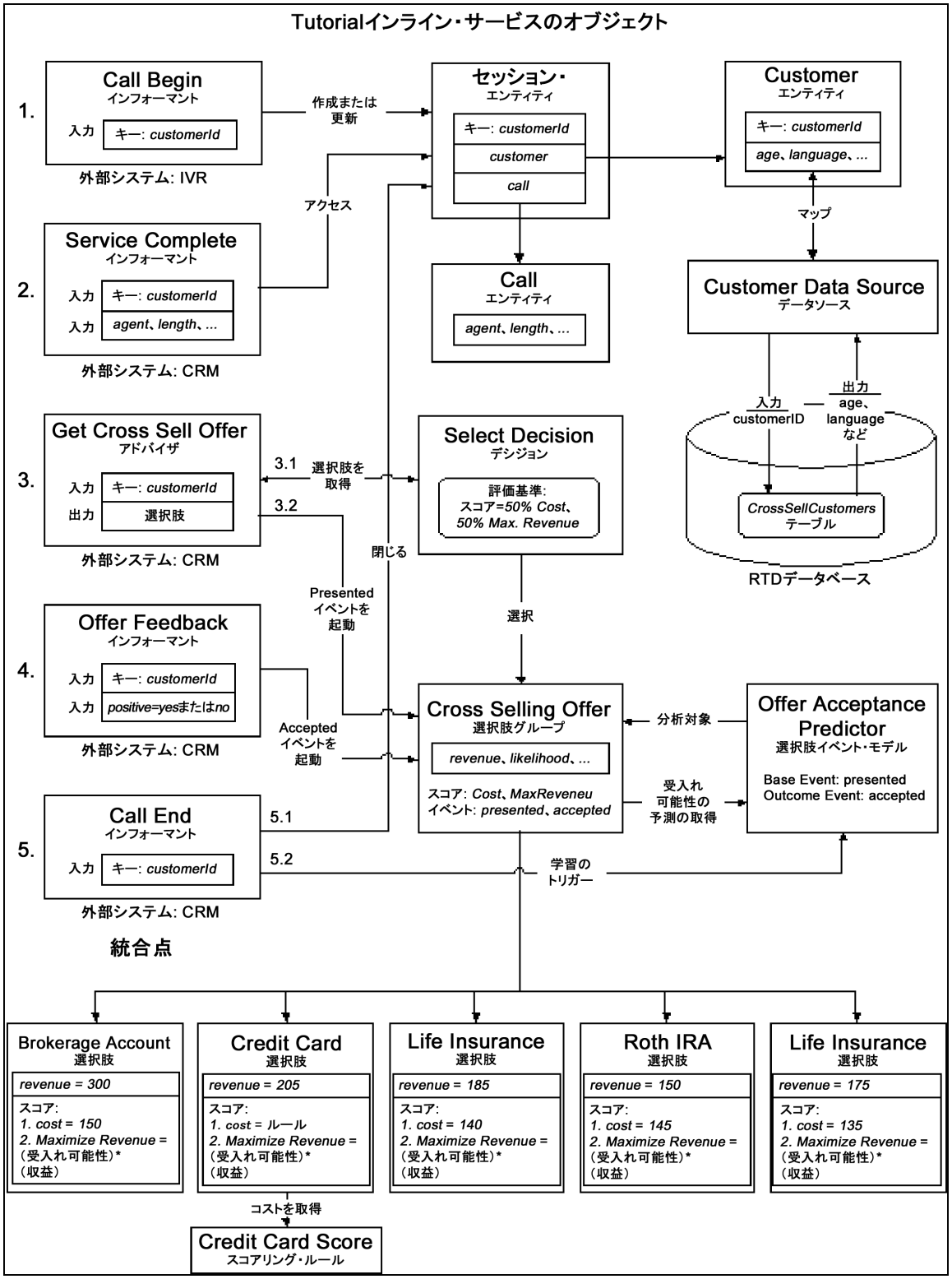
「**Log**」サブタブには、次のような結果が表示されます。

```
14:07:37,908 Integration Point - Get Cross Sell Offer

14:07:37,908   Customer age = 57

14:07:37,908   Offer presented: 'Brokerage Account'

14:07:37,908   Likelihood of Acceptance = 0.4482416848652328
```

Likelihood Of Acceptance で NaN（Not A Number）値を取得した場合は、受入れ確率値の計算に十分なデータがモデルにないことを意味します。モデルの収束（受入れ確率値が NaN でなくなる）に必要な反復実行回数は、アプリケーションとデータの質によって異なります。

このケースでは、オファーの受入れ確率を 30%にして（第5.1.10項を参照）、ランダムな顧客プロファイル・データを使用するため、Offer Acceptance Predictor モデルは短時間で収束し、数百回の反復実行で受入れ確率を計算できます。モデルが収束する前に、オファー選択プロセスは、組込みのスコア計算ロジックに基づいて決定します（スコアが必須かどうかに応じて処理される）。

5 次の図に、各オファーの合計スコアを Cost と Maximize Revenue の 2 つのスコアの重み付けされた合計として、Get Cross Sell Offer アドバイザが Cross Selling Offer 選択肢グループからオファーを取得するフローを示します。


Tutorialインライン・サービスのオブジェクト
1.
Call Begin
インフォーマント
入力
キー: customerId
外部システム: IVR
作成または
更新
セッション・
エンティティ
キー: customerId
customer
call
Customer
エンティティ
キー: customerId
age、language、...
マップ
アクセス
2.
Service Complete
インフォーマント
入力
キー: customerId
入力
agent、length、...
外部システム: CRM
Call
エンティティ
agent、length、...
Customer Data Source
データソース
入力
customerID
出力
age、
language
など
CrossSellCustomers
テーブル
RTDデータベース
3.
Get Cross Sell Offer
アドバイザ
入力
キー: customerId
出力
選択肢
外部システム: CRM
3.1
選択肢を
取得
3.2
Select Decision
デシジョン
評価基準:
スコア=50% Cost、
50% Max. Revenue
閉じる
Presented
イベントを
起動
選択
4.
Offer Feedback
インフォーマント
入力
キー: customerId
入力
positive=yesまたはno
外部システム: CRM
Accepted
イベントを
起動
Cross Selling Offer
選択肢グループ
revenue、likelihood、...
スコア: Cost、MaxReveneu
イベント: presented、accepted
分析対象
受入れ
可能性の
予測の取得
Offer Acceptance
Predictor
選択肢イベント・モデル
Base Event: presented
Outcome Event: accepted
5.
Call End
インフォーマント
入力
キー: customerId
外部システム: CRM
5.1
5.2
学習の
トリガー
統合点
Brokerage Account
選択肢
revenue = 300
スコア:
1. cost = 150
2. Maximize Revenue =
(受入れ可能性)*
(収益)
Credit Card
選択肢
revenue = 205
スコア:
1. cost = ルール
2. Maximize Revenue =
(受入れ可能性)*
(収益)
Life Insurance
選択肢
revenue = 185
スコア:
1. cost = 140
2. Maximize Revenue =
(受入れ可能性)*
(収益)
Roth IRA
選択肢
revenue = 150
スコア:
1. cost = 145
2. Maximize Revenue =
(受入れ可能性)*
(収益)
Life Insurance
選択肢
revenue = 175
スコア:
1. cost = 135
2. Maximize Revenue =
(受入れ可能性)*
(収益)
コストを取得
Credit Card Score
スコアリング・ルール

5.2.7 特定の顧客に対するオファー受入れバイアスの導入

これまで、Offer Feedback インフォーマントにおいて、提示されたオファーが受け入れられたかどうかを示す Positive インフォーマント・パラメータを指定しました。次に、このインフォーマントがコールされたときに、Positive パラメータに yes の値が 30%の割合で渡されるように、Load Generator スクリプトを更新しました（第5.1.10項を参照）。このパーセント値は顧客のプロファイル・データによって変わらないため、どのような顧客に対しても、提示されたすべてのオファーの受入れ確率が 30%程度になります。

この段階で Load Generator スクリプトを実行しても、モデルによって、顧客属性とオファー受入れとの間において密接な相関関係が示されることはありません。ここでは、2 人以上の子供を持つ顧客に Life Insurance オファーが提示された場合に、肯定的なオファー受入れを Offer Feedback インフォーマントのロジックに必ず記録する人為的なバイアスを導入します。このロジックでは、デフォルトの受入れ確率（Load Generator スクリプトに定義済）に加えて、Life Insurance オファーの受入れ確率を 30%よりも高い値に偏向させます。これによって、Decision Center では、子供の数と Life Insurance オファーの受入れ確率との間に明確な相関関係が見られるようになります。

1 「Inline Service Explorer」で、「**Integration Points**」の「**Informants**」にある「**Offer Feedback**」を選択します。エディタに移動します。

2 「**Logic**」タブで、2 人以上の子供を持つ顧客に Life Insurance オファーが提示された場合にオファー受入れバイアスを追加する行を追加して、既存コードを更新します。更新後のコードは次のようになります。

```
logInfo("Integration Point - Offer Feedback");

//"yes" or "no" to accept offer.
String      positive = request.getPositive();
positive = positive.toLowerCase();


//Get the offer id from session attribute 'OfferExtended'
String extendedOfferID = session().getOfferExtended();

if (extendedOfferID != null) {
   //Get the offer from choice group 'Cross Selling Offer'
   Choice offer = CrossSellingOffer.getChoice(extendedOfferID);

   if (offer != null){
        String offerId = offer.getSDOId();

        //Introduce artificial bias for customers with 2 or more
        //children to always accept "LifeInsurance" if it was
        //selected after scoring.
        //If data source is Oracle, change the following method from
        //getNumberOfChildren() to getNumberofchildren()
        int numOfChildren = session().getCustomer().getNumberOfChildren();
        if ( numOfChildren >= 2 && offerId.equals("LifeInsurance")) {
            positive="yes";
        }

        //If response is "yes", then record the offer as accepted.
        if (positive.equals("yes")) {
            offer.recordEvent ("accepted");
            logInfo("  Offer '" + offer.getSDOLabel() + "' accepted");
        }
```

```
    }

}
```

3 すべての変更を保存し、インライン・サービスをデプロイします。

5.2.8 Load Generator スクリプトの実行

第5.1.10項で、Get Cross Sell Offer アドバイザと Offer Feedback インフォーマントが含まれるように Load Generator スクリプトを更新しました。この時点では、オファー選択プロセスは、コストの最小化というパフォーマンス目標にのみ基づいていました。次に、2 番目のパフォーマンス目標として売上の最大化を追加し、それに Offer Acceptance Predictor モデルによって計算される受入れ確率の予測値を使用しました。これによってオファー選択プロセスは、両方のパフォーマンス目標に基づくことになります。さらに、特定のプロファイルに合致する顧客に Life Insurance オファーが提示された場合の受入れバイアスを導入しました。ここで、Load Generator スクリプトを再度実行して、結果を確認します。

1 OC4J または WebLogic を使用している場合は、*JAVA_HOME*¥bin ¥jconsole.exe を実行して JConsole を起動します。WebSphere を使用している場合は、JConsole の構成中に作成したバッチ・スクリプトを実行します。JConsole の使用方法の詳細は、『Oracle Real-Time Decisions インストレーションおよび管理ガイド』を参照してください。

2 「**Remote**」タブを選択します。インストール中に作成した管理者資格証明と適切なポート番号（通常は 12345）を入力し、「**Connect**」をクリックします。

3 「**MBean**」タブをクリックしてから、「**OracleRTD**」→「**InlineServiceManager**」→「**Tutorial**」→「**Development**」→「**Loadable**」MBean に移動します。

4 「**Operations**」タブをクリックしてから、**deleteAllOperationalData()**操作を使用して、このインライン・サービスの操作データをすべて（結果も含めて）削除します。

5 Load Generator を起動し、前の項で定義した Load Generator スクリプトを開きます。変更は不要です。

6 Load Generator スクリプトを実行します。200 回ほど実行後、一時停止ボタン  をクリックし、サーバーへのリクエストの送信を一時的に停止します。server.log ファイルにおけるサーバー出力内容を確認します。すべてのセッションでは、出力された受入れ確率値が NaN になります。これは、受入れ確率値を計算できるだけ十分なデータの学習がモデルにおいて行われていないことを意味します。確率値がなくても、オファーは提示されることに注意してください。オファーは、組込みのスコア計算ロジックを使用して選択されます。

7 Load Generator スクリプトの一時停止を解除し、実行回数が 2000 回になるまでスクリプトを実行します。今回は、サーバー出力に受入れ確率の実際値が出力されます。Life Insurance 以外のすべてのオファーでは 0.3 前後になりますが、Life Insurance オファーでは導入したバイアスの結果として値が高くなります。

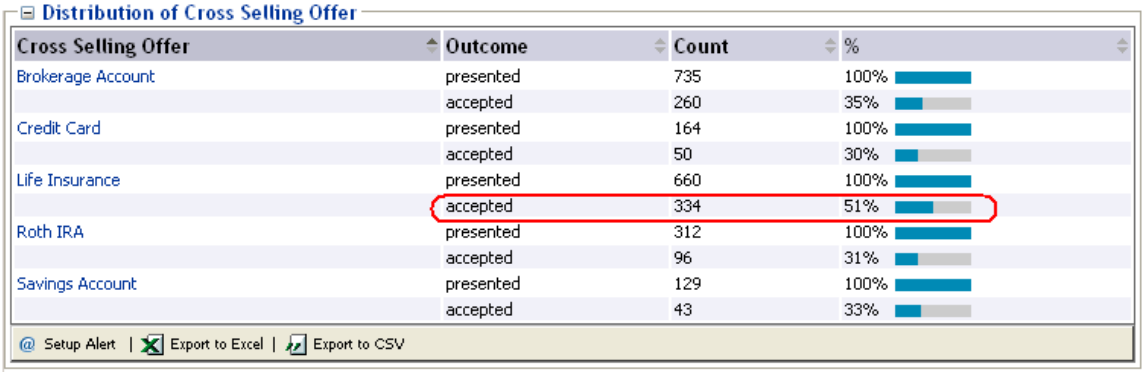
8 特定のオファーで、モデルが予測する受入れ確率値が顧客のプロファイルに応じて異なることに注目することが重要です。たとえば、顧客が 2 人（John と Tom）いて、子供の数のみが異なると仮定します。2 人の顧客の Life Insurance オファーに対する受入れ確率値を調べると（ある時点でのスナップショットとして）、Tom の値のほうが高いことが判明します。これは、Tom には 3 人の子供がいて、Life Insurance オファーが提示された場合の受入れ確率が高くなることによります。

顧客名	子供の数	Life Insurance オファーの受入れ確率値
John Doe	0	0.32
Tom Smith	3	0.89

顧客に提示されるオファーは Cost スコアと Maximize Revenue スコアの組合せに基づいて決定され、オファーごとにモデルによって予測される受入れ確率値に Maximize Revenue が基づくため、2 人以上の子供を持つ顧客では Life Insurance オファーの Maximize Revenue 値が高くなり、結果として、こうした顧客に対しては、他のオファーよりも Life Insurance の提示率（さらには受入れ率）が高くなります。

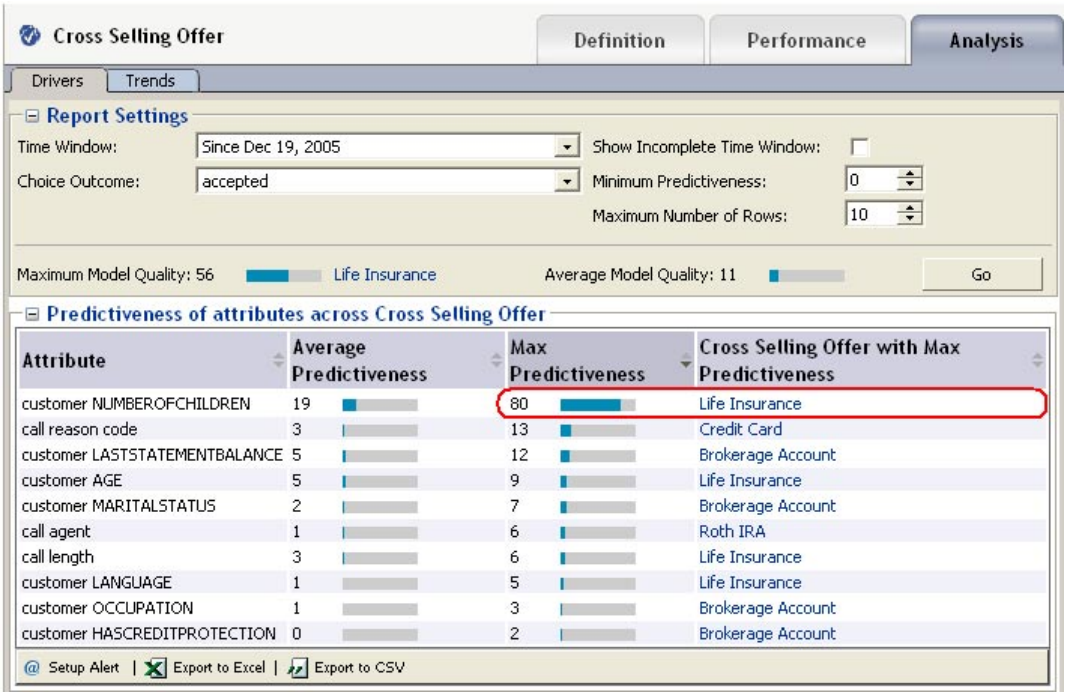
5.2.9 結果の検証

Load Generator の実行結果を参照するには、Decision Center にログインします。「**Cross Selling Offer**」選択肢グループを左側のナビゲーション・ボックスでクリックします。ここに Offer Acceptance Predictor モデルの結果が表示されます。「**Performance**」タブ→「**Counts**」サブタブをクリックします。次のような結果が表示されます。



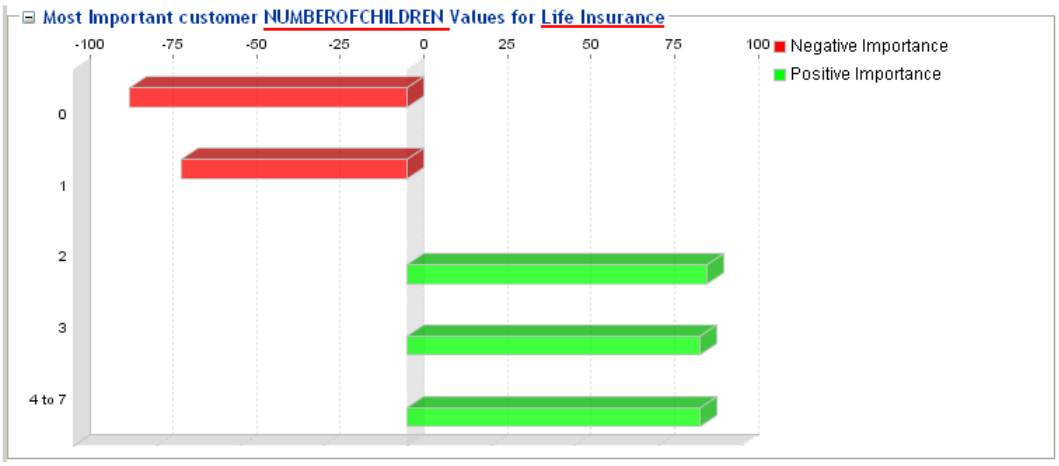
このテーブルはオファーの分布（つまり、オファーごとの提示数と受入れ数）を示しています。Life Insurance 以外の他のすべてのオファーでは、受入れ率が 30%前後になっています。これは、Load Generator スクリプトの設定の当然の結果です（第5.1.10項を参照）。Life Insurance の受入れ率は 30%よりも高くなっています。これは、第5.2.7項で人為的なバイアスを導入したためです。このバイアスによって、すべてのオファーの受入れ率が 30%程度になることに加えて、2 人以上の子供を持つ顧客に Life Insurance オファーが提示されたときは必ず受け入れるようにされます。

人為的なバイアスを設定したことで、Life Insurance オファーでは NumberOfChildren 属性がオファー受入れの重要指標となることが、モデル結果に反映されます。Decision Center レポートには、このことが示されます。「**Cross Selling Offer**」選択肢グループ→「**Analysis**」タブ→「**Drivers**」サブタブをクリックします。「**Report Settings**」セクションで、「**Minimum Predictiveness**」の値を 0 に変更し、「**Go**」をクリックします。属性の一覧が、予測値の高い順に表示されます。「Max Predictiveness」の最高値は、NumberOfChildren 属性になります。この属性が、追加した唯一の人為的なバイアスのためです。対応するオファーは Life Insurance で、次のようなレポートになります。



Life Insurance オファー固有のレポートを表示することで、このオファーに対して NumberOfChildren 属性が持つ重要性をさらに分析できます。Decision Center のナビゲーション・ボックスで、「**Cross Selling Offer**」選択肢グループを開き、選択肢の「**Life Insurance**」→「**Analysis**」タブ→「**Drivers**」タブをクリックします。このレポートには、この特定のオファー（**Life Insurance**）の受入れに対する重要な誘因が示されます。

「**Report Settings**」セクションで、「**Minimum Predictiveness**」の値を 0 に変更し、「**Go**」をクリックします。属性の一覧が、**予測**値の高い順に表示されます。NumberOfChildren 属性が、最も高い予測値を持ちます。属性名をクリックし、詳細レポートを表示します。最初のレポートには、次のようなグラフが表示されます。



このグラフは、NumberOfChildren の値が 2 以上の場合、オファー受入れと密接な正の相関関係があることを示しています。これは、この属性値（2 以上）の場合、このオファーの受入れ数が期待値よりも大きくなることを意味します。同様に、値が 0 または 1 の場合も相関関係が非常に密接になりますが、この相関関係は負で、子供がいないか子供が 1 人の顧客には期待するほど Life Insurance オファーが受け入れられないことを意味します。