Oracle Telephony Adapter SDK

開発者リファレンス・ガイド

リリース 11*i*

2003 年 1 月

部品番号：J07433-01

ORACLE®

Oracle Telephony Adapter SDK 開発者リファレンス・ガイド , リリース 11*i*

部品番号：J07433-01

原本名：Oracle Telephony Adapter SDK Developer's Reference Guide, Release 11*i*

原本部品番号：A97207-01

目次

3 開発環境

4 Oracle Telephony Adapter SDK のコンポーネント

5 Telephony Adapter の開発

6 Telephony Adapter のテスト

はじめに

対象読者

『Oracle Telephony Adapter SDK 開発者リファレンス・ガイド』のリリース 11*i* にようこそ。次の実務知識を持つ読者を対象にしています。

テレフォニ開発者

- Java および C プログラム言語
- TCP/IP ソケット
- マルチスレッド、同期および待機通知などのスレッド
- イベント・ドリブンや同期モードと非同期モードなどのリアルタイム・プログラミング

テレフォニ・アナリスト

- ACD キュー、ルート・ポイント、エージェント動作モード、ANI および DNIS など、コール・センターの概念
- 転送、会議およびコール ID の推移など、テレフォニ・プラットフォーム（PBX および CTI ミドルウェア）のコール・モデル
- Intel CT Connect や Cisco ICM など、ベンダー固有の CTI API およびテスト・ツール

Oracle Applications 製品情報の詳細は、「その他の情報ソース」を参照してください。

このマニュアルの使用方法

このマニュアルには、Oracle Telephony Adapter SDK を理解し、使用するために必要な情報が含まれています。

- 第 1 章「概要」では、Oracle Telephony Adapter SDK と Oracle Advanced Inbound の機能の概要について説明します。
- 第 2 章「主な機能」では、Oracle Telephony Adapter SDK の概念とキーや値などの詳細について説明します。
- 第 3 章「開発環境」では、ソフトウェアとハードウェアの最低要件、代表的な構成と拡張性およびパフォーマンス上のガイドラインについて説明します。
- 第 4 章「Oracle Telephony Adapter SDK のコンポーネント」では、SDK の内容とインストールおよび C プログラム言語と Java プログラム言語用の API について説明します。
- 第 5 章「Telephony Adapter の開発」では、Oracle Telephony Adapter SDK の実装手順について説明します。
- 第 6 章「Telephony Adapter のテスト」では、アダプタ実装の検証方法について説明します。
- 第 7 章「Telephony Adapter の配布」
- 付録 A「SDK 条件設定ワークシート」では、顧客サイトで Oracle Telephony Adapter SDK の統合が必要かどうかを判断する場合に使用するワークシートと、SDK の要件について説明します。
- 付録 B「SDK のスコープ分析」では、顧客サイトで SDK 要件を特殊化する場合に使用するワークシートについて説明します。
- 付録 C「Telephony Adapter のテスト・ケース」
- 付録 D「サンプル・コード」には、関連コードの例が記載されています。

文書のユーザー補助

オラクル社では、ハンディキャップのあるお客様にも Oracle の製品、サービスおよびサポート・マニュアルを簡単にご利用いただけることを目標としています。そのため、Oracle のマニュアルには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。このマニュアルには HTML 版も用意されており、ハンディキャップのあるお客様が簡単にアクセスできるマークアップをご利用いただけます。標準規格は改善されつつあります。オラクル社でも、Oracle のマニュアルをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。詳細は、Oracle Accessibility Program の Web サイト（http://www.oracle.com/accessibility/）を参照してください。

文書のコード例のユーザー補助

Windows スクリーン・リーダーである JAWS では、この文書内のコーディング例を正しく読み上げられない場合があります。コードを記述するときの規則では、閉じ括弧のみを 1 行に記述する必要があります。ただし、JAWS は通常、大括弧または中括弧のみでなるテキスト行は読み上げません。

文書内の外部 Web サイトへのリンクのユーザー補助

この文書には、オラクル社が所有または管理していない企業や組織の Web サイトへのリンクが含まれています。オラクル社はそれらの Web サイトのユーザー補助に関しては、評価も言及もしません。

その他の情報ソース

オンライン・マニュアル、研修およびサポート・サービスを含む多数の情報ソースを選択して Oracle Telephony Adapter SDK の知識と理解を深めることができます。

このマニュアルで他の Oracle Applications ドキュメントを参照する場合、それぞれのマニュアルのリリース 11*i* バージョンのみを使用してください。

オンライン・マニュアル

Oracle Applications のマニュアルは、すべてオンライン（HTML または PDF）で利用できます。オンライン・ヘルプのパッチはオラクル・サポート・ホームページで入手できます。

関連文書

Oracle Telephony Adapter SDK は、ビジネス情報や設定情報を他の Oracle Applications 製品と共有しています。したがって、Oracle Telephony Adapter SDK の設定や使用に際して、他の製品マニュアルの参照が必要になることがあります。

これらのマニュアルをオンラインで参照するには、HTML ヘルプ・ウィンドウの拡張メニューから「ライブラリ」を選択するか、メディア・パックに含まれている Oracle Applications Document Library CD から読み込むか、あるいは Web ブラウザを使用してシステム管理者から提供される URL にアクセスします。

すべての製品に関連するマニュアル

『Oracle Applications ユーザーズ・ガイド』

このマニュアルでは、このリリースの Oracle Telephony Adapter SDK（およびその他の Oracle Applications 製品）で利用できるグラフィカル・ユーザー・インタフェース（GUI）を使用したデータの入力、問合せ、レポートの実行およびナビゲートの方法について説明します。さらにこのマニュアルには、ユーザー・プロファイルの設定、ならびにレポートおよびコンカレント・プロセスの実行および検討の詳細が含まれます。

オンラインでこのマニュアルにアクセスするには、Oracle Applications ヘルプ・ファイルの任意のヘルプ・ファイルで「Oracle Applications のスタート・ガイド」を選択します。

この製品に関連するマニュアル

『Oracle Advanced Inbound Implementation Guide』

このマニュアルは、Oracle Advanced Inbound の構成に関するインストール後の実装手順について説明します。

『Oracle Interaction Center Concepts and Procedures』

このマニュアルは、Interaction Center サーバー・グループ・アーキテクチャの基本的な概念と管理について説明します。

『Oracle Interaction Center Implementation Guide』

このマニュアルは、Interaction Center サーバー・マネージャ（ICSM）のインストールおよび実装について説明します。

インストールおよびシステム管理者向け

『Oracle Applications 概要』

このマニュアルは、Oracle Applications リリース 11*i* の概念、機能、技術スタック、アーキテクチャおよび用語について説明します。Oracle Applications をインストールする前に読む必要のある有益な入門書です。また、このマニュアルでは、ビジネス・インテリジェンス（BIS）、言語、文字セットおよび Self-Service Web Applications といったアプリケーション規模での機能の背景となるコンセプトを紹介します。

『Oracle Applications のインストール』

このマニュアルは、Oracle Applications 製品のインストールを管理する方法について説明します。リリース 11*i* では、インストール・プロセスの大部分が、Oracle Rapid Install を使用して処理されます。必要なステップの多くが自動化されるため、Oracle Applications、Oracle8 テクノロジ・スタックおよび Oracle8*i* サーバー・テクノロジ・スタックのインストール所要時間が短縮されます。このマニュアルには、Oracle Rapid Install の使用のための指示を含めて、インストールを完了するために必要なタスクがリストされています。このマ

ニュアルは、各製品のユーザーズ・ガイドおよびインプリメンテーション・ガイドと併用する必要があります。

『Supplemental CRM Installation Steps』

このマニュアルは、少数の CRM 製品の完全インストールに必要な特定の手順について説明します。各手順は、『Oracle Applications のインストール』に記載されているタスクの完了直後に実行する必要があります。

『Oracle Applications のアップグレード』

このマニュアルは、Oracle Applications リリース 10.7 またはリリース 11.0 の製品をリリース 11*i* へアップグレードする場合に参照します。このマニュアルは、アップグレード・プロセスを説明し、データベース・アップグレード・タスクおよび製品固有のアップグレード・タスクをリストしています。リリース 11*i* にアップグレードするには、リリース 10.7（NCA、SmartClient またはキャラクタ・モード）かリリース 11.0 のどちらかがインストール済みである必要があります。10.7 以前のリリースからリリース 11*i* には直接アップグレードできません。

『Oracle Applications の保守』

このマニュアルは、AutoUpgrade、AutoPatch、AD Administration、AD Controller、AD Relink、License Manager およびその他の様々な AD ユーティリティを実行する際に役立ちます。このマニュアルには、AD ユーティリティの実行に必要な操作手順、スクリーン・ショットおよびその他の情報が記載されています。また、このマニュアルでは、Oracle Applications ファイル・システムおよびデータベースの保守に関する情報を提供しています。

『Oracle Applications システム管理者ガイド』

このマニュアルでは、Oracle Applications システム管理者向けに計画情報および参照情報を提供します。セキュリティの定義、メニューやオンライン・ヘルプのカスタマイズおよびコンカレント処理の管理に関する情報が記載されています。

『Oracle Alert ユーザーズ・ガイド』

このマニュアルでは、Oracle Applications データの状態をモニターする周期アラートとイベント・アラートの定義方法を説明します。

『Oracle Applications 開発者ガイド』

このマニュアルは、Oracle Applications 開発スタッフが採用したコーディング規格を記載しています。『Oracle Applications User Interface Standards for Forms-Based Products』で記述している Oracle Applications ユーザー・インタフェースを実装するのに必要な Oracle Applications Object Library コンポーネントを説明します。また、Oracle Applications と統合するために、独自の Oracle Forms Developer 6*i* フォームを作成する場合に有益な情報も提供します。

『Oracle Applications User Interface Standards for Forms-Based Products』

このマニュアルは、Oracle Applications 開発スタッフが採用したユーザー・インタフェース（UI ）規格を記載しています。Oracle Applications 製品の UI と Oracle Forms で作成したアプリケーションの設計への UI の適用方法について説明しています。

その他の実装マニュアル

『Oracle Applications における複数報告通貨』

取引を複数通貨で記録するために複数報告通貨機能を使用する場合は、Oracle Telephony Adapter SDK を実装する前にこのマニュアルを参照してください。このマニュアルには、この機能を Oracle Telephony Adapter SDK とともに実装するための詳細な追加手順と設定に関する考慮事項が記載されています。

『Oracle Applications における複数組織』

このマニュアルでは、Oracle Telephony Adapter SDK とともに Oracle Applications の複数組織サポート機能を設定して使用する方法を説明します。これにより、Oracle Telephony Adapter SDK の単一インストールを実行する場合に様々な組織体系を定義してサポートできます。

『Oracle Workflow ユーザーズ・ガイド』

このマニュアルでは、既存の Oracle Applications 組込みワークフロー・プロセスをカスタマイズする方法と、新しいワークフロー・ビジネス・プロセスを定義する方法について説明します。ワークフロー対応プロセスを含む Oracle Applications 製品に必要な設定手順を完了するためにも、このマニュアルを使用します。

『Oracle Applications フレックスフィールド・ガイド』

このマニュアルでは、Oracle Telephony Adapter SDK 実装チーム、ならびに Oracle Applications 製品データの継続的な保守を担当するユーザーを対象とした、フレックスフィールド計画、設定および参照情報を提供します。また、このマニュアルでは、フレックスフィールド・データについてのカスタム・レポートの作成に関する情報を提供します。

Oracle eTechnical Reference Manuals

各 eTechnical Reference Manual（eTRM）には、特定の Oracle Applications 製品のデータベース・ダイアグラムと、データベース表、フォーム、レポートおよびプログラムの詳細が記載されています。この情報は、既存のアプリケーションからのデータの変換、Oracle Applications データと非 Oracle Applications の統合、および Oracle Applications 製品のカスタム・レポートの作成に役立ちます。Oracle eTRM は、オラクル・サポート・ホームページで入手できます。

『Oracle Manufacturing APIs and Open Interfaces Manual』

このマニュアルには、他の Oracle Manufacturing アプリケーションや他のシステムとの統合に関する最新情報が記載されています。また、Oracle Manufacturing の API とオープン・インタフェースについても説明しています。

『Oracle Order Management Suite API およびオープン・インタフェース・マニュアル』

このマニュアルには、他の Oracle Manufacturing アプリケーションや他のシステムとの統合に関する最新情報が記載されています。また、Oracle Order Management Suite の API とオープン・インタフェースについても説明しています。

『Oracle Applications メッセージ・リファレンス・ガイド』

このマニュアルでは、Oracle Applications メッセージについて説明します。このマニュアルは、リリース 11*i* のドキュメント CD-ROM に HTML 版で用意されています。

『Oracle CRM Application Foundation インプリメンテーション・ガイド』

CRM Application Foundation のコンポーネントは多数の CRM 製品で使用されます。このマニュアルを使用して、CRM Application Foundation を正常に実装します。

データベース・ツールによる Oracle Applications データの変更の禁止

マニュアルに指示がない限り、SQL*Plus、Oracle Data Browser、データベース・トリガーまたは他のツールを使用して、Oracle Applications データを変更しないことをお薦めします。

オラクル社は、Oracle データベースでの情報の作成、記憶、変更、検索または保守に使用できる強力なツールを提供します。しかし、SQL*Plus のような Oracle ツールを使用して Oracle Applications データを変更すると、データの整合性が損われたり、データの変更を監査できなくなるおそれがあります。

Oracle Applications の表は相互に関連付けられているので、Oracle Applications を使用して表を変更すると、一度に多数の表が更新されます。しかし、Oracle Applications を使用せずに Oracle Applications データを変更すると、1 つの表のある行を変更した場合に、関連する表で反映させるべき変更が行われないことがあります。各表をお互いに同期させていないと、誤った情報を検索したり、Oracle Applications で予測できない結果が生ずるおそれがあります。

Oracle Applications を使用してデータを変更すると、その変更が有効であるかどうかが Oracle Applications により自動的にチェックされます。Oracle Applications は、情報を変更したユーザーを記録することもできます。データベース・ツールを使用してデータベース表に情報を入力すると、無効な情報が格納されることがあります。また、SQL*Plus や他のデータベース・ツールを使用したときは、変更履歴が残らないため誰が情報を変更したのか追跡することができなくなります。

オラクル社について

オラクル社は、財務管理、サプライ・チェーン管理、製造、プロジェクト・システム、人事管理および顧客関係管理の 160 を超すソフトウェア・モジュールの統合スイートである Oracle Applications のみでなく、データベース管理、アプリケーション開発、意思決定サポートおよびオフィス・オートメーションといった統合ソフトウェア製品ラインの開発と販売を行っています。

Oracle 製品は、メインフレーム、ミニコンピュータ、パーソナル・コンピュータ、ネットワーク・コンピュータおよびパーソナル・デジタル・アシスタントで使用可能で、組織は、様々なコンピュータ、オペレーティング・システム、ネットワークおよびデータベース管理システムを、単一かつ統一されたコンピューティングおよび情報リソースに統合できます。

オラクル社は、情報管理において世界最大のソフトウェア・サプライヤであり、ソフトウェア会社としても世界第 2 位の会社です。オラクル社は、世界 145 か国以上で、データベース、ツールおよびアプリケーション製品を、関連するコンサルティング、研修およびサポート・サービスとともに提供しています。

1

概要

1.1 Oracle Interaction Center と Oracle Advanced Inbound

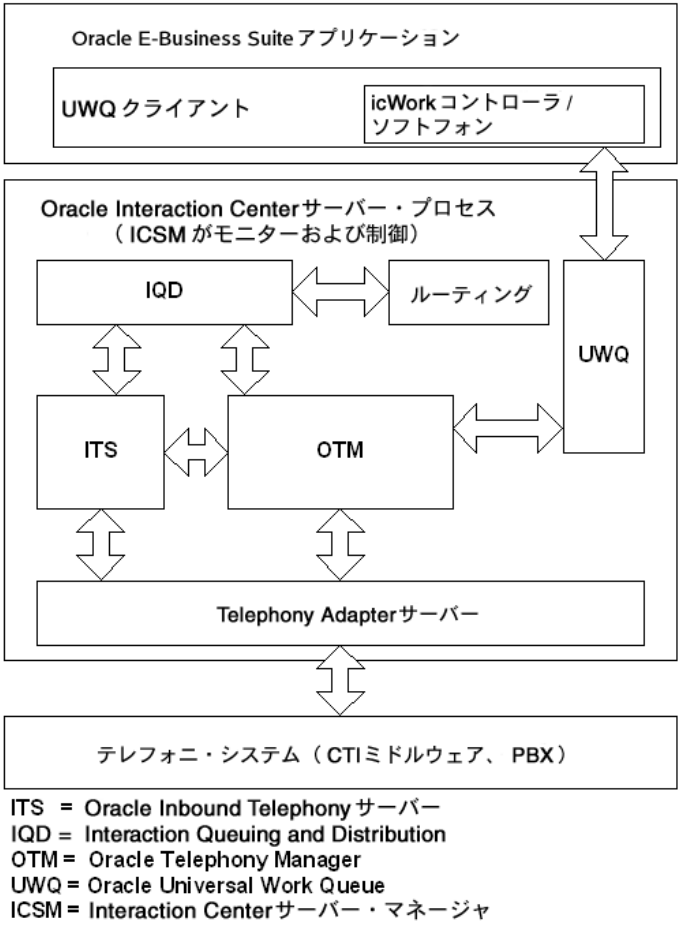
Oracle Interaction Center（OIC）は、Oracle の E-Business アプリケーション・パッケージのうちテレフォニを使用可能にする基盤となる製品ファミリです。Oracle Advanced Inbound は、インバウンド・コール・センター向けに特化された Oracle Interaction Center 製品であり、次の主要機能を備えています。

- エージェントのソフトフォン
- エージェントのスクリーンポップ
- 仮想キューイング
- コール・ルーティング
- コールおよびデータ転送

Oracle Advanced Inbound は、次を含むサーバー・プロセスのグループで構成されています。

- Inbound Telephony サーバー（ITS）。ACD キューとルート・ポイントに到達する着信をモニターします。
- Interaction Queueing and Distribution（IQD）。エージェントごとにメディア項目の仮想キューを保守します。
- Routing サーバー（RS）。コールをルーティングし、分類します。
- Oracle Telephony Adapter サーバー（OTAS）。テレフォニ・プラットフォーム固有のメッセージとイベントを正規化し、Oracle Interaction Center と基礎となる CTI プラットフォームの間に単一のインタフェースを提供します。
- Oracle Telephony Manager（OTM）。エージェントの内線をモニターし、制御します。
- Oracle Universal Work Queue サーバー（UWQ）。エージェントごとに仮想作業キューを保守し、Oracle Interaction Center と Oracle E-Business Suite アプリケーションの間に単一のインタフェースを提供します。

次の図に、Oracle Advanced Inbound のコンポーネント間の関係を示します。



この図のように、Telephony Adapter サーバーはテレフォニ・システムと Oracle Advanced Inbound の中間に位置します。また、Oracle Universal Work Queue は、Oracle Advanced Inbound と、icWork コントローラやソフトフォンのようなエージェントのクライアント・システムの中間に位置します。

1.2 Oracle Telephony Adapter SDK の概要

Oracle Telephony Adapter SDK（Software Development Kit）は、Oracle Advanced Inbound で直接サポートされないテレフォニ・プラットフォームを統合する手段をコンサルタントやパートナに提供します。現場のチームは、Oracle Telephony Adapter SDK を使用して、Telephony Adapter サーバーへのプラグイン・アダプタを開発し、顧客のテレフォニ・プラットフォームに対する CTI の統合をサポートできます。

Oracle Telephony Adapter SDK を構成するコンポーネントは、次のとおりです。

- Oracle Telephony Adapter API。アダプタによる実装が必要で Oracle Advanced Inbound からコールされるインタフェースと、Oracle Advanced Inbound により実装されアダプタからコールされるインタフェースです。
- 『Oracle Telephony Adapter SDK 開発者リファレンス・ガイド』（このマニュアル）。API を使用してアダプタの実装と配布、アダプタ実装のテストなどを行う方法について説明します。
- アダプタ実装をテストして検証するための検証ツール。

リリース 11.5.8 の Oracle Telephony Adapter SDK には、次の機能が用意されています。

- Oracle Advanced Inbound への PBX または ACD とサード・パーティの CTI ミドルウェアの統合
- コンサルタント、パートナ、PBX/ACD や CTI ミドルウェアのベンダーが Oracle Interaction Center（OIC）への統合を実装できるようにオープン・インタフェースを提供する Java API
- API として C しか使用できないスイッチおよび CTI ミドルウェアと統合するための C API
- コンサルティング・チームに各統合のテスト手段を提供するドキュメント、ユニット・テスト装置およびテスト・ケース一式
- ログイン、ログアウト、通話可、通話不可、転送、会議、応答、切断、保留および再接続の CTI 機能
- Oracle Applications 内の顧客データのスクリーンポップ
- コール目的を識別するコール分類
- CTI の使用が可能なパッシブ・モードによる、Oracle Advanced Inbound と通話の PBX ベース・ルーティング、キューイングおよび分配
- Oracle Advanced Inbound を次の 2 つのモードで統合して構成するためのオプション
  - コールのインテリジェント・ルーティング、キューイングおよび分配を行うアクティブ・モード
  - OIC でのキュー件数やその他のレポート機能など、ACD ベースのコール・ルーティングについて ACD ルート・ポイントをモニターする拡張パッシブ・モード

- 手動、プレビューまたはプログレシブ・ダイヤル・モードに対する Oracle Advanced Outbound サポート
- 完全なソフトフォン機能セット
- コールとエージェントのライフ・サイクル・セグメントなど、顧客対応履歴内の拡張コールおよびエージェント・データと、転送 / 会議の追跡
- ルーティングおよびキュー・セグメントや放棄されたコール・データなど、アクティブ・モードまたは拡張パッシブ・モードによる完全な顧客対応履歴

Oracle Telephony Adapter SDK の将来のリリースでは、統合レベルが基本レベル、標準レベルおよび拡張レベルの 3 レベルになり、様々な統合プロジェクトの複雑度に応じて機能のサブセットを統合するためのオプションが提供されます。

1.3 SDK プロジェクトのアプローチ

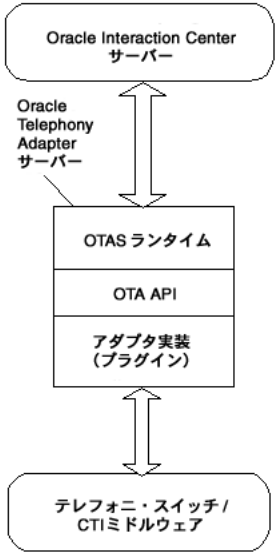
SDK プロジェクトを計画する手順の概略は次のとおりです。

1. SDK 実装プロジェクトの条件を設定します。付録 A「SDK 条件設定ワークシート」を参照してください。
2. プロジェクトの範囲を設定し、プログラマに概略を説明します。付録 B「SDK のスコープ分析」を参照してください。
3. Telephony Adapter を開発します。C および Javadoc API については、第 5 章「Telephony Adapter の開発」を参照してください。
4. アダプタのユニット・テストを実施します。付録 C「Telephony Adapter のテスト・ケース」および第 6 章「Telephony Adapter のテスト」を参照してください。
5. アナリストが成果物を検討して承認します。
6. Telephony Adapter を配布します。第 7 章「Telephony Adapter の配布」を参照してください。

2

主な機能

Oracle Telephony Adapter サーバー（OTAS）は、通信コア（OTAS ランタイム）、Oracle Telephony Adapter API（OTA API）および OTA API への実装と書込みを行うアダプタ・コードで構成されています。



この図のように、Oracle Telephony Adapter サーバーは、OTAS ランタイム、OTA API およびアダプタ実装（プラグイン）で構成されています。OTAS により、テレフォニ・スイッチおよび CTI ミドルウェアが Oracle Interaction Center に接続されます。

2.1 アダプタ実装

アダプタ実装はプラットフォーム固有のカスタム・コードであり、テレフォニ・プラットフォーム（PBX/ACD および CTI ミドルウェア）の API を Oracle Telephony Adapter API にマップします。

2.2 Oracle Telephony Adapter API のオブジェクト

Oracle Telephony Adapter API では、アダプタ・コードで実装して通信時に経由する必要がある 3 つのオブジェクトが指定されます。これらのオブジェクトは、TeleDeviceFactory、TelesetDevice および RoutePointDevice です。

2.2.1 TeleDeviceFactory

Oracle では、TeleDeviceFactory オブジェクトがスイッチ・リソースの割当てと割当て解除に使用されます。現行リリースの場合、割当てできるのは Teleset リソースと Route Point リソースのみです。

2.2.2 TelesetDevice

テレセット・デバイスは、エージェントにより使用される実際の電話を表します（現在はデジタル・テレセットのみで、アナログ・テレセットは含まれません）。エージェントはテレセットにログインし、このオブジェクトを使用して、コールの実行、保留設定および切断など、各種のコール制御機能を実行します。また、テレセット・デバイスは、状態の変化やコールの進行を示すスイッチ・イベントを、テレフォニ・プラットフォームから Oracle Interaction Center に中継します。

拡張レベルの統合の場合、テレセット・デバイスには次の特性があります。

- 各テレセットについて、Oracle ソフトフォンでは 3 回線が表示されます。
- 各回線で 1 つのコールを保持できます。
- テレフォニ・プラットフォームによっては、物理テレセット・マップ上の 1 つのキーが、Oracle ソフトフォン上の 1 つ以上の回線にマップします。たとえば、Avaya Definity の場合は、1 つの物理テレセットが同じ局番を共有する 3 つのキーでプログラムされます。この場合、Oracle ソフトフォンでは、1 つの回線が Avaya 物理テレセットの 1 つのキーを表します。Nortel Meridian の場合、1 つの物理テレセットは Personal DN および ACD DN という 2 つのキーでプログラムされます。この場合、Oracle ソフトフォンでは、回線 1 は Personal DN キー、回線 2 は ACD DN キー、回線 3 はどちらかのキーからのコンサルタント・コールを表します。
- Oracle Interaction Center で発生するほとんどのイベントでは、イベントが発生する回線（回線インデックス 0、1 または 2）を指定する必要があります。

- ほとんどのコール制御コマンドは、holdCall など、API コールで指定された回線で実行されます。API は makeCall をコールし、consultationCall は、異なる回線でイベントを戻すことができます。
- コンサルタント・コールは、一度に 1 つのみ実行できます。したがって、保留にできる転送または会議は 1 つのみです。

2.2.3 RoutePointDevice

ルート・ポイント・デバイスは、スイッチ・キューを表します。スイッチ・キューには、Oracle Advanced Inbound がコールをアクティブにルーティングする制御対象キューと、スイッチが実際にコールをルーティングする時にキュー内のコールの追跡のみを行うモニター対象キューがあります。

ルート・ポイント・デバイスの特性は、次のとおりです。

- 各ルート・ポイントには、保留コールのキューがあります。
- キュー内の各コールはコール識別子で識別されます。これは通常、テレフォニ・プラットフォームから割り当てられるコール ID です。
- コール識別子はコール制御コマンドで使用されます。発生するイベントは、コール識別子を含む必要があります。
- ブラインド転送が原因でコール ID が変化しても、対応するコール識別子は変化しません。

2.3 コール・データ

コール・データは、コールに関連付けられたデータ、特に ANI、DNIS および IVR データです。このデータは、Oracle Interaction Center でコールを適切に分類してルーティングするために使用され、スクリーンポップのベースにもなります。アダプタは、コールの開始時または開始直後にテレフォニ・プラットフォームからのコール・データを予期します。

2.4 メディア項目 ID

Oracle Interaction Center では、メディア項目識別子（メディア項目 ID）で識別される論理的な顧客対応が追跡されます。スイッチ固有のコール ID はコールの転送時に変更されることがありますが、メディア項目 ID は同一である必要があります。メディア項目 ID が Oracle により割り当てられるときに、実装者はエージェント間でコールが移動するにつれてメディア項目 ID が伝播することを確認する必要があります。ある回線で転送または会議が完了したためにコール ID が変化しても、対応するメディア項目 ID は変化しません。

2.4.1 メディア項目 ID モデル

メディア項目 ID モデルの特性は、次のとおりです。

注意： 次の例には、すべてのイベントが示されているわけではありません。

表 2-1 エージェント A がエージェント B をコールする場合

Oracle Interaction Center での処理	アダプタ・サーバーでの処理 / 状態	Oracle Interaction Center の状態
A: makeCall（MI: 1、宛先 : B）[5]	A: ダイヤル（宛先 : B）が結果コール ID 1 を MI 1[1] にマップ	
	A: beginCallEvent（MI: 1、回線 : 0）[2]	A: 回線 0 の MI は 1[1]
	B: beginCallEvent（MI: 1、回線 : 0）[2]	B: 回線 0 の MI は 1[1]
A: releaseCall（回線 : 0）	A: 切断（コール ID 1）	
	A: callReleasedEvent（回線 : 0）	A: 回線 0 は消去
	B: callReleasedEvent（回線 : 0）	B: 回線 0 は消去

表 2-2 顧客 X がエージェント A をコールする場合

顧客による処理	アダプタ・サーバーでの処理 / 状態	Oracle Interaction Center の状態
X: A をコール	A: beginCallEvent（MI: 該当せず、回線 : 0）、OTAS[3] は MI[1] を認識せず	A: 回線 0 の MI は 2[4]、OIC はコールにメディア項目 ID を割当て
X: コールを切断	A: callReleasedEvent（回線 : 0）	A: 回線 0 は消去

表 2-3 顧客 X がエージェント A をコンサルタント・コールする場合

顧客 /OIC による処理	アダプタ・サーバーでの処理 / 状態	Oracle Interaction Center の状態
X: A をコール	A: beginCallEvent（MI: 該当せず、回線 : 0）、コール ID は 2、OTAS[3] は MI[1] を認識せず	A: 回線 0 の MI は 3[4]、OIC はコールにメディア項目 ID を割当て
A: コンサルタント（MI[1]: 3、回線 : 0、宛先 : B）[5]	A: コンサルタント（コール ID: 2、宛先 : B）、コール ID 2 を MI[1] 3 にマップ、コンサルタント・コール ID 3 を MI 3 にマップ	
	A: beginCallEvent（MI: 3、回線 : 1）[2]	A: 回線 1 の MI は 3
	B: beginCallEvent（MI: 3、回線 : 0）[2]	B: 回線 0 の MI は 3
A: 転送（回線 : 1、旧回線 : 0）	A: 転送（コール ID 3、コール ID 2）	
	A: callReleasedEvent（回線 : 0）	A: 回線 0 は消去
	A: callReleasedEvent（回線 : 1）	A: 回線 1 は消去
	B のコール ID が変化（ABA モデル）	OIC に対するイベントはなし[6]

脚注

1. メディア項目 ID
2. 着信にすでにメディア項目 ID がある場合は、それを受信デバイスに渡す必要があります。
3. Oracle Telephony Adapter サーバー
4. 着信にメディア項目 ID がない場合は、Oracle Interaction Center により新規メディア項目 ID が割り当てられ、それが以降のコンサルタント・コールに使用されます。
5. 新規コールには、ダイヤル・コマンドの実行時に新規メディア項目 ID が割り当てられます。
6. メディア項目 ID とスイッチ固有のコール ID には、1 対多の関係があります。実際のコールの転送中または会議中にコール ID が変化しても、メディア項目 ID は変化しません。

3

開発環境

Oracle Telephony Adapter SDK の要件は、次のとおりです。

3.1 最低限のハードウェア要件

Oracle Telephony Adapter SDK のハードウェア要件は、次のとおりです。

開発マシン

- CPU: 500MHz
- メモリー（RAM）: 256MB
- ディスク領域 : 10G
- オペレーティング・システム : C API を使用する場合は Windows NT 4.0 SP6 または Windows NT 2000、Java API を使用する場合は Linux および UNIX
- テレフォニ接続に必要なハードウェア

テレフォニ・ハードウェア

- PBX
- テレセット

3.2 最低限のソフトウェア要件

Oracle Telephony Adapter SDK のソフトウェア要件は、次のとおりです。

全般

- Java Development Kit リリース 1.1.8、1.2 または 1.3
- （オプション）Oracle JDeveloper など、Java 用の統合開発環境（IDE）

Java 用 SDK

サード・パーティの CTI Java ライブラリ

C 用 SDK

- サード・パーティの Windows NT および Windows 2000 用 CTI ソフトウェア
- C コンパイラおよび make ユーティリティ
- （オプション）Visual C++ など、Windows 用の統合開発環境（IDE）

4

Oracle Telephony Adapter SDK のコンポーネント

Oracle Telephony Adapter SDK には、C および Java プログラム言語用の API が用意されています。

Oracle Telephony Adapter の Java API に用意されているオープン・インタフェースにより、コンサルタント、スイッチ・ベンダーおよび CTI ミドルウェア・ベンダーは Java プログラム言語を使用して、任意のテレフォニ・プラットフォームについて Oracle Interaction Center（OIC）への統合を実装できます。

Oracle Telephony Adapter の C API により、プログラマは C プログラム言語を使用してスイッチと CTI ミドルウェアを統合できます。

Oracle Telephony Adapter の C API は、次の API とファンクションで構成されています。

- TeleDeviceFactory（TeleDeviceFactoryAPI.h）、TelesetDevice（TelesetDeviceAPI.h）および RoutePointDevice（RoutePointDeviceAPI.h）用のヘッダー・ファイル
- EventListener API（TelesetEventListenerAPI.h および RouteEventListenerAPI.h）
- 共通ユーティリティのファンクション、型定義および定数（occt_pub.h、occt_md.h）
- OHashtable ファンクション（Hashtable.h）

C アダプタの実装者は、ヘッダー・ファイルに定義されたすべてのメソッドを実装します。

この項の内容は、次のとおりです。

- Oracle Telephony Adapter SDK のインストール
- パッケージの内容

4.1 Oracle Telephony Adapter SDK のインストール

次の手順に従って Oracle Telephony Adapter SDK をインストールします。

前提条件

Oracle Telephony Adapter SDK をインストールする前に、Java Development Kit または Java Runtime Environment をインストールする必要があります。JDK バージョン 1.3 を使用することをお薦めします。

cctsdk.zip のダウンロード

Telephony Adapter は単一の zip ファイル cctsdk.zip としてパッケージされています。このファイルの入手方法は、オラクル社にお問い合せください。

解凍

Zip ファイル抽出プログラムを使用し、空白を含めずに指定したディレクトリ・パスに cctsdk.zip を解凍します。cctsdk.zip の解凍後に、ファイル oracle/apps/cct/bin/sdkenv.cmd を編集し、環境変数 SDK_JRE_HOME の値を JDK のインストール位置に変更します。JDK のインストール先ディレクトリのパスは、空白を含めずに指定してください。

4.2 パッケージの内容

Oracle Telephony Adapter SDK パッケージの cctsdk.zip には、次のファイルが含まれています。

表 4-1 SDK パッケージ内のファイル

ディレクトリ	説明
bin	アダプタ・サーバーとテスト・ツールの起動用スクリプト
lib	オラクル社提供の Java および C ランタイム・ライブラリ
docs	ドキュメント
c	C 用 SDK
c/include	C 用 SDK: C のインクルード・ファイル
c/lib	C 用 SDK: オラクル社提供の C アダプタ開発用 C ライブラリ
c/impl	C 用 SDK: C アダプタ実装の格納用ディレクトリ
c/sample	C 用 SDK: C のサンプル・コード
java	Java 用 SDK

表 4-1 SDK パッケージ内のファイル（続き）

ディレクトリ	説明
java/javadoc	Java 用 SDK: Javadoc
java/classes	Java 用 SDK: Java アダプタ実装のクラス・ファイル格納用ディレクトリ
java/sample	Java 用 SDK: サンプル・コード
3rdParty	スイッチ・ベンダーおよび CTI ミドルウェア・ベンダーが提供する Java および C ライブラリの格納用ディレクトリ

4.2.1 Java 用 SDK

Java 用 SDK により、開発者は Java プログラム言語を使用してアダプタを実装できます。CT Connect や JTAPI など、統合対象のスイッチおよび CTI ミドルウェアからの Java API を使用できる場合は、Java 用 SDK を使用する必要があります。

Java 用 SDK を使用して実装するアダプタは、JAR ファイルとしてパッケージされています。

4.2.2 C 用 SDK

C プログラム言語用の SDK を使用する場合、開発者がアダプタを実装できるのは Windows NT プラットフォームと Windows 2000 プラットフォームのみです。Cisco ICM や TAPI など、統合対象のスイッチおよびミドルウェアから使用できるオプションが C の API のみの場合は、C 用の SDK を使用する必要があります。Windows NT および Windows 2000 用の SDK は、スイッチと CTI ミドルウェアへの統合を実装する C 関数のセットです。これらの関数は、1 つのアダプタとしてまとめてパッケージされています。

Windows NT および Windows 2000 用の SDK を使用して実装するアダプタは、Windows DLL（動的ロード・ライブラリ）ファイルとしてパッケージされています。

4.2.3 Oracle Telephony SDK 統合テスト・ユーティリティ

Oracle Telephony SDK 統合テスト・ユーティリティは、Telephony Adapter を実行、構成およびテストするためのユーザー・インタフェースです。このユーティリティを使用すると、開発者はアダプタ実装をオフライン・モードでテストできます。オフライン・モードでは、Oracle Applications データベースや Oracle Interaction Center サーバーを必要としません。ただし、テスト・ユーティリティを使用するには、Telephony Adapter をテレフォニ・プラットフォームに接続する必要があります。テスト・ユーティリティを使用するのは主としてアダプタ開発の初期段階であり、この段階では開発者が Oracle Applications データベースに接続せずに各自の開発ワークステーションで作業できます。統合テスト・ユーティリティには、3 つのコンポーネントである Oracle Telephony Adapter サーバー、TelesetDevice テスト・ユーティリティおよび Telephony Adapter 検証ツールが組み込まれています。

4.2.3.1 Oracle Telephony Adapter サーバー

テスト・ユーティリティには、Oracle Telephony Adapter サーバー（OTAS）が組み込まれています。開発者は、OTAS を使用してアダプタ実装を実行し、テストできます。

4.2.3.2 TeleDevice テスト・ユーティリティ

TeleDevice テスト・ユーティリティには、TelesetDevice オブジェクトと RoutePointDevice オブジェクトをテストするためのユーザー・インタフェースが用意されています。開発者は、このユーティリティを使用してソフトフォンを起動し、それを使用してテスト・ケースを実行できます。

4.2.3.3 Telephony Adapter 検証ツール

Telephony Adapter 検証ツールを使用すると、開発者は事前定義済みのテレフォニ・テスト・ケース・セットと対照してアダプタ実装を検証できます。

5

Telephony Adapter の開発

この章では、Oracle Telephony Adapter SDK の開発プロセスについて説明します。この章の内容は、次のとおりです。

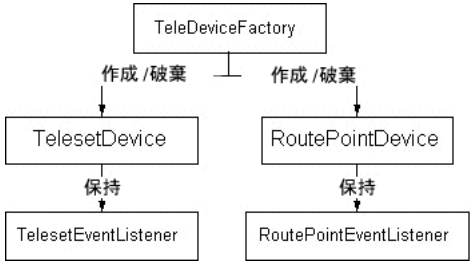
- 主要インタフェースの概要
- 開発プロセスの概要
- コール使用例
- Oracle Telephony Adapter のライフ・サイクル
- TeleDeviceFactory の実装
- TelesetDevice の実装
- RoutePointDevice の実装
- OTAS メッセージ・サービス
- Java のその他 API
- Windows NT と Windows 2000 のその他 API

5.1 主要インタフェースの概要

Oracle Telephony Adapter SDK は、次の 5 つの主要インタフェースで構成されています。

- TeleDeviceFactory
- TelesetDevice
- RoutePointDevice
- TelesetEventListener
- RoutePointEventListener

次の図に、Oracle Telephony Adapter SDK の主要インタフェースを示します。



Java 用 SDK

- Java 抽象クラス AbstractTelesetDevice および AbstractRoutePointDevice を拡張するクラスを作成し、Java インタフェース TeleDeviceFactory を実装するクラスを作成します。
- すべてのクラスを JAR ファイルとしてパッケージします。

C 用 SDK

- TeleDeviceFactoryAPI.h、TelesetDeviceAPI.h および RoutePointDeviceAPI.h に定義されている C 関数の実装を作成します。
- すべての関数を DLL ファイルにパッケージします。

5.2 開発プロセスの概要

次の手順リストでは、Oracle Telephony Adapter SDK の開発の概要について説明します。手順 2 および 7 は繰り返して行います。

1. Oracle Telephony Adapter SDK のインストール
2. 開発
3. ビルド
4. Oracle Telephony Adapter サーバーの実行
5. テストおよび再開発
6. Interaction Center へのアダプタ実装の配布
7. テストと Interaction Center の統合および再開発
8. 本稼働

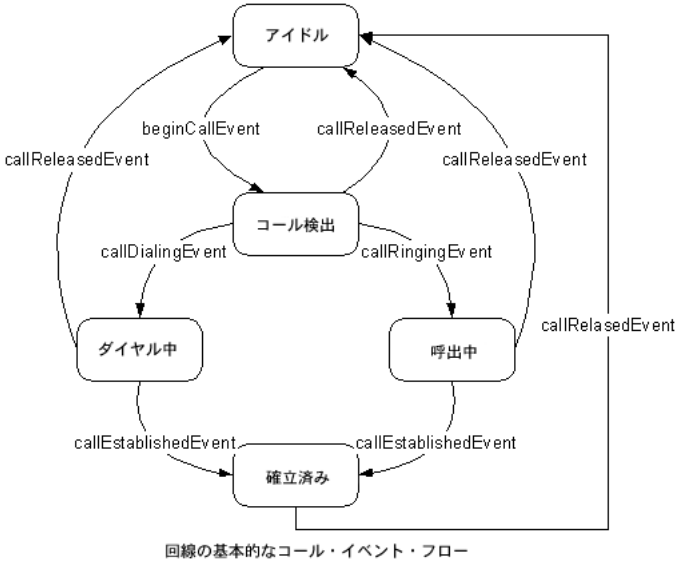
5.3 コール使用例

次のコール使用例では、TelesetEventListener インタフェースでの API コールの典型的なコール・フロー・イベントについて説明します。

5.3.1 基本的なインバウンド・コール・フロー

次の図に、回線の基本的なコール・イベント・フローを示します。

図 5-1 回線の基本的なコール・イベント・フロー

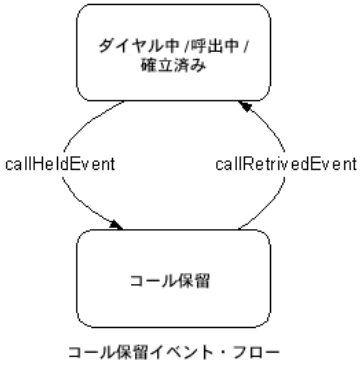


前述の図は、顧客やエージェントから直接またはルート・ポイントを介して着信する場合の、回線の基本的なコール・イベント・フローを示しています。アイドル状態から、beginCallEvent が着信を検出します。callDialingEvent がダイヤルを起動し、callRingEvent が受信回線で呼出しを起動します。コールは callEstablishedEvent を使用して受信回線で確立されます。一方のパーティが切断すると、callReleasedEvent がコールを終了します。

5.3.2 コール保留イベント・フロー

次の図に、コール保留イベント・フローを示します。

図 5-2 コール保留イベント・フロー



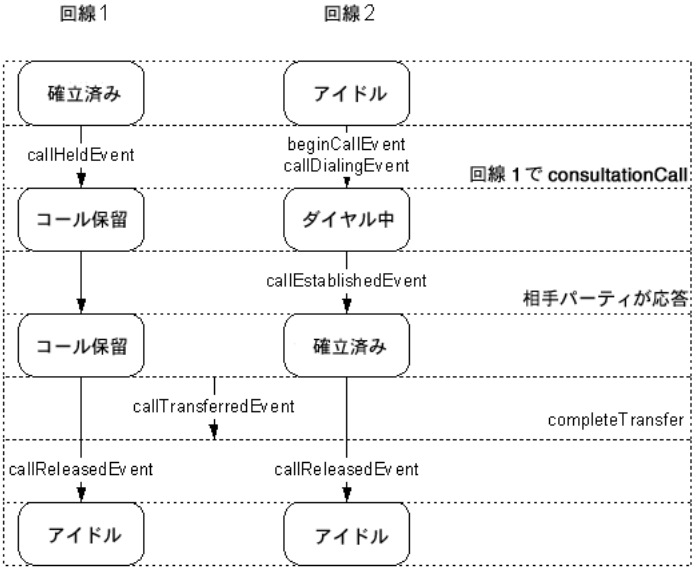
前述の図は、一方のパーティが他方のパーティを保留にして再接続できるコールを示しています。このコールには、顧客とエージェントの間のコールと、エージェント間のコールがあります。

ダイヤル中、呼出中または確立済み状態のコールは、保留にすることができます。callHeldEvent が起動側に対して起動されます。コールが保留から再接続されると、起動側に対して callRetrievedEvent が起動されます。

5.3.3 2 ステップ転送イベント・フロー

次の図に、2 ステップ転送イベント・フローを示します。

図 5-3 2 ステップ転送イベント・フロー



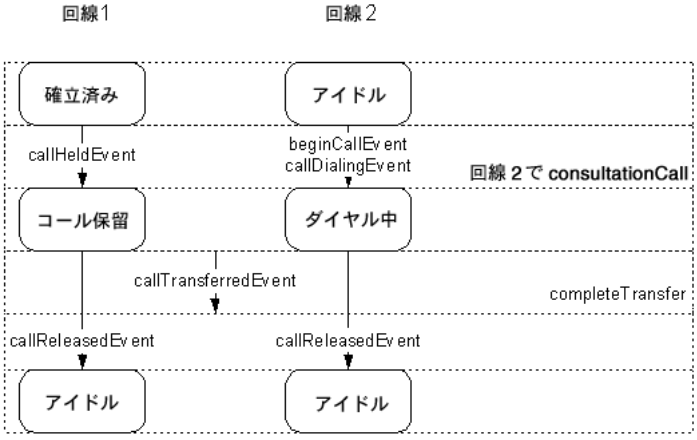
2ステップ・コール転送イベント・フロー

前述の図のように、コンサルト転送では制御エージェントが別のエージェントにコールを転送し、転送完了前にそのエージェントと通話できます。制御エージェントは回線 1 で callHeldEvent を受信し、回線 2 で beginCallEvent および callDialingEvent を受信します。相手エージェントが応答すると、制御エージェントは回線 2 で callEstablishedEvent を受信します。次に、コールの転送を完了し、callTransferredEvent を受信します。転送完了後に、回線 1 で callReleasedEvent を受信し、回線 2 でもう 1 つ callReleasedEvent を受信します。

注意： スイッチと CTI ミドルウェアの組合せによっては、初期コール ID と最後のコール ID が同じになります。たとえば、連番 ABA の場合、初期コール ID は A、コンサルタント・コールのコール ID は B で、転送済みコールの最後のコール ID も A です。スイッチと CTI ミドルウェアの組合せによっては、転送済みコールのコール ID がコンサルタント・コールのコール ID のままで、連番 ABB となる場合もあります。

5.3.4 ブラインド転送イベント・フロー

次の図に、ブラインド転送イベント・フローを示します。



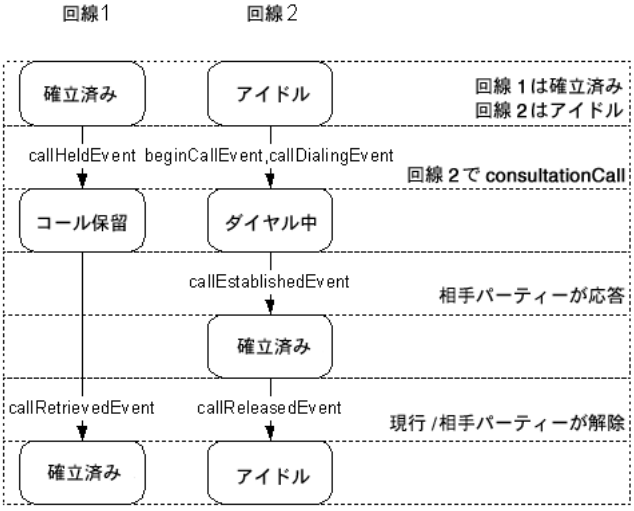
2 ステップ・ブラインド転送イベント・フロー

前述の図のように、ブラインド転送では、エージェントは他方のエージェントと通話せずに、回線 1 でそのエージェントにコールを転送します。制御エージェントは回線 1 で callHeldEvent を受信し、回線 2 で beginCallEvent および callDialingEvent を受信します。その後、コールの転送を完了し、callTransferredEvent を受信します。転送完了後に、回線 1 で callReleasedEvent を受信し、回線 2 でもう 1 つ callReleasedEvent を受信します。

5.3.5 コンサルト取消イベント・フロー

次の図に、コンサルト取消イベント・フローを示します。

図 5-4 コンサルト取消（転送または会議コール）



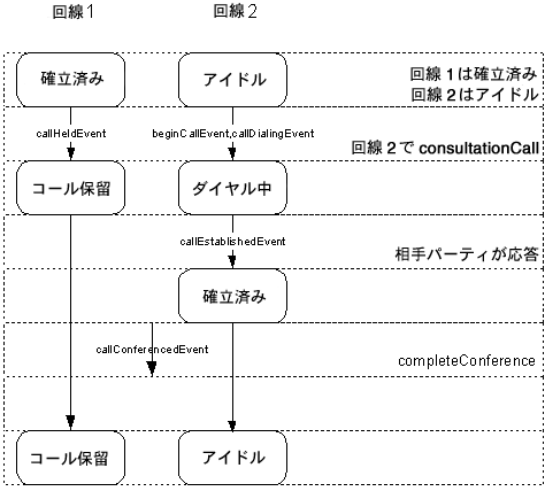
コンサルト取消（転送または会議）

前述の図のように、コンサルト転送では制御エージェントが別のエージェントにコールを転送し、転送完了前にそのエージェントと通話できます。制御エージェントは回線 1 で callHeldEvent を受信し、回線 2 で beginCallEvent および callDialingEvent を受信します。相手エージェントが応答すると、制御エージェントは回線 2 で callEstablishedEvent を受信します。次に、転送を取り消し、回線 2 で callReleasedEvent、回線 1 で callRetrievedEvent を受信します。この時点で、エージェントはオリジナルのコール元に戻ります。

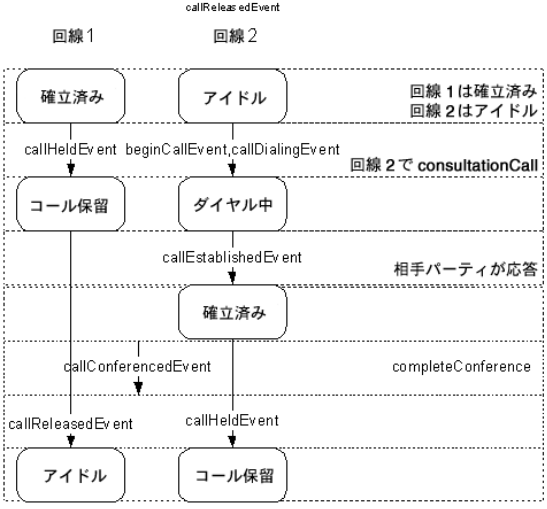
5.3.6 アクティブでない回線を使用した 2 ステップ会議イベント・フロー

次の図に、アクティブでない回線を使用した 2 ステップ会議イベント・フローを示します。

図 5-5 アクティブでない会議回線を使用したコール会議イベント・フロー



コール会議イベント・フロー（ ABA、アクティブでない会議回線）



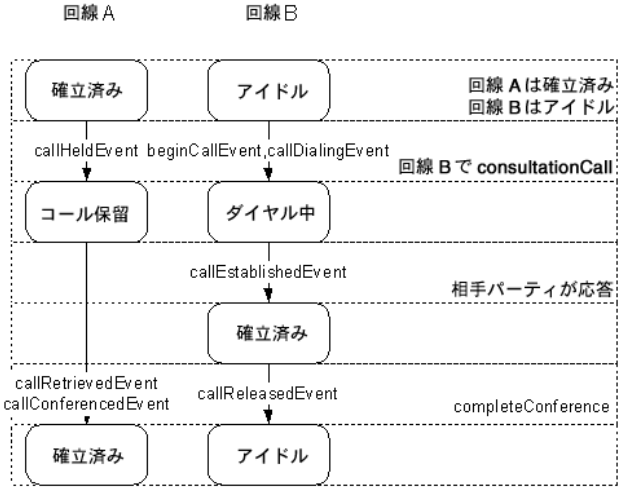
コール会議イベント・フロー（ ABB、アクティブでない会議回線）

コンサルト会議中に、制御エージェントは別のエージェントとコールに対して会議を行い、会議完了前にそのエージェントと通話できます。制御エージェントは回線 1 で callHeldEvent を受信し、回線 2 で beginCallEvent および callDialingEvent を受信します。相手エージェントが応答すると、回線 2 で callEstablishedEvent を受信します。その後、制御エージェントは会議を完了し、callConferencedEvent を受信します。会議完了後に、回線 1 または回線 2 で callReleasedEvent を受信します。

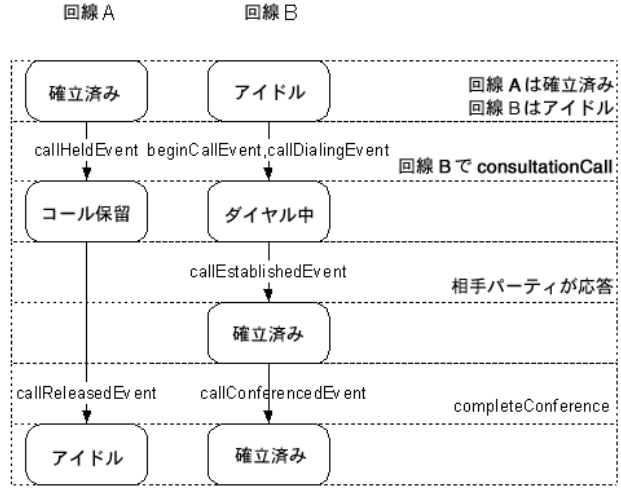
5.3.7 コール会議イベント・フロー

次の図に、アクティブな回線を使用したコール会議イベント・フローを示します。

図 5-6 アクティブな会議回線を使用したコール会議イベント・フロー



コール会議イベント・フロー（ ABA、アクティブな会議回線）



コール会議イベント・フロー（ ABB、アクティブな会議回線）

前述の図のように、コール会議イベント・フローはアクティブでない回線を使用した 2 ステップ会議イベント・フローと同様です。標準的なコール会議イベント・フローの場合、制御エージェントは他のパーティに対してコンサルタント・コールを実行し、会議完了前にそのパーティと対話できます。コンサルタント・コールの実行後に、制御エージェントは回線 1（オリジナル・コール）で callHeldEvent を受信し、回線 2（コンサルタント・コール）で beginCallEvent および callDialingEvent を受信します。相手パーティが応答すると、制御側は回線 2 で callEstablishedEvent を受信します。この時点で制御側が会議を完了すると、一方の回線で callReleasedEvent、他方の回線で callConferencedEvent を受信します。コールが保留になっている場合、callConferencedEvent はコールが再接続されることを暗黙的に示します。

5.4 Oracle Telephony Adapter のライフ・サイクル

この項では、Oracle Telephony Adapter の作成および破棄ライフ・サイクルの Java および C 用 SDK について説明します。この項の内容は、次のとおりです。

- Oracle Telephony Adapter サーバーの起動
- TelesetDevice のライフ・サイクル
- RoutePointDevice のライフ・サイクル
- Oracle Telephony Adapter サーバーの停止

5.4.1 Oracle Telephony Adapter サーバーの起動

Java 用 SDK

- OTAS プロセスが起動します。
- OTAS がアダプタ構成をロードします。
- OTAS が構成から TeleDeviceFactory クラス名を検索します。
- OTAS が TeleDeviceFactory クラスをロードして新規インスタンスを作成します。
- OTAS が TeleDeviceFactory.init() をコールし、すべての構成データをハッシュテーブルとして渡します。
- TeleDeviceFactory 自体が構成データを使用して適切に初期化されます。

C 用 SDK

- OTAS プロセスが起動します。
- OTAS がアダプタ構成をロードします。
- OTAS がアダプタ DLL をロードします。
- OTAS が factInit() 関数をコールします。
- 構成データは、C 関数 occtProviderConfigGet を介して使用できます。
- C のアダプタ自体が構成データを使用して適切に初期化されます。

5.4.2 TelesetDevice のライフ・サイクル

5.4.2.1 作成

Java 用 SDK

- Oracle Telephony Adapter サーバー（OTAS）が TeleDeviceFactory.createTelesetDevice() メソッドをコールして、新規 TelesetDevice を作成します。
- TeleDeviceFactory が TelesetDevice の新規インスタンスを戻します。
- OTAS が TelesetDevice.addTelesetEventListener() メソッドをコールし、新規に作成した TelesetDevice に TelesetEventListener を登録します。
- TelesetDevice が OTAS からの API コールの受信を開始し、TelesetEventListener でイベント・メソッドのコールを開始します。

C 用 SDK

- OTAS が OcctTelesetDevice の新規内部構造を作成します。
- OTAS が OcctTelesetDevice ポインタを使用して tsInit() 関数をコールします。
- アダプタが tsInit() 関数内での初期化を実行します。
- アダプタが API ファンクション・コール（tsXXX()）の受信を開始し、イベント関数（tsXXXEvent()）のコールを開始します。

5.4.2.2 破棄

Java 用 SDK

- OTAS が TeleDeviceFactory.destroyTeleDevice() をコールします。
- OTAS が TelesetDevice.removeTelesetEventListener() メソッドをコールして、TelesetDevice から TelesetEventListener を登録解除します。

- TeleDeviceFactory がクリーン・アップを実行し、TelesetDevice に割り当てられたリソースを解放します。
- TeleDevice が OTAS からの API コールの受信を停止し、TelesetEventListener でのイベント・メソッドのコールを停止します。
- その後、TeleDevice オブジェクトは JVM によりガベージ・コレクションが行われます。

C 用 SDK

- OTAS が tsDestroy() 関数をコールします。
- アダプタがクリーン・アップを実行し、tsDestroy() で TelesetDevice に割り当てられたリソースを解放します。
- アダプタが API ファンクション・コール（tsXXX()）の受信を停止し、イベント関数（tsXXXEvent()）のコールを停止します。
- OcctTelesetDevice 構造体が解放され、OcctTelesetDevice へのポインタは使用できなくなります。

5.4.3 RoutePointDevice のライフ・サイクル

5.4.3.1 作成

Java 用 SDK

- OTAS が TeleDeviceFactory.createRoutePointDevice() メソッドをコールして、新規 RoutePointDevice を作成します。
- TeleDeviceFactory が RoutePointDevice の新規インスタンスを戻します。
- OTAS が RoutePointDevice.addRoutePointEventListener() メソッドをコールして、新規に作成した RoutePointDevice に RoutePointEventListener を登録します。
- RoutePointDevice が OTAS からの API コールの受信を開始し、RoutePointEventListener でイベント・メソッドのコールを開始します。

C 用 SDK

- OTAS が OcctRoutePointDevice の新規内部構造を作成します。
- OTAS が RoutePointDevice ポインタを使用して rpInit() 関数をコールします。
- アダプタが rpInit() 関数内での初期化を実行します。
- アダプタが API ファンクション・コール（rpXXX()）の受信を開始し、イベント関数（rpXXXEvent()）のコールを開始します。

5.4.3.2 破棄

Java 用 SDK

- OTAS が TeleDeviceFactory.destroyTeleDevice() をコールします。
- OTAS が RoutePointDevice.removeRoutePointEventListener() メソッドをコールして、RoutePointDevice から RoutePointEventListener を登録解除します。
- TeleDeviceFactory がクリーン・アップを実行し、RoutePointDevice に割り当てられたリソースを解放します。
- RoutePointDevice が OTAS からの API コマンドの受信を停止し、RoutePointEventListener でのイベント・メソッドのコールを停止します。
- その後、RoutePointDevice オブジェクトは JVM によりガベージ・コレクションが行われます。

C 用 SDK

- OTAS が rpDestroy() 関数をコールします。
- アダプタがクリーン・アップを実行し、rpDestory() で TelesetDevice に割り当てられたリソースを解放します。
- アダプタが API ファンクション・コール（rpXXX()）の受信を停止し、イベント関数（rpXXXEvent()）のコールを停止します。
- OcctRoutePointDevice 構造体が解放され、OcctRoutePointDevice へのポインタは使用できなくなります。

5.4.4 Oracle Telephony Adapter サーバーの停止

- OTAS が、作成される TeleDevice ごとに 1 つずつ TeleDeviceFactory.destroyTeleDevice() をコールし、既存の TeleDevice オブジェクトをクリーン・アップします。
- OTAS プロセスが停止します。

5.5 TeleDeviceFactory の実装

この項では、TeleDeviceFactory の実装に使用可能なメソッドについて説明します。TeleDeviceFactory は、サード・パーティのコンサルタントまたはスイッチや CTI ミドルウェアのプロバイダにより実装されるインタフェースです。この実装により、TeleDevice オブジェクトと RoutePointDevice オブジェクトの作成と破棄へのアクセスが制御されます。TeleDevice は、この 2 つのオブジェクトのベース・クラスです。

5.5.1 メソッド

TeleDeviceFactory の実装には、次のメソッドを使用します。

5.5.1.1 init

定義

- 起動とスイッチおよび CTI ミドルウェアへの接続に必要なプロバイダ固有の情報を使用して、プロバイダを初期化します。
- 前提条件 : インスタンス化された TeleDevice オブジェクトがないこと。
- 成果 : TeleDevice オブジェクトをインスタンス化できるようになります。

パラメータ

`data`: プロバイダ・データによるキー値のペア。

Java 用 SDK

```
public abstract void init(Hashtable data) throws TeleDeviceException.
```

C 用 SDK

```
OEXPORT OcctCodes factInit();
IMPORT_EXPORT char** occtProviderConfigGetKeys(int *len);
IMPORT_EXPORT char* occtProviderConfigGet(char* key);
```

5.5.1.2 createTelesetDevice

定義

TelesetDevice コマンドを受け入れて Oracle Interaction Center に TelesetEventListener イベントを送信できるオブジェクトを作成します。このメソッドはテレセットごとに一度コールされ、TelesetDevice オブジェクトまたは例外を戻します。

- 前提条件 : TelesetDevice がインスタンス化されていないこと。
- 成果 : TelesetDevice がインスタンス化されます。

パラメータ

- `extension1`: テレセットの回線 1 の DN。
- `extension2`: テレセットの回線 2 の DN。extension1 と同じ DN を指定できます。
- `extension3`: テレセットの回線 3 の DN。extension1 または extension2 と同じ DN を指定できます。また、extension1 および extension2 のスピルオーバー回線を示す * も指定できます。

Java 用 SDK

```
public abstract TelesetDevice createTelesetDevice(String extension1, String
extension2, String extension3) throws TeleDeviceException.
```

C 用 SDK

```
OEXPORT OcctCodes tsInit(OcctTelesetDevice* ptr, const char* extension1, const char*
extension2, const char* extension3);
```

5.5.1.3 createRoutePointDevice

定義

RoutePointDevice コマンドを受け入れて、Oracle Interaction Center に RoutePointEventListener イベントを送信するオブジェクトを作成します。このメソッドはルート・ポイントごとに一度コールされ、RoutePointDevice オブジェクトまたは例外を戻します。

- 前提条件 : RoutePointDevice がインスタンス化されていないこと。
- 成果 : RoutePointDevice がインスタンス化されます。

パラメータ

- `routePointExtension`: ルート・ポイントの DN。
- `getRouteRequest`:
  - プロバイダが送信要求を Oracle Telephony Manager に転送する必要がある場合は True。
  - False は、ルートが受動的にモニターされていることを示します。
- `data`: ルート・ポイントに関するプロバイダ固有の init データ。

Java 用 SDK

```
public abstract RoutePointDevice createRoutePointDevice(String routePointExtension,
boolean getRouteRequest, Hashtable data) throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes rpInit(OcctRoutePointDevice* ptr, const char* routePointExtension,
oboolean getRouteRequest, OHashtable* data);
```

5.5.1.4 destroyTeleDevice

定義

Oracle Interaction Center が特定の TeleDevice で終了することを示す通知。プロバイダは、必要に応じてリソースを再生できます。このメソッドの戻り時に、破棄された TeleDevice が再作成されるまで、プロバイダはその TeleDevice に関するイベントを送信できない場合があります。

- 前提条件 : CreateTeleDevice。
- 成果 : teleDevice に割り当てられていたリソースが解放されます。

パラメータ

`teleDevice`: 破棄する TeleDevice。

Java 用 SDK

```
public abstract void destroyTeleDevice(TeleDevice teleDevice) throws
TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes rpDestroy(OcctRoutePointDevice* ptr);
OEXPORT OcctCodes tsDestroy(OcctTelesetDevice* ptr);
```

5.6 TelesetDevice の実装

- TelesetDevice インタフェース
- TelesetEventListener インタフェース

5.6.1 TelesetDevice インタフェース

Oracle Interaction Center は TelesetDevice インタフェースを使用して、スイッチまたは CTI ミドルウェアのテレフォニ・リソースにアクセスします。このインタフェースの実装は、サード・パーティのコンサルタントまたはスイッチや CTI ミドルウェアのベンダーにより提供されます。各 TelesetDevice には、3 つの回線があるものと見なされます。各回線は、一度に 1 コールのみが想定されています。回線には、0、1、2 など、0 ベースの索引が付いています。

5.6.1.1 メソッド

TelesetDevice インタフェースに定義されているほとんどのメソッドでは、回線インデックスを使用して操作の対象となる回線を指定します。

TelesetDevice の実装には、次のメソッドを使用します。

answerCall

説明

特定の呼出しコールに応答します。

パラメータ

`lineIndex`: コールが呼び出している回線（0 ベース）。

前提条件

アクティブな回線を指定する必要があります。指定した回線で呼出中のアクティブなコールが存在する必要があります。

成功時の結果

callEstablishedEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_ANSWER_CALL
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、NO_CURRENT_CALL、LINE_INDEX_OUT_OF_RANGE、LINE_ALREADY_IN_USE

スロー

TeleDeviceException -

Java 用 SDK

```
public void answerCall(int lineIndex)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsAnswerCall(OcctTelesetDevice*, oint);
```

makeCall

説明

特定のテレセットからコールを実行します。

前提条件

コールが実行されていないアクティブな回線を指定する必要があります。

成功時の結果

callDialingEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_RELEASE_CALL
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、DESTINATION_INVALID、DESTINATION_BUSY、LINE_INDEX_OUT_OF_RANGE、LINE_ALREADY_IN_USE

パラメータ

- `mediaItemId`: このコマンドによって発生する新しいコールにより、特定のメディア項目 ID が戻される必要があります。
- `lineIndex`: コール元の位置（0 ベース）。
- `destinationNumber`: ダイヤルする番号。

スロー

TeleDeviceException -

Java 用 SDK

```
public void makeCall(java.lang.String mediaItemId,
int lineIndex,
        java.lang.String destinationNumber)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsMakeCall(OcctTelesetDevice*, const char*, oint, const char*);
```

releaseCall

説明

特定の確立済みコールを解放（切断）します。

前提条件

指定した回線にアクティブなコールが存在する必要があります。

成功時の結果

callReleasedEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_RELEASE_CALL
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

`lineIndex`: コール元の位置（0 ベース）。

スロー

TeleDeviceException -

Java 用 SDK

```
public void releaseCall(int lineIndex)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsReleaseCall(OcctTelesetDevice*, oint);
```

holdCall

説明

特定の確立済みコールを保留にします。

前提条件

指定した回線にアクティブなコールが存在する必要があります。

成功時の結果

callHeldEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_HOLD_CALL
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

`lineIndex`: コール元の位置（0 ベース）。

スロー

TeleDeviceException -

Java 用 SDK

```
public void holdCall(int lineIndex)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsHoldCall(OcctTelesetDevice*, oint);
```

retrieveCall

説明

特定の保留コールを再接続します。

前提条件

指定した回線で保留中のアクティブなコールが存在する必要があります。

成功時の結果

callRetrievedEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_RETRIEVE_CALL
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

`lineIndex`: コール元の位置（0 ベース）。

スロー

TeleDeviceException -

Java 用 SDK

```
public void retrieveCall(int lineIndex)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsRetrieveCall(OcctTelesetDevice*, oint);
```

consultationCall

説明

コンサルタント・コールを実行します。

前提条件

指定した回線にアクティブなコールが存在する必要があります。

成功時の結果

callDialingEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_CONSULTATION_CALL

- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、DESTINATION_INVALID、DESTINATION_BUSY、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

- `mediaItemId`: このコマンドによって発生する新しいコールにより、特定のメディア項目 ID が戻される必要があります。
- `lineIndex`: ベースとなるコール元の位置（0 ベース）。
- `destinationNumber`: ダイヤルする番号。
- `consultationType`: oracle.apps.cct.sdk.constants.ConsultationCallTypes を参照してください。

スロー

TeleDeviceException -

参照

ConsultCallTypes

Java 用 SDK

```
public void consultationCall(java.lang.String mediaItemId,
            int lineIndex,
            java.lang.String destinationNumber,
            ConsultCallTypes consultationType)
            throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsConsultationCall(OcctTelesetDevice*, const char*, oint, const
char*, oint);
```

completeTransfer

説明

- コール転送を完了します。

前提条件

- 事前に consultationCall がコールされていること。

成功時の結果

保留回線の場合は callTransferredEvent、アクティブな回線の場合は callReleasedEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_COMPLETE_TRANSFER
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

- `activeLineIndex`: アクティブ・コールの回線インデックス（0 ベース）。
- `heldLineIndex`: 保留コールの回線インデックス（0 ベース）。

スロー

TeleDeviceException -

Java 用 SDK

```
public void completeTransfer(int activeLineIndex,
            int heldLineIndex)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsCompleteTransfer(OcctTelesetDevice*, oint, oint);
```

completeConference

説明

コール会議を完了します。

前提条件

事前に consultationCall がコールされていること。

成功時の結果

callConferencedEvent、callReleasedEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_COMPLETE_CONFERENCE
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

- `activeLineIndex`: アクティブ・コールの回線インデックス（0 ベース）。
- `heldLineIndex`: 保留コールの回線インデックス（0 ベース）。

スロー

TeleDeviceException -

Java 用 SDK

```
public void completeConference(int activeLineIndex,
            int heldLineIndex)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsCompleteTransfer(OcctTelesetDevice*, oint, oint);
```

blindTransfer

説明

コールを宛先番号にブラインド転送します。

前提条件

指定した回線にアクティブなコールが存在する必要があります。

成功時の結果

callTransferredEvent、callReleasedEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_BLIND_TRANSFER
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、DESTINATION_INVALID、DESTINATION_BUSY、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

- `mediaItemId`: このコマンドによって発生する新しいコールにより、特定のメディア項目 ID が戻される必要があります。
- `lineIndex`: ベースとなるコールの回線インデックス（0 ベース）。
- `destinationNumber`: ダイヤルする番号。

スロー

TeleDeviceException -

Java 用 SDK

```
public void blindTransfer(java.lang.String mediaItemId,
            int lineIndex,
            java.lang.String destinationNumber)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsBlindTransfer(OcctTelesetDevice*, const char*, oint, const
char*);
```

swapWithHeld

説明

アクティブ・コールを保留にし、保留コールを再接続します。

前提条件

指定した回線にアクティブなコールが存在する必要があります。

成功時の結果

callHeldEvent および対応するコールの callRetrievedEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_SWAP_WITH_HELD
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

- `activeLineIndex`: アクティブ・コールの位置（0 ベース）。
- `heldLineIndex`: 保留コールの位置（0 ベース）。

スロー

TeleDeviceException -

Java 用 SDK

```
public void swapWithHeld(int activeLineIndex,
            int heldLineIndex)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsSwapWithHeld(OcctTelesetDevice*, oint, oint);
```

sendDtmf

説明

コールに dtmf ダイヤル・トーンを送信します。

前提条件

指定した回線にアクティブなコールが存在する必要があります。

成功時の結果

ミドルウェアに対する dtmf の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_SEND_DTMF
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、LINE_INDEX_OUT_OF_RANGE、NO_CURRENT_CALL

パラメータ

- `lineIndex`: コールの回線インデックス（0 ベース）。
- `dtmfDigits`: 送信される dtmf ダイヤル・トーン。

スロー

TeleDeviceException -

Java 用 SDK

```
public void sendDtmf(int lineIndex,
            java.lang.String dtmfDigits)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsSendDtmf(OcctTelesetDevice*, oint, const char*);
```

loginAgent

説明

エージェントをテレセットにログインします。

前提条件

エージェントがログインされていないこと。

成功時の結果

agentLogintEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_LOGIN_AGENT
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、AGENT_INVALID、AGENT_PASSWORD_INVALID、TELESET_IN_USE

パラメータ

- `acdAgentId`: スイッチ固有のエージェント ID。
- `acdAgentPassword`: スイッチ固有のエージェント・パスワード。
- `acdQueue`: スイッチ固有のエージェント ACD キュー。

スロー

TeleDeviceException -

Java 用 SDK

```
public void loginAgent(java.lang.String acdAgentId,
            java.lang.String acdAgentPassword,
            java.lang.String acdQueue)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsLoginAgent(OcctTelesetDevice*, const char*, const char*, const char*);
```

logoutAgent

説明

エージェントをテレセットからログアウトします。

前提条件

エージェントがログインされていること。

成功時の結果

agentLogoutEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_LOGOUT_AGENT
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED、AGENT_PASSWORD_INVALID

パラメータ

- `acdAgentId`: スイッチ固有のエージェント ID。
- `acdAgentPassword`: スイッチ固有のエージェント・パスワード。
- `acdQueue`: スイッチ固有のエージェント ACD キュー。

スロー

TeleDeviceException -

Java 用 SDK

```
public void logoutAgent(java.lang.String acdAgentId,
            java.lang.String acdAgentPassword,
            java.lang.String acdQueue)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsLogoutAgent(OcctTelesetDevice*, const char*, const char*, const char*);
```

addTelesetEventListener

説明

TelesetEventListener をリスナー・リストに追加します。

パラメータ

`eventListener`: この TelesetDevice のイベントに関係する TelesetEventListener。

Java 用 SDK

```
public void addTelesetEventListener(TelesetEventListener eventListener)
```

C 用 SDK

該当なし

removeTelesetEventListener

説明

指定したリスナーをリスナー・リストから削除します。

パラメータ

`eventListener`: 削除する TelesetEventListener。

Java 用 SDK

```
public void removeTelesetEventListener(TelesetEventListener eventListener)
```

C 用 SDK

該当なし

agentNotReady メソッド

定義

- エージェントの状態を「電話を自分にルーティングしない」に設定します。
- 前提条件 : agentLoginEvent。
- 成果 : agentNotReadyEvent。

パラメータ

使用不可

Java 用 SDK

```
public abstract void agentNotReady() throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsAgentNotReady(OcctTelesetDevice* ptr);
```

agentReady メソッド

定義

エージェントの状態を「電話を自分にルーティングする」に設定します。

前提条件

エージェントがログインされていること。

成功時の結果

agentReadyOnEvent の送信

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_TELESET_AGENT_READY
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED

スロー

TeleDeviceException -

Java 用 SDK

```
public void agentReady()
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes tsAgentReady(OcctTelesetDevice*);
```

5.6.2 TelesetEventListener インタフェース

TelesetEventListener インタフェースは、サード・パーティのコンサルタントまたはスイッチや CTI ミドルウェアのプロバイダが、Oracle Interaction Center にテレセット・イベントを通知するために使用します。このインタフェースを実装するオブジェクトは、イベント・レポートのために Oracle Telephony Manager から TelesetDevice オブジェクトに提供されます。

5.6.2.1 メソッド

TelesetEventListener の実装には、次のメソッドを使用します。

beginCallEvent

定義

このイベントは、特定の TelesetDevice の新しいコールに関する最初のイベントです。このイベントには、テレセットの状態は関連しません。

パラメータ

- `mediaItemId`: コールに関連したメディア項目。
- `mediaItemIdComplete`: アダプタがこのイベントに有効な mediaItemId を持つ場合は true、アダプタが updateMediaItemId をコールする可能性がある場合は false。
- `lineIndex`: 新しいコールの位置。
- `ani`:
- `dnis`:
- `data`: コールに関連した IVR および他のデータ。
- `dataComplete`: アダプタにこのコールに関連した全データがある場合は true、アダプタが updateCallData をコールする可能性がある場合は false。

Java 用 SDK

```
public void beginCallEvent(java.lang.String mediaItemId,
boolean mediaItemIdComplete,
            int lineIndex,
```

```
            java.lang.String ani,
            java.lang.String dnis,
            java.util.Hashtable data,
            boolean dataComplete)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsBeginCallEvent(OcctTelesetDevice*, const char*, oboolean,
oint, const char*, const char*, OHashtable*, oboolean);
```

callDialingEvent

定義

- コールがダイヤル中です。
- 予期される以前のイベント : beginCallEvent
- 予期される次のイベント : callEstablishedEvent または callReleasedEvent

パラメータ

`lineIndex`: コールの位置。

Java 用 SDK

```
public void callDialingEvent(int lineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallDialingEvent(OcctTelesetDevice*, oint);
```

callRingingEvent

定義

- コールが呼出中です。
- 予期される以前のイベント : beginCallEvent
- 予期される次のイベント : callEstablishedEvent または callReleasedEvent

パラメータ

`lineIndex`: コールの位置。

Java 用 SDK

```
public void callRingingEvent(int lineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallRingingEvent(OcctTelesetDevice*, oint);
```

callEstablishedEvent

定義

- コールが確立済みです。関係するパーティが接続されています。
- 予期される以前のイベント : beginCallEvent、callDialingEvent、callRingingEvent。
- 予期される次のイベント : callReleasedEvent、callHeldEvent

パラメータ

`lineIndex`: コールの位置。

Java 用 SDK

```
public void callEstablishedEvent(int lineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallEstablishedEvent(OcctTelesetDevice*, oint);
```

callHeldEvent

定義

- コールが TelesetDevice により保留になっています。コールを保留するために確立する必要はありません。
- 予期される以前のイベント : callDialingEvent または callEstablishedEvent。
- 予期される次のイベント : callRetrievedEvent。

パラメータ

`lineIndex`: コールの位置。

Java 用 SDK

```
public void callHeldEvent(int lineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallHeldEvent(OcctTelesetDevice*, oint);
```

callRetrievedEvent

定義

- コールが TelesetDevice により再接続されています。
- 予期される以前のイベント : callHeldEvent
- 予期される次のイベント : callReleasedEvent

パラメータ

`lineIndex`: コールの位置。

Java 用 SDK

```
public void callRetrievedEvent(int lineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallRetrievedEvent(OcctTelesetDevice*, oint);
```

callReleasedEvent

定義

- コールは解放または切断されています。これは、特定回線でのコールに対する最終イベントです。
- 予期される以前のイベント : callDialingEvent、callRingingEvent または callEstablishedEvent
- 予期される次のイベント : なし

パラメータ

`lineIndex`: コールの位置。

Java 用 SDK

```
public void callReleasedEvent(int lineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallReleasedEvent(OcctTelesetDevice*, oint);
```

callTransferredEvent

定義

- コールは転送済みです。このイベントは転送元にのみ送信され、転送元はコールから外れています。このイベントは、転送元が callReleasedEvent を受信する前に発生する必要があります。
- 予期される以前のイベント : callEstablishedEvent
- 予期される次のイベント : callReleasedEvent

Java 用 SDK

```
public void callTransferredEvent()
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallTransferredEvent(OcctTelesetDevice*);
```

callConferencedEvent

定義

- コールは会議済みです。このイベントは、会議の制御側にのみ送信されます。回線インデックスを指定すると、その回線でのコールがアクティブになる（保留になった場合に暗黙的に再接続する）ことを意味します。2 つのコールがアクティブになる場合は、回線ごとに 1 つずつ、2 つの callConferencedEvent を送信する必要があります（スイッチによりベースとなるコールとコンサルタント・コールが 1 つのコールに組み合されることはありません）。このイベントは、callReleasedEvent が送信される前に会議の制御側に送信する必要があります。
- 予期される以前のイベント : callEstablishedEvent
- 予期される次のイベント : callReleasedEvent

パラメータ

`activeLineIndex`: 同時に再接続するコールの回線インデックス。

Java 用 SDK

```
public void callConferencedEvent(int activeLineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallConferencedEvent(OcctTelesetDevice*, oint);
```

callConferencedEvent

定義

- コールは会議済みで、アクティブな回線はありません。このイベントは、会議の制御側が、両方のコールが保留の状態で、結果的にコールが保留のまま完了できるスイッチに対するものです。このイベントは、releasedEvent が送信される前に会議の制御側に送信する必要があります。
- 予期される以前のイベント : callEstablishedEvent
- 予期される次のイベント : callRetrivedEvent または callReleasedEvent

Java 用 SDK

```
public void callConferencedEvent()
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsCallConferencedEvent2(OcctTelesetDevice*);
```

agentLoginEvent

定義

エージェントがログインしました。

Java 用 SDK

```
public void agentLoginEvent()
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsAgentLoginEvent(OcctTelesetDevice*);
```

agentLogoutEvent

定義

エージェントがログアウトしました。

Java 用 SDK

```
public void agentLogoutEvent()
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsAgentLogoutEvent(OcctTelesetDevice*);
```

agentReadyEvent

定義

エージェントが「通話可」状態になっています。

Java 用 SDK

```
public void agentReadyEvent()
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsAgentReadyEvent(OcctTelesetDevice*);
```

agentNotReadyEvent

定義

エージェントが「通話不可」状態になっています。

Java 用 SDK

```
public void agentNotReadyEvent()
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsAgentNotReadyEvent(OcctTelesetDevice*);
```

updateMediaItemIdEvent

定義

- コールのメディア項目 ID を更新します。これは、指定した回線でのコールがメディア項目 ID に関連付けられていることを示します。このイベントは、指定したコールの beginCallEvent が mediaItemIdComplete = false の状態で送信される場合のみ可能です。また、このイベントは、beginCallEvent が送信されてから 7 秒以内に送信する必要があります。
- 予期される以前のイベント : beginCallEvent

パラメータ

- `mediaItemId`: 新規 ID。
- `lineIndex`: 対象となるコール。

Java 用 SDK

```
public void updateMediaItemIdEvent(int lineIndex,
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsUpdateMediaItemIdEvent(OcctTelesetDevice*, oint, const
char*);
```

updateCallDataEvent

定義

- コールのコール・データを更新します。これは、指定した回線でのコールが、指定したコール・データに関連付けられていることを示します。このイベントは、指定したコールの beginCallEvent が dataComplete = false の状態で送信される場合のみ可能です。このイベントは、beginCallEvent の送信後に随時送信できます。
- 予期される以前のイベント : beginCallEvent

パラメータ

- `data`: 新規データ。
- `lineIndex`: 対象となるコール。

Java 用 SDK

```
public void updateCallDataEvent(int lineIndex,
            java.util.Hashtable data)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsUpdateCallDataEvent(OcctTelesetDevice*, oint,
OHashtable*);
```

updateLineIndexEvent

定義

- コールの回線インデックスが変更されました。このイベントは、beginCallEvent の後に随時送信できます。
- 予期される以前のイベント : beginCallEvent

パラメータ

- `oldLineIndex`: (0+)
- `newLineIndex`: (0+)

Java 用 SDK

```
public void updateLineIndexEvent(int oldLineIndex,
            int newLineIndex)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsUpdateLineIndexEvent(OcctTelesetDevice*, oint, oint);
```

updateSoftphoneDisplayEvent

定義

- ソフトフォンの表示を更新します。このイベントは、beginCallEvent の後に随時送信できます。
- 予期される以前のイベント : beginCallEvent

パラメータ

`softphoneDisplayData`: 情報画面に表示されるデータ。

Java 用 SDK

```
public void updateSoftphoneDisplayEvent(java.util.Hashtable softphoneDisplayData)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsUpdateSoftphoneDisplayEvent(OcctTelesetDevice*,
OHashtable*);
```

updateOtherPartyNumberEvent

定義

- 相手パーティ番号を更新します。このイベントは、beginCallEvent の後に随時送信できます。
- 予期される以前のイベント : beginCallEvent

パラメータ

`lineIndex`: 相手パーティ番号を変更するコール。

`otherPartyNumber`: 新規の相手パーティ番号。

Java 用 SDK

```
public void updateOtherPartyNumberEvent(int lineIndex,
            java.lang.String otherPartyNumber)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsUpdateOtherPartyNumberEvent(OcctTelesetDevice*, oint,
const char*);
```

errorEvent

定義

一般エラー。

パラメータ

- `lineIndex`: コールの位置。エラーに関連した回線がない場合は -1。
- `errorType`: エラーのコマンドまたはカテゴリ。
- `errorCode`: 特定のエラー。
- `errorMessage`: 「その他のエラー」。

参照

ErrorTypes、ErrorCodes

Java 用 SDK

```
public void errorEvent(int lineIndex,
            ErrorTypes errorType,
            ErrorCodes errorCode,
            java.lang.String errorMessage)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL tsErrorEvent(OcctTelesetDevice*, oint, oint, oint, const
char*);
```

5.7 RoutePointDevice の実装

RoutePointDevice には、次のインタフェースが適用されます。

- RoutePointDevice インタフェース
- RoutePointEventListener インタフェース

5.7.1 RoutePointDevice インタフェース

Oracle Telephony Manager は RoutePointDevice インタフェースを使用して、スイッチまたは CTI ミドルウェアのルート・ポイント・リソースにアクセスします。このインタフェースの実装は、サード・パーティのコンサルタントまたはスイッチや CTI ミドルウェアのベンダーにより提供されます。

assignMediaItemId

説明

コール ID にメディア項目 ID を割り当てます。アダプタはコールのルーティング時に、このメディア項目 ID を（TelesetEventListener インタフェースを介して）宛先に渡す必要があります。または、コールがルーティング済みの場合、アダプタは updateMediaItemIdEvent を（TelesetEventListener インタフェースを介して）宛先に送信する必要があります。このメソッドは、パッシブ・モードでのみコールされます。

前提条件

このコールの前に、指定したコール ID の callQueuedEvent を受信する必要があります。

成功時の結果

メディア項目 ID をコール ID に関連付けます。マッピングはアダプタにより内部的に保持されます。

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_ROUTE_POINT_ASSIGN_MEDIA_ITEM_ID
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED

パラメータ

- `callId`: mediaItemId に関連付ける callId。
- `mediaItemId`: このコマンドによって発生する新しいコールにより、特定のメディア項目 ID が戻される必要があります。

スロー

TeleDeviceException -

Java 用 SDK

```
public void assignMediaItemId(java.lang.String callId,java.lang.String mediaItemId)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes rpAssignMediaItemId(OcctRoutePointDevice*, const char*, const
char*);
```

routeCall

定義

コールを指定の宛先にルーティングします。

前提条件

このコールの前に、指定したコール ID の routeRequestEvent を受信する必要があります。

成功時の結果

コールが宛先にルーティングされ、callDivertedEvent が送信されます。

失敗時の結果

- errorEvent の送信または TeleDeviceException のスロー
- ErrorType: ErrorTypes.ERROR_TYPE_ROUTE_POINT_ROUTE_CALL
- 考えられるエラー・コード : ErrorCodes.FAILED、FUNCTION_NOT_SUPPORTED

パラメータ

- `callId`: 宛先を変更する送信要求。
- `destinationNumber`: コールのルーティング先の番号。
- `mediaItemId`: このコマンドによって発生する新しいコールにより、特定のメディア項目 ID が戻される必要があります。

スロー

TeleDeviceException -

Java 用 SDK

```
public void routeCall(java.lang.String callId,
            java.lang.String destinationNumber,
            java.lang.String mediaItemId)
        throws TeleDeviceException
```

C 用 SDK

```
OEXPORT OcctCodes rpRouteCall(OcctRoutePointDevice*, const char*, const char*, const char*);
```

addRoutePointEventListener

定義

RoutePointEventListener をリスナー・リストに追加します。

パラメータ

`eventListener`: この RoutePointDevice のイベントに関係する RoutePointEventListener。

Java 用 SDK

```
public void addRoutePointEventListener(RoutePointEventListener eventListener)
```

C 用 SDK

該当なし

removeRoutePointEventListener

定義

指定したリスナーをリスナー・リストから削除します。

パラメータ

`eventListener`: 削除する RoutePointEventListener。

Java 用 SDK

```
public void removeRoutePointEventListener(RoutePointEventListener eventListener)
```

C 用 SDK

該当なし

5.7.2 RoutePointEventListener インタフェース

RoutePointEventListener インタフェースは、サード・パーティのコンサルタントまたはスイッチや CTI ミドルウェアのプロバイダが、Oracle Interaction Center にルート・ポイント・イベントを通知するために使用します。このインタフェースを実装するオブジェクトは、イベント・レポートのために Oracle Telephony Manager から RoutePointDevice オブジェクトに提供されます。

beginCallEvent

定義

- このイベントは、特定の RoutePointDevice の新しいコールに関する最初のイベントにする必要があります。
- 予期される次のイベント : callQueuedEvent、routeRequestedEvent（アクティブ・モード）

パラメータ

- `mediaItemId`: コールに関連したメディア項目。
- `mediaItemIdComplete`: このコールですべてのメディア項目 ID を使用可能な場合は **true**、アダプタが updateMediaItemId をコールする可能性がある場合は **false**。
- `callId`: アダプタ固有のコール ID。
- `ani`:
- `dnis`:
- `data`: コールに関連した IVR および他のデータ。
- `dataComplete`: このコールですべてのコール・データを使用可能な場合は **true**、アダプタが updateCallData をコールする可能性がある場合は **false**。

Java 用 SDK

```
public void beginCallEvent(java.lang.String mediaItemId,
            boolean mediaItemIdComplete,
            java.lang.String callId,
            java.lang.String ani,
            java.lang.String dnis,
            java.util.Hashtable data,
            boolean dataComplete)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpBeginCallEvent(OcctRoutePointDevice*, const char*,
oboolean, const char*, const char*, const char*, OHashtable*, oboolean);
```

callQueuedEvent

定義

- コールはルート・ポイントでキューに入れられました。
- 予期される以前のイベント : beginCallEvent
- 予期される次のイベント : routeRequestedEvent（アクティブ・モード）または callDivertedEvent

パラメータ

`callId`: アダプタ固有のコール ID。

Java 用 SDK

```
public void callQueuedEvent(java.lang.String callId)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpCallQueuedEvent(OcctRoutePointDevice*, const char*);
```

callAbandonedEvent

定義

- コールは、エージェントへの接続前に解放または切断されました。
- 予期される以前のイベント : callQueuedEvent

パラメータ

`callId`: アダプタ固有のコール ID。

Java 用 SDK

```
public void callAbandonedEvent(java.lang.String callId)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpCallAbandonedEvent(OcctRoutePointDevice*, const char*);
```

callDivertedEvent

定義

- コールの宛先が変更されました（コールに応答しないブラインド転送）。
- 予期される以前のイベント : callQueuedEvent

パラメータ

`callId`: アダプタ固有のコール ID。

Java 用 SDK

```
public void callDivertedEvent(java.lang.String callId)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpCallDivertedEvent(OcctRoutePointDevice*, const char*);
```

routeRequestedEvent

定義

- ルート・ポイントが、ルーティングのためにコールを発行しました。
- 予期される以前のイベント : beginCallEvent

パラメータ

`callId`: アダプタ固有のコール ID。

Java 用 SDK

```
public void routeRequestedEvent(java.lang.String callId)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpRouteRequestedEvent(OcctRoutePointDevice*, const char*);
```

updateMediaItemIdEvent

定義

- コールのメディア項目 ID を更新します。これは、指定したコール ID がメディア項目 ID に関連付けられていることを示します。このイベントは、対応するコールの beginCallEvent が mediaItemIdComplete = false の状態で送信される場合のみ可能です。また、このイベントは、beginCallEvent が送信されてから **5 秒**以内に送信する必要があります。
- 予期される以前のイベント : beginCallEvent（mediaItemIdComplete = false）

パラメータ

- `mediaItemId`: 新規 ID。
- `callId`: 特定のコール。

Java 用 SDK

```
public void updateMediaItemIdEvent(java.lang.String callId,java.lang.String
mediaItemId)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpUpdateMediaItemIdEvent(OcctRoutePointDevice*, const
char*, const char*);
```

updateCallDataEvent

定義

- コールのコール・データを更新します。これは、指定したコール ID が、指定したコール・データに関連付けられていることを示します。このイベントは、beginCallEvent が dataComplete = false の状態で送信される場合のみ可能です。また、このイベントは、beginCallEvent の後に随時送信できます。
- 予期される以前のイベント : beginCallEvent（dataComplete = false）

パラメータ

- `data`: 新規データ。
- `callId`: 対象となるコール。

Java 用 SDK

```
public void updateCallDataEvent(java.lang.String callId,
            java.util.Hashtable data)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpUpdateCallDataEvent(OcctRoutePointDevice*, const char*,
OHashtable*);
```

errorEvent

定義

一般エラー。

パラメータ

- `callId`: 対象となるコール。エラーに関連したコールがない場合は NULL。
- `errorType`: エラーのコマンドまたはカテゴリ。
- `errorCode`: 特定のエラー。
- `errorMessage`: 「その他のエラー」。

Java 用 SDK

```
public void errorEvent(java.lang.String callId,
            ErrorTypes errorType,
            ErrorCodes errorCode,
            java.lang.String errorMessage)
```

C 用 SDK

```
IMPORT_EXPORT void OCALL rpErrorEvent(OcctRoutePointDevice*, const char*, oint,
oint, const char*);
```

参照

ErrorTypes、ErrorCodes

5.8 OTAS メッセージ・サービス

OTAS メッセージ・サービスは、内線で識別される TeleDevice に Point-to-Point メッセージを送信する単純なユーティリティです。開発者は、受信側 TeleDevice がローカルかリモートかに関係なくメッセージを送信できます。

拡張性とフォルト・トレランスのために、同じサーバー・グループで複数の OTAS を実行できます。Interaction Center サーバーは、OTAS の負荷に基づくラウンドロビン・アルゴリズムを使用して、サーバー・グループ内で使用可能なすべての OTAS に接続します。

Telephony Adapter の開発時には、各 TeleDevice（TelesetDevice と RoutePointDevice の両方）を相互に独立した状態に保つ必要があります。

注意： TelesetDevice と RoutePointDevice の間で相互参照しないでください。TeleDevice の正常動作とイベント生成は、他の TeleDevice へのアクセスに依存させないようにする必要があります。どの実装でも、TeleDevice のグローバル表を作成しないでください。これは、特定の TeleDevice の局所性を想定できず、アクセスを必要とするターゲット TeleDevice が別個の OTAS サーバー・プロセスに存在する場合があるためです。

スイッチと CTI ミドルウェアの特定の組合せのコール使用例では、一部のイベントまたは状態情報を TeleDevice 間で通信することが必要な場合があります。この問題に対処するために OTAS メッセージ・サービスを使用します。

5.8.1 OTAS メッセージ・サービス API

メッセージは、Java では文字列、C では char* として定義されます。OTAS メッセージ・サービスには、メッセージを指定の宛先に配信する機能があります。アダプタ開発者は、文字列メッセージの書式と解析方法を決定できます。

5.8.1.1 メッセージの送信

Java と C でメッセージを送信するには、次のメソッドおよび関数を使用します。

Java

メッセージを送信するには、Messenger ユーティリティのクラス・メソッドを使用します。

```
public static boolean sendMessage(String destnExtnNumber, String message)
```

C

メッセージを送信するには、occtSendMessage 関数を使用します。

```
oint occtSendMessage(char* destnExtnNumber, char* message);
```

- `destExtnNumber` は、ターゲット TeleDevice の内線番号です。TelesetDevice の場合は、3 回線の任意の内線を使用できます。RoutePointDevice の場合は、内線番号です。
- `message` は、送信するメッセージです。

5.8.1.2 メッセージの受信

Java と C でメッセージを受信するには、次のメソッドおよび関数を使用します。

Java

メッセージを受信するには、TelesetDevice と RoutePointDevice について次のメソッドを実装します。

```
public void receiveMessage(String destnExtnNumber, String message);
```

C

メッセージを受信するには、次の関数を実装します。

```
void tsReceiveMessage(const char* destnExtnNumber, const char* message);
void rpReceiveMessage(const char* destnExtnNumber, const char* message);
```

5.8.1.3 OTAS メッセージ・サービスを使用するサンプル使用例

次のサンプル使用例では、OTAS メッセージ・サービスを使用して、TeleDevice イベントのフェイクを作成し、メディア項目 ID を伝播させる方法について説明します。

TeleDeviceEvent のフェイクを作成

特定の TeleDevice に必要な TeleDeviceEvent をその TeleDevice が受信する CTI ミドルウェア・イベントのみに基づいて生成するための論理を、アダプタに実装できない場合や、困難な場合があります。たとえば、コールが ACD によりルート・ポイントからエージェントのテレセットにルーティングされるときに、CT Connect が RoutePointDevice に NULL/Diverted イベントを送信しないと、アダプタの論理がターゲット・エージェントの TelesetDevice により受信される RECEIVE/INBOUND_CALL を利用して callDivertedEvent のフェイクを作成しないかぎり、アダプタは callDivertedEvent を生成できません。OTAS メッセージ・サービスを使用すると、TeleDevice 間で相互参照してイベントのフェイクを作成できます。

次の使用例では、TeleDevice イベントのフェイクを作成する処理について説明します。

1. 顧客 X がルート・ポイント RP をコールします。RP の RoutePointDevice が CT Connect から QUEUED/INBOUND_CALL イベントを取得し、RP の RoutePointDevice が beginCallEvent を送信します。
2. ACD がコールを RP からエージェント A にルーティングします。RP の RoutePointDevice は CT Connect から NULL/DIVERTED イベントを受信しませんが、A の TelesetDevice は RECEIVE/INBOUND_CALL イベント（DNIS=RP）を受信し、sendMessage(RP, "EVT:callDiverted;CALLID:12345") をコールして、OTAS メッセージ・サービス経由で RP の RoutePointDevice にメッセージを送信します。
3. RP の RoutePointDevice は、OTAS メッセージを受信するとインバウンド・コール・キュー件数が正常に判別されるように、callDivertedEvent のフェイクを作成します。

メディア項目 ID の伝播

Oracle Advanced Inbound では、スイッチや CTI ミドルウェアにより割り当てられたコール ID がコールの転送時や会議時に変化する場合でも、同じメディア項目 ID は、同じ初期コールに関連するすべての物理コールに関連付け、伝播させる必要があります。この要件を満たすために、コールにデータを関連付ける CTI ミドルウェアの機能を利用する方法があります。たとえば、アダプタでは、メディア項目 ID を CT Connect のアプリケーション・データ・フィールドまたは Cisco ICM のコール変数に格納できます。OTAS メッセージ・サービスには、別の方法が用意されています。つまり、アダプタはメディア項目 ID を OTAS メッセージ経由で宛先 TeleDevice に渡すことができます。

次の使用例では、メディア項目 ID の伝播について説明します。

1. エージェント A は、回線 1 で顧客 X とすでにコール中です。
2. エージェント A がエージェント B に対するコンサルタント・コールを開始します。
3. OTM が A の TelesetDevice.consultationCall(Media Item Id=555, line=1, destination=B, consult type=transfer) を起動します。
4. B が次のコールを着信する前に、A の TelesetDevice が sendMessage(B,"MIID:555;ANI:A") を起動し、B の TelesetDevice に ANI=A とメディア項目 ID 555 を関連付ける必要があることを通知します。
5. A の TelesetDevice が、consultCall(callid, destination) をスイッチまたはミドルウェア上で起動します。
6. B の TelesetDevice がメッセージを受信して情報を格納します。
7. B の TelesetDevice が CT Connect から RECEIVE/INBOUND_CALL イベント（ANI=A）を受信し、B の TelesetDevice が beginCallEvent(MIID=555) を生成します。

5.9 Java のその他 API

この項では、アダプタ実装者が使用できる Java 用 Oracle Telephony Adapter SDK のその他の Java クラスについて説明します。この項の内容は、次のとおりです。

- クラス Logger
- クラス ErrorCodes
- クラス ConsultCallTypes
- クラス TeleDeviceEventMulticaster

5.9.1 クラス Logger

クラス Logger は、Oracle Interaction Center の標準ロギング・インフラストラクチャを使用するログ・ユーティリティです。現行のログ・レベルは、Oracle Telephony Adapter サーバーによりブート時に自動的に初期化されます。

Logger API は静的メソッドで構成されており、次のように起動する必要があります。

```
if (Logger.logStatus (Logger.LOG_LEVEL_VERBOSE))
{
    // Do any processing required to assemble message.
    String msg = "Message.";
    Logger.log (Logger.LOG_LEVEL_VERBOSE,msg);
}
```

5.9.1.1 静的定数

5 つのロギング・レベルである FATAL、ERROR、WARNING、INFO および VERBOSE が定義されています。ログ・メッセージについては、次の表に示すガイドラインに従ってください。

ログ・レベル	説明
LOG_LEVEL_FATAL	リカバリ不能なエラーを示す致命的ログ・レベル
LOG_LEVEL_ERROR	重大なエラーを示すエラー・ログ・レベル（デフォルト）
LOG_LEVEL_WARNING	リカバリ可能なエラーを示す警告ログ・レベル
LOG_LEVEL_INFO	情報メッセージのみを示す情報ログ・レベル
LOG_LEVEL_VERBOSE	デバッグ・メッセージのみを示す冗長ログ・レベル

public static boolean logStatus(int loglevel)

OTAS により設定された現行のログ・レベルが loglevel 引数より上位の場合は、true を戻します。

パラメータ

`loglevel`: 書き込むログ・メッセージのターゲット・ログ・レベル

public static void log(int loglevel, String message)

特定のメッセージをログ・ファイルに書き込みます。アダプタでは、logStatus メソッドが true を戻す場合にのみメッセージをログに書き込む必要があります。

パラメータ

- `loglevel`: 書き込むログ・メッセージのターゲット・ログ・レベル
- `message`: 書き込むメッセージ

5.9.2 クラス ErrorCodes

ErrorCodes オブジェクトは、TeleDeviceException および errorEvent メソッドとともに使用される一意のエラー条件を表します。

ErrorCodes には、異なる ErrorCode 値を表す ErrorCodes オブジェクトの静的定数がすべて保持されます。

静的定数

静的フィールド	説明
MISC_ERROR	その他のエラー。エラー・メッセージに説明があります。
MISC_INVALID_STATE	スイッチ・オブジェクトが無効な操作状態です。
USER_ALREADY_LOGGED_IN	ユーザーがすでに別のテレセットにログインしているため、ログインに失敗しました。

5.9.3 クラス ConsultCallTypes

ConsultCallTypes オブジェクトは、各種 API メソッドで引数として使用される一意のコンサルタント・コール・タイプを表します。

ConsultCallTypes には、異なる ConsultCallType 値を表す ConsultCallType オブジェクトの静的定数がすべて保持されます。

静的定数

静的フィールド	説明
TRANSFER	転送に関するコンサルタント・コール
CONFERENCE	会議に関するコンサルタント・コール

5.9.4 クラス TeleDeviceEventMulticaster

クラス TeleDeviceEventMulticaster は、TelesetEventListener および RoutePointEventListener のリストを管理するためのユーティリティ・クラスです。このクラスは、次の例のように TelesetDevice および RoutePointDevice インタフェースの実装で使用する必要があります。

```
public class TelesetDeviceImpl implements TelesetDevice
{
    TelesetEventListener    m_listener ;
    public void addTelesetEventListener(TelesetEventListener eventListener)
    {
     m_listener = TeleEventMulticaster.add(m_listener,eventListener);
    }
    public void removeTelesetEventListener(TelesetEventListener eventListener)
    {
         m_listener = TeleEventMulticaster.remove(m_listener,eventListener);
    }
    public void fireCallRingingEvent(int lineIndex)
    {
        if(m_listener != null)
        {
        m_listener.callRingingEvent(lineIndex);
        }
    }
    .... implemation of other methods ....
}
```

public static RoutePointEventListener add(RoutePointEventListener listener, RoutePointEventListener toBeAdded)

定義

新規の RoutePointEventListener を追加します。

パラメータ

- `listenerList`: オリジナルのリスナー（リスト）
- `toBeAdded`: 追加するリスナー

public static RoutePointEventListener remove(RoutePointEventListener listener, RoutePointEventListener toBeRemoved)

定義

RoutePointEventListener を削除します。

パラメータ

- `listenerList`: オリジナルのリスナー（リスト）
- `toBeRemoved`: 削除するリスナー

public static TelesetEventListener add(TelesetEventListener listener, TelesetEventListener toBeAdded)

定義

新規の TelesetEventListener を追加します。

パラメータ

- `listenerList`: オリジナルのリスナー（リスト）
- `toBeAdded`: 追加するリスナー

public static TelesetEventListener remove(TelesetEventListener listener, TelesetEventListener toBeRemoved)

定義

TelesetEventListener を削除します。

パラメータ

- `listenerList`: オリジナルのリスナー（リスト）
- `toBeRemoved`: 削除するリスナー

5.10 Windows NT と Windows 2000 のその他 API

この項では、Windows NT および Windows 2000 オペレーティング・システム用の SDK で使用可能なその他の C 関数について説明します。これらの関数は、ヘッダー・ファイル occt_pub.h および Hashtable.h に定義されています。この項の内容は、次のとおりです。

- 定数
- occtLogPrintf 関数
- OHashtable 関数
- OcctTelesetDevice 関数
- OcctRoutePointDevice 関数

5.10.1 定数

定数は、ログ・レベル、コンサルタント・タイプおよびエラー・コードについて occt_pub.h に定義されています。

ログ・レベル	ConsultCall のタイプ
VERBOSE_LEVEL	CODE_CONFERENCE
INFO_LEVEL	CODE_TRANSFER
WARNING_LEVEL	
ERROR_LEVEL	
FATAL_LEVEL	

エラー・タイプ	エラー・コード
TYPE_UNKNOWN	CODE_SUCCESS
TYPE_TELESET_CREATE_TELESET_DEVICE	CODE_FAILURE
TYPE_TELESET_ANSWER_CALL	CODE_FUNCTION_NOT_SUPPPORTED
TYPE_TELESET_RELEASE_CALL	CODE_LINE_INDEX_OUT_OF_RANGE
TYPE_TELESET_HOLD_CALL	CODE_NO_CURRENT_CALL
TYPE_TELESET_RETRIEVE_CALL	CODE_DESTINATION_BUSY
TYPE_TELESET_CONSULTATION_CALL	CODE_DESTINATION_INVALID
TYPE_TELESET_COMPLETE_TRANSFER	CODE_AGENT_INVALID
TYPE_TELESET_COMPLETE_CONFERENCE	CODE_AGENT_PASSWORD_INVALID
TYPE_TELESET_BLIND_TRANSFER	CODE_TELESET_IN_USE
TYPE_TELESET_SWAP_WITH_HELD	CODE_TELESET_INVALID
TYPE_TELESET_SEND_DTMF	CODE_SWITCH_CONNECTION_LOST
TYPE_TELESET_LOGIN_AGENT	CODE_SWITCH_RECONNECT_SUCCESS
TYPE_TELESET_LOGOUT_AGENT	
TYPE_TELESET_AGENT_READY	
TYPE_TELESET_AGENT_NOT_READY	
TYPE_ROUTE_POINT_CREATE_ROUTE_POINT_DEVICE	
TYPE_ROUTE_POINT_ASSIGN_MEDIA_ITEM_ID	

エラー・タイプ	エラー・コード
TYPE_ROUTE_POINT_ROUTE_CALL	
TYPE_NETWORK	

5.10.2 occtLogPrintf 関数

関数プロトタイプ

```
void occtLogPrintf(LogLevelCodes level,const char* tmpl,...);
```

定義

occtLogPrintf 関数は、Java 用 SDK の Logger クラスと同様です。この関数は OTAS ランタイムのロギング・サービスにアクセスし、ログ・メッセージを標準 OTAS ログ・ファイルに書き込みます。この関数の動作は、書式化されたオペランドを OTAS ログ・ファイルに書き込むという点で、通常の標準 C printf 関数に似ています。引数オペランドは、tmpl オペランドの制御により書式化されます。

パラメータ

- `level`: このメッセージのログ・レベル
- `tmpl`: "string = %s, int = %d\n" など、C printf スタイルのテンプレート
- `...`: テンプレートのトークンに対応する引数の変数リスト

5.10.3 OHashtable 関数

C 用の Oracle Telephony Adapter SDK には、ハッシュテーブルの単純な実装が用意されています。開発者は、これらの関数を使用してハッシュテーブルにデータを作成して操作できます。

5.10.3.1 occtHashtableNew

関数プロトタイプ

```
OHashtable* occtHashtableNew();
```

定義

新規の OHashtable を作成します。

5.10.3.2 occtHashtableGet

関数プロトタイプ

```
void* occtHashtableGet ( OHashtable* ht, char* key);
```

定義

キーの値を取得します。

パラメータ

- `ht`: OHashtable へのポインタ
- `key`: キー

5.10.3.3 occtHashtablePut

関数プロトタイプ

```
void* occtHashtablePut ( OHashtable* ht, char* key, void* value);
```

定義

ハッシュテーブルにキー - 値ペアを入れます。

パラメータ

- `ht`: OHashtable へのポインタ
- `key`: キー
- `value`: 値

5.10.3.4 occtHashtableDestroy

関数プロトタイプ

```
void occtHashtableDestroy( OHashtable* ht);
```

定義

OHashtable を破棄し、この構造体に割り当てられていたリソースをすべて解放します。

パラメータ

`ht`: OHashtable へのポインタ

5.10.3.5 occtHashtableSize

関数プロトタイプ

```
int occtHashtableSize( OHashtable* ht);
```

定義

ハッシュテーブルのサイズを取得します。

パラメータ

- `ht`: OHashtable へのポインタ
- `key`: キー

5.10.3.6 occtHashtableGetKeys

関数プロトタイプ

```
char** occtHashtableGetKeys( OHashtable* ht, int *len);
```

定義

ハッシュテーブルからすべてのキーを取得します。

パラメータ

- `ht`: OHashtable へのポインタ
- `len`: int 変数へのポインタ。この変数に設定されるキーの合計数は、キーの配列（2 ディメンションの char 配列）を戻します。

5.10.3.7 occtHashtableRemoveKey

関数プロトタイプ

```
void* occtHashtableRemoveKey ( OHashtable* ht, char* key);
```

定義

ハッシュテーブルからキー - 値ペアを削除します。

パラメータ

- `ht`: OHashtable へのポインタ
- `key`: キー

5.10.4 OcctTelesetDevice 関数

各 OcctTelesetDevice ポインタは、OHashtable に内部的に関連付けられます。開発者は、次の関数を使用して、ハッシュテーブルに格納されたデータにアクセスできます。

5.10.4.1 occtTelesetDeviceGet

関数プロトタイプ

```
void* occtTelesetDeviceGet(OcctTelesetDevice* device, char* key);
```

定義

キーの値を取得します。

パラメータ

- `device`: OcctTelesetDevice 構造体へのポインタ
- `key`: 値を取得するためのキー

5.10.4.2 occtTelesetDevicePut

関数プロトタイプ

```
void* occtTelesetDevicePut(OcctTelesetDevice* device, char* key,void* value);
```

定義

ハッシュテーブルにキー - 値ペアを入れます。

パラメータ

- `device`: OcctTelesetDevice 構造体へのポインタ
- `key`: キー
- `value`: 値

5.10.4.3 occtTelesetDeviceRemoveKey

関数プロトタイプ

```
void* occtTelesetDeviceRemoveKey(OcctTelesetDevice* device, char* key);
```

定義

ハッシュテーブルからキー - 値ペアを削除します。

パラメータ

- `device`: OcctTelesetDevice 構造体へのポインタ
- `key`: キー

5.10.4.4 occtTelesetDeviceGetKeys

関数プロトタイプ

```
char** occtTelesetDeviceGetKeys(OcctTelesetDevice* device, int *len);
```

定義

ハッシュテーブルからすべてのキーを取得します。

パラメータ

- `device`: OcctTelesetDevice 構造体へのポインタ
- `len`: int 変数へのポインタ。この変数に設定されるキーの合計数は、キーの配列（2 ディメンションの char 配列）を戻します。

5.10.5 OcctRoutePointDevice 関数

各 OcctRoutePointDevice ポインタは、OHashtable に内部的に関連付けられます。開発者は、次の関数を使用して、ハッシュテーブルに格納されたデータにアクセスできます。

5.10.5.1 occtRoutePointDeviceGet

関数プロトタイプ

```
void* occtRoutePointDeviceGet(OcctRoutePointDevice* device, char* key);
```

定義

キーの値を取得します。

パラメータ

- `device`: OcctRoutePointDevice 構造体へのポインタ
- `key`: 値を取得するためのキー

5.10.5.2 occtRoutePointDevicePut

関数プロトタイプ

```
void* occtRoutePointDevicePut(OcctRoutePointDevice* device, char* key,void* value);
```

定義

ハッシュテーブルにキー - 値ペアを入れます。

パラメータ

- `device`: OcctRoutePointDevice 構造体へのポインタ
- `key`: キー
- `value`: 値

5.10.5.3 occtRoutePointDeviceRemoveKey

関数プロトタイプ

```
void* occtRoutePointDeviceRemoveKey(OcctRoutePointDevice* device, char* key);
```

定義

ハッシュテーブルからキー - 値ペアを削除します。

パラメータ

- `device`: OcctRoutePointDevice 構造体へのポインタ
- `key`: キー

5.10.5.4 occtRoutePointDeviceGetKeys

関数プロトタイプ

```
char** occtRoutePointDeviceGetKeys(OcctRoutePointDevice* device, int *len);
```

定義

ハッシュテーブルからすべてのキーを取得します。

パラメータ

- `device`: OcctRoutePointDevice 構造体へのポインタ
- `len`: int 変数へのポインタ。この変数に設定されるキーの合計数は、キーの配列（2 ディメンションの char 配列）を戻します。

6

Telephony Adapter のテスト

Telephony Adapter 実装のテストには、Oracle Telephony SDK 統合テスト・ユーティリティを使用します。このユーティリティは、Oracle Telephony Adapter サーバーによりロードされ、実行されます。Oracle Telephony SDK 統合テスト・ユーティリティには、Oracle Telephony Adapter サーバー、検証ツールおよび TeleDevice テスト・ユーティリティを実行するためのユーザー・インタフェースが用意されています。

開発者は、Oracle Telephony SDK 統合テスト・ユーティリティを使用して TelesetDevice および RoutePointDevice 実装のすべての機能をテストできます。

この項の内容は、次のとおりです。

- 統合テスト・ユーティリティの概要
- Oracle Telephony Adapter サーバー（OTAS）の使用
- TeleDevice テスト・ユーティリティの使用
- 検証ツールの使用

6.1 Oracle Telephony SDK 統合テスト・ユーティリティの概要

ここでは、Oracle Telephony SDK 統合テスト・ユーティリティの全般的な機能と使用について説明します。

6.1.1 テスト・ユーティリティの起動と停止

Oracle Telephony SDK 統合テスト・ユーティリティを起動するには、ディレクトリ oracle/apps/cct/bin 内でファイル sdkrun.cmd を実行します。

Oracle Telephony SDK 統合テスト・ユーティリティのインタフェースが開きます。

Oracle Telephony SDK 統合テスト・ユーティリティを停止するには、次の 2 つの方法があります。

- ウィンドウを閉じます。
- 「ファイル」>「終了」を選択します。

6.1.2 Oracle Telephony SDK 統合テスト・ユーティリティ構成の保存

「ファイル」>「保存」を選択すると、Oracle Telephony SDK 統合テスト・ユーティリティの構成設定を保存できます。

6.1.3 Oracle Telephony SDK 統合テスト・ユーティリティのコンポーネント

Oracle Telephony SDK 統合テスト・ユーティリティのインタフェースは、メイン・ウィンドウと次の 4 つの内部ウィンドウで構成されています。

- 「ヘルプ」ウィンドウ。Javadoc が表示されます。
- 「Oracle Telephony Adapter サーバー」ウィンドウ。Oracle Telephony Adapter サーバーの構成と実行に使用します。
- 「TeleDevice テスト・ユーティリティ」ウィンドウ。TelesetDevice、RoutePointDevice およびソフトフォンの構成と実行に使用します。
- 「検証ツール」ウィンドウ。検証ツールの構成と実行に使用します。

各ウィンドウは、「表示」メニューのウィンドウ・オプションをオンにして表示したり、オフにして閉じることができます。「ウィンドウ」>「重ねて表示」または「ウィンドウ」>「並べて表示」を選択すると、各ウィンドウを移動、最小化、最大化またはサイズ変更したり、並べ替えることができます。

6.1.4 Javadoc の表示

Oracle Telephony Adapter SDK の Javadoc 生成 API のドキュメントを表示するには、「ヘルプ」ウィンドウの「インタフェース・サマリー」のリンクをクリックします。各クラスとインタフェースの詳細が表示されます。

6.2 Oracle Telephony Adapter サーバー（OTAS）の使用

Oracle Telephony Adapter サーバー（OTAS）のユーザー・インタフェースは、OTAS の起動と停止およびログの構成と表示に使用する「サーバー」タブと、CTI ミドルウェアの構成と OTAS の起動に使用する「ミドルウェア」タブで構成されています。

この項の内容は、次のとおりです。

- OTAS ログの構成
- CTI ミドルウェアの構成
- Oracle Telephony Adapter サーバーの起動と停止

6.2.1 OTAS ログの構成

次の手順に従って OTAS ログ・ファイルを構成します。

前提条件

Oracle Telephony Adapter SDK がインストールおよび構成されていること。

手順

1. 「Oracle Telephony Adapter サーバー」ウィンドウの「サーバー」タブをクリックします。
2. 「ログ・レベル」リストで、次の 4 つの OTAS ログ・レベルから 1 つ選択します。
   - 「エラー」。エラー・メッセージのみが表示されます。
   - 「警告」。エラー・メッセージと警告メッセージが表示されます。
   - 「情報」。エラー・メッセージ、警告メッセージおよび情報メッセージが表示されます。
   - 「冗長」。すべてのメッセージが表示されます。
3. 「ログ・ディレクトリ」フィールドに、ログ・ファイルのディレクトリ・パスを入力します。OTAS により作成されるログ・ファイルには、OTASyyyymmdd_hhmmss.log 形式の名前が使用されます。デフォルトのディレクトリは oracle/apps/cct/bin です。
4. 「ファイル」>「保存」を選択して構成を保存します。
5. 必要な場合は、右マウス・メニューで「消去」を選択し、「ログ」ウィンドウのログ・メッセージを消去します。

6.2.2 CTI ミドルウェアの構成

CTI ミドルウェア構成は、次の 3 タイプのデータで構成されています。

- 「ミドルウェア・タイプ」、「ライブラリ名」および「ファクトリ・クラス名」など、OTAS で適切なタイプのアダプタをロードするために必要なデータ。
- IP アドレスとポートおよびカスタム属性など、アダプタをサード・パーティの CTI ミドルウェア・システムに接続するために必要なカスタム・データ。このデータは、TeleDeviceFactory.init() メソッドに渡されます。
- Oracle Telephony Manager でコールの実行位置に基づいてローカル・ダイヤル文字列を生成するために必要な、「ダイヤル」セクション内のダイヤル・データ。通常、開発者が「ダイヤル」セクションのデータを使用する必要はありません。

次の手順に従って CTI ミドルウェアを OTAS 用に構成します。

前提条件

Oracle Telephony Adapter SDK がインストールおよび構成されていること。

手順

1. 「Oracle Telephony Adapter サーバー」ウィンドウの「ミドルウェア」タブをクリックします。
2. 次のどちらかを実行します。
   - Java で開発されたアダプタの場合 :
   - **a.** 「ミドルウェア・タイプ」リストから**「Java アダプタ」**を選択します。
   - **b.** 「ファクトリ・クラス名」フィールドに、TeleDeviceFactory クラス名を入力します。
   - C で開発されたアダプタの場合 :
   - **a.** 「ミドルウェア・タイプ」リストから**「C アダプタ」**を選択します。
   - **b.** 「ライブラリ名」フィールドに、DLL ライブラリ名を入力します。
3. 他のすべてのフィールドは、ハッシュテーブルに格納されるキー値ペアとして TeleDeviceFactory.init() メソッドに渡されます。各フィールドに使用されるハッシュテーブルのキーは、インタフェース Constants（Java の場合）と occt_pub.h（C の場合）に定義され、次の表にリストされます。

表 6-1 OTAS のフィールドとハッシュテーブルのキー

フィールド	ハッシュテーブルのキー（Java）
CTI サーバー IP アドレス 1	KEY_CTI_SERVER_IP_1
CTI サーバー IP アドレス 2	KEY_CTI_SERVER_IP_2
CTI サーバー・ポート 1	KEY_CTI_SERVER_PORT_1
CTI サーバー・ポート 2	KEY_CTI_SERVER_PORT_2
アダプタ・サーバー情報 1	KEY_ADAPTER_SERVER_INFO_1
アダプタ・サーバー情報 2	KEY_ADAPTER_SERVER_INFO_2
アダプタ・サーバー情報 3	KEY_ADAPTER_SERVER_INFO_3
アダプタ・サーバー情報 4	KEY_ADAPTER_SERVER_INFO_4
アダプタ・サーバー情報 5	KEY_ADAPTER_SERVER_INFO_5

表 6-1 OTAS のフィールドとハッシュテーブルのキー（続き）

フィールド	ハッシュテーブルのキー（Java）
アダプタ・サーバー情報 6	KEY_ADAPTER_SERVER_INFO_6
国内発信番号	KEY_DOMESTIC_PREFIX
国際発信番号	KEY_IDD_PREFIX
発信番号	KEY_OUTGOING_PREFIX
サイト市外局番	KEY_SITE_AREA_CODE
サイト国番号	KEY_SITE_COUNTRY_CODE
サイト国内番号桁数	KEY_SITE_INTERNAL_NUM_LENGTH
サイト・ローカル番号最大桁数	KEY_SITE_LOCAL_NUM_MAX_LENGTH
サイト・オーバーレイ	KEY_SITE_OVERLAY
パッシブ・モード	KEY_PASSIVE_MODE

4. 「ファイル」>「保存」を選択して構成を保存します。

6.2.3 Oracle Telephony Adapter サーバーの起動と停止

OTAS を起動するには、「Oracle Telephony Adapter サーバー」ウィンドウの「サーバー」タブで「開始」をクリックします。

OTAS が、バックグラウンドの Java プロセスとして現行のミドルウェア構成を使用して起動します。OTAS 起動後に行った構成変更は、OTAS を停止して再起動した後に有効となります。

OTAS を停止するには、「Oracle Telephony Adapter サーバー」ウィンドウの「サーバー」タブで「停止」をクリックします。

OTAS バックグラウンド・プロセスが終了します。

6.3 TeleDevice テスト・ユーティリティの使用

TeleDevice テスト・ユーティリティは、TelesetDevice オブジェクトと RoutePointDevice オブジェクトを制御し、モニターするためのユーザー・インタフェースです。OTAS のクライアントとして、開発者は TeleDevice テスト・ユーティリティを使用して TeleDevice および RoutePointDevice の実装で API メソッドを起動できます。TeleDevice および RoutePointDevice の実装から送信されるイベントは、TeleDevice テスト・ユーティリティで受信され、ユーザー・インタフェースに表示されます。また、開発者は TeleDevice テスト・ユーティリティを使用してソフトフォンを起動し、TelesetDevice 実装のユニット・テストを実行できます。

TeleDevice テスト・ユーティリティのユーザー・インタフェースは、メッセージ・ウィンドウと TelesetDevice および RoutePointDevice クライアントの構成と起動に使用する「デバイス」タブで構成されています。

「デバイス」タブには「テレセット」と「ルート・ポイント」の 2 つのサブタブがあり、それぞれに 10 行の表が含まれています。各行が 1 つの TeleDevice オブジェクトを表します。

この項の内容は、次のとおりです。

- TelesetDevice の割当て
- RoutePointDevice の割当て
- TelesetDevice と RoutePointDevice の割当て解除
- OTAS への TeleDevice または RoutePointDevice の接続
- イベントの表示
- API メソッドの起動
- TeleDevice の切断
- ソフトフォンの起動とクローズ

6.3.1 TelesetDevice の割当て

OTAS に接続できるように、TeleDevice オブジェクトを特定の構成に関連付ける必要があります。この関連付けは、TeleDevice テスト・ユーティリティでは「割当て」と呼ばれます。最大 10 の TeleDevice と 10 の RoutePointDevice を割り当てることができます。

次の手順に従って TelesetDevice を割り当てます。

前提条件

Oracle Telephony Adapter SDK をインストールして構成します。

手順

1. 「TeleDevice テスト・ユーティリティ」ウィンドウの「テレセット」タブをクリックします。
2. 割り当てる行を選択します。
3. 選択した行を右クリックします。

   ポップアップ・メニューが表示されます。
4. **「割当」**を選択します。

   「テレセットの割当」ウィンドウが表示されます。
5. 3 つの「内線」フィールドに回線の内線番号を入力します。

6. **「OK」**をクリックします。

   メッセージ・ウィンドウに、テレセット割当ての進行状況と完了を示すメッセージが表示されます。

7. 「ファイル」>「保存」を選択して構成を保存します。

6.3.2 RoutePointDevice の割当て

OTAS に接続できるように、TeleDevice オブジェクトを特定の構成に関連付ける必要があります。この関連付けは、TeleDevice テスト・ユーティリティでは「割当て」と呼ばれます。最大 10 の RoutePointDevice を割り当てることができます。

次の手順に従って RoutePointDevice を割り当てます。

前提条件

Oracle Telephony Adapter SDK がインストールおよび構成されていること。

手順

1. 「TeleDevice テスト・ユーティリティ」ウィンドウの「ルート・ポイント」タブをクリックします。
2. 割り当てる行を選択します。
3. 選択した行を右クリックします。

   ポップアップ・メニューが表示されます。

4. **「割当」**を選択します。

   「ルート・ポイントの割当」ウィンドウが表示されます。

5. ルート・ポイント番号を入力します。
6. このルート・ポイントを Inbound Telephony サーバーでモニターしない場合は、必要に応じて**「未モニタ」**をクリックします。
7. CT Connect のみを使用する Nortel Meridian の場合は、「即時処理」フィールドで、このルート・ポイントの CDN（制御ディレクトリ番号）で受信するインバウンド・コールの即時処理を指定します。呼出音の場合は ##R、ミュージックの場合は ##M、無音の場合は ##S を入力します。
8. Nortel Meridian で CT Connect のみを使用しており、手順 7 でミュージック処理（##M）を指定した場合は、「音楽ルート番号」フィールドに、Meridian PBX 内で構成されている音源のルート番号を指定します。# に続けて 16 進数で 2 桁のルート番号を入力してください。たとえば、音楽ルート番号が 10 の場合は、「音楽ルート番号」フィールドに `#0A` と入力します。

9. **「OK」**をクリックします。

   メッセージ・ウィンドウに、ルート・ポイント割当ての進行状況と完了を示すメッセージが表示されます。

10. 「ファイル」>「保存」を選択して構成を保存します。

6.3.3 TelesetDevice と RoutePointDevice の割当て解除

次の手順に従ってテレセット・デバイスまたはルート・ポイント・デバイスを割当て解除します。

前提条件

1 つ以上の TelesetDevice または RoutePointDevice を割り当てていること。

手順

1. 「TeleDevice テスト・ユーティリティ」ウィンドウの「テレセット」タブをクリックします。
2. 割当てを解除する行を選択します。
3. 選択した行を右クリックします。

   メニューが表示されます。

4. **「割当解除」**を選択します。

   割当てを解除した行に関連する構成が削除されます。

5. 「ファイル」>「保存」を選択して構成を保存します。

6.3.4 OTAS への TeleDevice または RoutePointDevice の接続

割当てを完了した TeleDevice を、OTAS に接続する必要があります。次の手順に従って TeleDevice を OTAS に接続します。

注意： TeleDevice または RoutePointDevice を接続する前に、OTAS が起動し、稼働中であることを確認してください。

前提条件

1 つ以上の TelesetDevice または RoutePointDevice を割り当てていること。

手順

1. 「TeleDevice テスト・ユーティリティ」ウィンドウの「テレセット」タブをクリックします。

2. 接続する行を選択します。
3. 選択した行を右クリックします。

   ポップアップ・メニューが表示されます。
4. **「接続」**を選択します。

   「TeleDevice」ウィンドウが表示され、「ルート・ポイント X が正常に接続されました」または「テレセット X が正常に接続されました」というメッセージが表示されます。
5. 「ファイル」>「保存」を選択して構成を保存します。

6.3.5 イベントの表示

TeleDevice オブジェクト実装により送信されるイベントが最大 30 までキャッシュされ、「イベント履歴」ウィンドウに表示できます。

「イベント履歴」ウィンドウを開くには、メニューから「イベント」>「イベント履歴」を選択します。

6.3.6 API メソッドの起動

TelesetDevice および RoutePointDevice API のメソッドは、「メソッド」メニューからメソッドを選択することで起動できます。

「メソッド」>「RoutePointDevice」を選択し、次のメソッドを起動します。

- assignMediaItemId
- routeCall

「メソッド」>「TelesetDevice」を選択し、次のメソッドを起動します。

- makeCall
- answerCall
- releaseCall
- holdCall
- retrieveCall
- loginAgent
- logoutAgent
- agentReadyOn
- agentReadyOff
- blindTransfer

- consultationCall(Transfer)
- completeTransfer
- consultationCall(Conference)
- completeConference

6.3.7 TeleDevice の切断

TeleDevice を切断するには、メニューから「切断」を選択するか、「TeleDevice」ウィンドウを閉じます。

6.3.8 ソフトフォンの起動とクローズ

次の手順に従ってソフトフォンを起動したりクローズします。

前提条件

なし

手順

1. 「テレセット」メニューから**「ソフトフォンの起動」**を選択します。

   「loginAgent」ウィンドウが表示されます。
2. acdAgentId、acdAgentPassword および acdQueue を入力します。
3. **「OK」**をクリックします。
4. ソフトフォンをクローズするには、「テレセット」メニューから**「ソフトフォンの終了」**を選択します。

6.4 検証ツールの使用

検証ツールにより次の機能が実行され、アダプタ実装のテストが自動化されます。

- 一連の共通テスト・ケースが実行され、実装の妥当性がチェックされます。
- 独自の Oracle 内部構文で記述されたテスト・ケース・ファイルが読み込まれます。
- テスト・ケースの一連の各ステップが順番に実行されます。
- テスト・ケースの各ステップの後にイベントが待機され、収集されます。
- テスト・ステップごとに、予期される（ソリューション・ファイルからの）イベント・セットを使用して、収集されたイベント・セットが検証されます。
- 検証結果が結果ファイルとログ・ウィンドウに出力されます。

この項の内容は、次のとおりです。

- 検証プロセスの構成
- エージェント情報の構成
- 検証ツールの起動と停止
- 検証ツール・ログの表示と消去

6.4.1 検証プロセスの構成

次の手順に従って、検証ツールの「制御」タブで、検証プロセスの起動と停止、入力ファイル、出力ファイルおよびログ・レベルの構成などの設定を構成します。

前提条件

Telephony Adapter の実装。

手順

1. 「検証ツール」ウィンドウの「制御」タブをクリックします。

   「制御」ページが表示されます。

2. 「ソリューション・ファイル」フィールドに、デフォルトのソリューション・ファイル Solutions.txt を入力するか、参照します。パスは、oracle/apps/cct/scripts ディレクトリです。異なるソリューション・ファイルを使用するには、「ソリューション・ファイル」ボックスにファイルのフルパスとファイル名を入力します。ソリューション・ファイルには、各 SDK API の起動について予期されるイベントとイベント詳細が含まれており、実行時に収集されたイベントが予期されるイベントと比較されます。

3. 「テスト・ケース・ファイル」フィールドに、デフォルトのテスト・ケース・ファイル TestCases.txt を入力するか、参照します。このファイルのパスは、oracle/apps/cct/scripts ディレクトリです。テスト・ケース・ファイルには、検証ツールにより実行されるテスト・ケースすべてのリストが含まれています。このファイルの各行は、特定のスクリプト・ファイルを指します。各スクリプト・ファイル名には、絶対ファイル名またはインストール環境の bin ディレクトリへの相対ファイル名を使用する必要があります。異なるテスト・ケース・ファイルを使用するには、必要に応じて「テスト・ケース・ファイル」フィールドにファイルのフルパスとファイル名を入力します。TestCases.txt ファイルを編集し、スクリプト・ファイルの実行順序を変更したり、実行するスクリプト・ファイルを追加または削除できます。

4. 「結果ファイル」フィールドに、デフォルトの結果ファイル output.txt を入力するか、参照します。結果ファイルは、検証ツールにより結果が書き込まれる出力ファイルです。異なる結果ファイルを使用するには、必要に応じて「結果ファイル」フィールドにファイルのフルパスとファイル名を入力します。
5. 「ログ・レベル」リストで、次の 4 つのログ・レベルから 1 つ選択します。
   - 「エラー」- エラー・メッセージのみ
   - 「警告」- エラー・メッセージと警告メッセージ
   - 「情報」- エラー・メッセージ、警告メッセージおよび情報メッセージ
   - 「冗長」- すべてのメッセージ
6. 「イベント遅延しきい値」リストから、スクリプトの各ステップが実行される間に検証ツールが待機するのに必要な秒数を選択します。「イベント遅延しきい値」の値は、コマンドが実行されてからテレフォニ・プラットフォームでイベントが受信されるまでの時間遅延以上の秒数に設定してください。
7. 「ファイル」>「保存」を選択して構成を保存します。

6.4.2 エージェント情報の構成

次の手順に従って、「構成」タブで、エージェントや、ツールの実行に必要なテレセット番号とルート・ポイント番号など、検証ツールの設定を構成します。

前提条件

- Telephony Adapter を実装します。
- 検証ツールの「制御」タブの設定を構成します。

手順

1. 「検証ツール」ウィンドウの「構成」タブをクリックします。

   「構成」ページが表示されます。
2. 検証ツールでは、ほとんどのテスト・ケースに 3 つのテレセット（エージェント A、エージェント B およびエージェント C）が使用されます。「回線 n の内線番号」フィールドに、各テレセットの回線内線番号を入力します。
3. テレセットごとに、AgentLoginId と AgentPassword を入力します。検証ツールでは、スクリプトの実行前に、これらの構成値が各テレセットへのログインに使用されます。

4. 一部のテスト・ケースでは、外部番号へのアウトバウンド・コールと、外部番号からのインバウンド・コールが必要です。これらのテスト・ケースで収集されたイベントの ANI 値と DNIS 値を検証するために、「外部番号」フィールドに、検証ツールでコールの実行とコールへの応答に使用できる外部電話番号を入力します。
5. また、ビジー番号と無効番号のコールが必要なテスト・ケースもあります。「ビジー番号」フィールドと「無効番号」フィールドに、これらのテスト・ケースで収集されたイベントの ANI 値と DNIS 値を検証するために検証ツールで使用できる電話番号を入力します。
6. 検証ツールでは、RoutePointDevice 実装のテストに、アクティブ・ルート・ポイントと受動ルート・ポイントの 2 つが使用されます。「アクティブ RP#」フィールドに、アクティブ・ルート・ポイントの内線番号を、「パッシブ RP#」フィールドに、受動ルート・ポイントの内線番号を入力します。
7. 「ファイル」>「保存」を選択して構成を保存します。

6.4.3 検証ツールの起動と停止

検証ツールを起動するには、「検証ツール」ウィンドウの「制御」タブをクリックし、「開始」をクリックします。

検証ツールは、現行の構成を使用してバックグラウンド・プロセスとして起動します。検証ツールの起動後に行った変更は、このツールを停止して再起動した後に有効になります。

検証ツールを停止するには、「検証ツール」ウィンドウの「制御」タブをクリックし、「停止」をクリックします。

バックグラウンド・プロセスが終了します。

6.4.4 検証ツール・ログの表示と消去

検証ツールの出力は、「制御」タブの「ログ」ウィンドウに表示されます。「ログ・レベル」リストを使用すると、異なるメッセージ・レベルを構成できます。ログ表示を消去するには、右マウス・メニューを開いて「消去」をクリックします。

7

Telephony Adapter の配布

この章では、Oracle Telephony Adapter をパッケージしてアップロードし、実装を本番の Interaction Center 環境に配布するための手順について説明します。配布手順の概略は、次のとおりです。

- Telephony Adapter のパッケージ
- ICSM への Telephony Adapter のアップロード
- CTI ミドルウェア、テレセットおよびルート・ポイントの構成
- サーバー・グループの構成
- Interaction Center、ソフトフォンおよび UWQ を使用したテスト

前提条件

- Oracle eBusiness Suite のインストール
- Oracle Interaction Center サーバー・マネージャのインストール

手順

1. Telephony Adapter をパッケージします。Oracle Telephony Adapter のファイル・パッケージのタイプは、アダプタ実装の開発言語に応じて異なります。

   次のどちらかを実行します。

   - Java で開発されたアダプタの場合は、impl.jar などの Java JAR ユーティリティを使用して、すべての Java クラスを 1 つの jar ファイルにパッケージする必要があります。
   - C で開発されたアダプタの場合は、すべてのコードを impl.dll などの 1 つの DLL ファイルにコンパイルします。

2. Oracle Telephony Adapter サーバーを配置するノードを識別します。

3. jar ファイルまたは dll ファイルを、Interaction Center サーバー・マネージャのインストール・ディレクトリ直下にある 3rdParty ディレクトリにコピーします。
4. CTI ミドルウェア、テレセットおよびルート・ポイントを構成します。Oracle Telephony Adapter サーバー（OTAS）でアダプタ実装のロードを開始する前に、次のどちらかの方法で CTI ミドルウェア構成を定義します。
   - Java アダプタ実装の場合は、「カスタム Java アダプタ」ミドルウェア・タイプを使用します。
   - C アダプタ実装の場合は、「カスタム C アダプタ」ミドルウェア・タイプを使用します。
5. サーバー・グループを構成します。OTAS を起動する前に、Oracle Telephony Adapter サーバー（OTAS）と関連サーバーを定義します。

関連項目

- 『Oracle Interaction Center Implementation Guide』
- 『Oracle Advanced Inbound Implementation Guide』

A

SDK 条件設定ワークシート

次の質問項目を使用して、特定の顧客サイトで Oracle Telephony Adapter SDK を統合する必要があるかどうかと、統合が必要な場合の SDK 要件を判断します。

1. サポートされる PBX/ACD

- 名称、バージョンおよびリリース（現行またはレガシー、レガシーの場合はアップグレード計画の有無）_____
  _____
- CTI インタフェースの有無
  - あり
  - なし
- 使用する ACD テレセット _____
- 各エージェントのテレセットに表示する内線 / 回線数 _____
- インバウンド・コールを受信する回線 _____
- アウトバウンド・コールに使用する回線 _____

2. サポートされる CTI ミドルウェア（存在する場合）

- 名称、バージョンおよびリリース _____
  - サーバー・サイドの統合のサポート
  - C API
  - Java API

3. 顧客側に CTI ミドルウェアもミドルウェア作業環境もない場合の、CTI ミドルウェアによる ACD/PBX サポートの有無

- なし
- 次の CTI ミドルウェアによる ACD/PBX のサポートあり
  - CT Connect（Java または C API）
  - Cisco ICM（C API のみ）
  - Genesys（C API のみ）

4. Interaction Center のエージェント数

単一サイト上で OTAS/CCC ごとに 1000 エージェント

5. 顧客が CTI 対応を希望しているサイト数

- Oracle Advanced Inbound のこのリリースの Oracle Telephony Adapter SDK では、単一サイトのルーティング、キューイングおよび配分がサポートされます。
- 顧客側に複数のサイト、マルチサイトのルーティング、コールおよびデータ転送機能が存在する場合、顧客は Cisco ICM または Genesys など、マルチサイト機能を持つ CTI ミドルウェアを候補として検討できます。

6. （#5 でサイト数を指定した場合のみ）顧客が複数サイト対応を希望している場合に必要な機能

- エンタープライズ・ルーティング（任意のサイトで着信し、任意のサイトに配分）
- エンタープライズ・コールおよびデータ転送

7. Oracle eBusiness Suite を導入済みの顧客かどうか

- 次のアプリケーションをインストール済み
  - TeleService
  - TeleSales
  - iSupport
  - iMeeting
  - Collections
  - iStore
  - Marketing Online

- 次のアプリケーションを購入予定
  - TeleService
  - TeleSales
  - iSupport
  - iMeeting
  - Collections
  - iStore
  - Marketing Online

8. 顧客が関心を持っている Oracle Advanced Inbound の機能

- コール・ルーティング
  - PBX スキル・ベース
  - DNIS
  - ビジネス・データのルーティング
  - ルール・ベースのルーティング
- スクリーンポップ
- その他のコール・データ（IVR の場合は #9 を参照）
- コールおよびデータ転送
- ソフトフォン GUI があるか、または必要か

9. Interaction Center の自動アウトバウンド・ダイヤル機能の有無（アウトバウンド・テレセールス・キャンペーン）

- あり。ダイヤル方法。
  - プレビュー : レコード配信エージェントがアウトバウンド・コールを開始
  - プログレシブ : エージェントへのレコード配信により自動的にダイヤル
  - プレディクティブ : エージェントに稼働している担当が転送された場合にのみシステムが顧客をダイヤル
- なし

10. Interaction Center の IVR の有無

- あり
  - IVR 製造業者 _____

- CTI 対応 / インテリジェント（IVR により IVR 終了コールのコール ID を保守）
- IVR が Oracle データベースに（PL/SQL）接続
- IVR がコールのアプリケーション・データ・フィールドを更新

– なし。

11. ライセンス収入合計

- E-Business Suite_____
- Interaction Center_____

12. 実装プロジェクトのスケジュール _____
_____

13. コンサルティング予算 _____
_____

B

SDK のスコープ分析

次のワークシートを使用して、顧客サイトの SDK 要件を決定します。

1. Oracle Advanced Inbound Development は、この PBX- ミドルウェアの組合せを保証またはサポートしていますか、または保証パイプラインに含まれていますか。

- はい
- いいえ

2. この PBX- ミドルウェアの組合せが Oracle Advanced Inbound Development により保証またはサポートされていない場合は、オラクル社のコンサルタントまたは Oracle SSI 提供の Telephony Adapter ソリューションを利用できますか、またはソリューションが進行中ですか。

利用できるソリューションをチェックするには、Oracle Solution Services ePortal（http://i3sn011e.idc.oracle.com:7777/servlet/page?_pageid=180,85,89,153,115,123,101,159,111,133,137,149,163&_dad=portal30&_schema=PORTAL30）を参照してください。

- はい
- いいえ

3. この PBX と CTI ミドルウェアについてコンサルティング・ベースのパイプラインを利用できますか。

オラクル社のプロフェッショナル・コミュニティ（itccon_us のメーリング・リストなど）に同様のケースがあるかどうか問い合せることを考慮してください。

- はい
- いいえ

4. PBX 製品の調査とデータ収集

a. モデル :

b. 現行のリリース :

c. CTI インタフェース :

d. ルート・ポイントとルーティングの動作 :

e. 独自の PBX CTI インタフェースにおけるソフトウェア CTI トランスレータの必要性

- あり
- なし

f. PBX による、コールのルーティングを実行する隣接プラットフォーム、Oracle Advanced Inbound のサポートの有無

- あり
- なし

g. PBX/ACD のタイプ :

- TDM ベース
- IP ベース

h. この PBX の ACD（自動コール分配）機能の有無

- あり
- なし

i. プラットフォーム :

- CSTA CTI 規格（PBX は CSTA スイッチ）
- 独自のコール・モデル

5. CTI ミドルウェア

a. 製造業者が CTI ミドルウェアの既存の主流ブランドかどうか

- はい
- いいえ

b. CTI ミドルウェア :

- CT Connect
- Cisco ICM
- Genesys
- その他

c. この CTI ミドルウェアによる顧客の PBX プラットフォームのサポートの有無

- あり
- なし

d. 使用可能なドキュメント

-
-
-
-

e. 使用可能なツールとシミュレータ

-
-
-
-
-

f. API の開発者用ライセンスの必要性

- あり
- なし

g. API の開発者用ライセンスの入手可能性の有無

- あり
- なし

h. CTI ミドルウェアによるルーティング機能の有無

- あり
- なし

i. CTI ミドルウェアによる IVR 統合機能の有無
   - あり
   - なし

j. CTI ミドルウェアによるサーバー・レベル統合のサポートの有無
   - あり、C ベースの API
   - あり、Java ベースの API
   - なし

6. Advanced Inbound 要件に関する追加情報

- ログイン、ログアウト
- 通話可、通話不可
- コールの実行
- コールへの応答
- コールの転送
  - ルート・ポイントへのコンサルタント転送
  - エージェントへのブラインド転送
  - ルート・ポイントへのブラインド転送
- 会議コール
  - コンサルタント会議
  - エージェントへのブラインド転送
  - ルート・ポイントへのブラインド転送
- 保留
- コールおよびデータ転送

7. どのようなレポート処理要件がありますか。

- PBX レポート
- CTI ミドルウェア・レポート
- Interaction Center Intelligence レポート

8. 拡張アウトバウンド・プレディクティブは必要ですか。

- いいえ
- はい

a. PBX による VDU への回線サイドの T1 インタフェースのサポートの有無

- あり
- なし

b. CTI ミドルウェアによる回線サイド・アナログ PBX 内線のモニターと制御のサポートの有無

- あり
- なし

C

Telephony Adapter のテスト・ケース

各イベントは、特定の時点でチェックされる Oracle Telephony Manager イベントであり、[<Agent>/<EventName>] で示されています。可能な Oracle Telephony Manager イベントすべてが常にチェックされるわけではありません。

表 C-1 エージェントの状態

番号	テスト手順
A1	エージェント A の実際の電話でログアウトした状態で、ソフトフォンを使用してエージェント A にログインします。
A2	エージェント A の実際の電話でログインし、通話不可の状態で、ソフトフォンを使用してエージェント A にログインし、電話が自動的にログアウトされてからログインされるかどうかを確認します。
A3	エージェント A の実際の電話でログインし、通話可の状態で、ソフトフォンを使用してエージェント A にログインし、電話が自動的にログアウトされてからログインされるかどうかを確認します。
A4	エージェント A の実際の電話でログインし、コール中の状態で、ソフトフォンを使用してエージェント A にログインし、適切なエラー・メッセージが表示されるかどうかを確認して、電話の残りのコールを手動で切断します。ソフトフォンを使用して再度エージェント A にログインし、電話が自動的にログアウトされてからログインされるかどうかを確認します。
A5	エージェント A をログアウトします。
A6	エージェント A を通話可の状態に設定します。
A7	エージェント A を通話不可の状態に設定します。

表 C-2 コールの実行 / コールへの応答

番号	テスト手順
B1	A が B をコールします。B は応答しません。A が切断します。
B2	A が B をコールします。B が応答します [NO B/ExternalWithData]。A が切断します。
B3	A が B をコールします。B が応答します。B が切断します。
B4	A が X に対してアウトバウンド・コールを実行します。X は応答しません。A が切断します。
B5	A が X に対してアウトバウンド・コールを実行します。X が応答します [NO A/ExternalWithData]。A が切断します。
B6	A が X に対してアウトバウンド・コールを実行します。X が応答します。X が切断します。
B7	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A は応答しません。X が切断します。
B8	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A が切断します。
B9	A が X からインバウンド・コールを受信します。A が応答します。X が切断します。
B10	A がビジー状態の B をコールします。
B11	A がビジー状態の X に対してアウトバウンド・コールを実行します。
B12	A が無効な内部番号をコールします。
B13	A が無効な外部番号に対してアウトバウンド・コールを実行します。

表 C-3 保留 / 再接続

番号	テスト手順
C1	A が B をコールします。B が応答します。A がコール保留にします。A がコールに再接続します。
C2	A が B をコールします。B が応答します。B がコール保留にします。B がコールに再接続します。
C3	A が B をコールします。B が応答します。A がコール保留にします。B がコール保留にします。A がコールに再接続します。B がコールに再接続します。
C4	A が B をコールします。B が応答します。A がコール保留にします。B が切断します。

表 C-3 保留 / 再接続（続き）

番号	テスト手順
C5	A が B をコールします。B が応答します。B がコール保留にします。A が切断します。
C6	A が X に対してアウトバウンド・コールを実行します。X が応答します。A がコール保留にします。A がコールに再接続します。
C7	A が X に対してアウトバウンド・コールを実行します。X が応答します。A がコール保留にします。X が切断します。
C8	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A がコール保留にします。A がコールに再接続します。
C9	A が X からインバウンド・コールを受信します。A が応答します。A がコール保留にします。X が切断します。
C10	A が B をコールします。A がコール保留にします。B が応答します。A がコールに再接続します。
C11	A が X に対してアウトバウンド・コールを実行します。A がコール保留にします。X が応答します。A がコールに再接続します。

表 C-4 コンサルタント転送

番号	テスト手順
D1	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します [B/TransferWithData、コール・データは前述の A/ExternalWithData と同じ]。A が転送を完了します。
D2	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A が転送を完了します。
D3	A が X に対してアウトバウンド・コールを実行します。X が応答します [NO A/ExternalWithData]。A が B に対してコンサルタント・コールを実行します。B が応答します [NO B/TransferWithData]。A が転送を完了します。
D4	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A が転送を完了します。
D5	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答します [NO C/TransferWithData]。B が転送を完了します。

表 C-4 コンサルタント転送（続き）

番号	テスト手順
D6	A が B をコールします。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答します。B が転送を完了します。
D7	A が X からインバウンド・コールを受信します。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。B がコンサルタント・コールを保留にします。A が転送を完了します。B が転送されたコールに再接続します。
D8	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。B がコンサルタント・コールを保留にします。A が転送を完了します。B が転送されたコールに再接続します。
D9	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答します。C がコンサルタント・コールを保留にします。B が転送を完了します。C が転送されたコールに再接続します。
D10	A が B をコールします。B が応答します。A がコール保留にします。B が C に対してコンサルタント・コールを実行します。C が応答します [NO C/TransferWithData]。B が転送を完了します。A が転送されたコールに再接続します。
D11	A が B をコールします。B が応答します。A がコール保留にします。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答します。B が転送を完了します。A が転送されたコールに再接続します。

表 C-5 セクション E - ブラインド転送

番号	テスト手順
E1	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答する前に A が転送を完了します。B が応答します [B/TransferWithData]。
E2	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答する前に A が転送を完了します。Y が応答します。
E3	A が X に対してアウトバウンド・コールを実行します。X が応答します [NO A/ExternalWithData]。A が B に対してコンサルタント・コールを実行します。B が応答する前に A が転送を完了します。B が応答します [NO B/TransferWithData]。
E4	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答する前に A が転送を完了します。Y が応答します。

表 C-5 セクション E - ブラインド転送（続き）

番号	テスト手順
E5	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答する前に B が転送を完了します。C が応答します [NO C/TransferWithData]。
E6	A が B をコールします。B が応答します。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答する前に B が転送を完了します。X が応答します。
E7	A が B をコールします。B が応答します。A がコール保留にします。B が C に対してコンサルタント・コールを実行します。C が応答する前に B が転送を完了します。C が応答します [NO C/TransferWithData]。A が転送されたコールに再接続します。
E8	A が B をコールします。B が応答します。A がコール保留にします。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答する前に B が転送を完了します。X が応答します。A が転送されたコールに再接続します。

表 C-6 コンサルタント会議

番号	テスト手順
F1	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します [B/ConferenceWithData、コール・データは前述の A/ExternalWithData と同じ]。A が会議を完了します。
F2	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A が会議を完了します。
F3	A が X に対してアウトバウンド・コールを実行します。X が応答します [NO A/ExternalWithData]。A が B に対してコンサルタント・コールを実行します。B が応答します [NO B/ConferenceWithData]。A が会議を完了します。
F4	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A が会議を完了します。
F5	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答します [NO C/ConferenceWithData]。B が会議を完了します。
F6	A が B をコールします。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答します。B が会議を完了します。
F7	A が X からインバウンド・コールを受信します。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。B がコンサルタント・コールを保留にします。A が会議を完了します。B が会議済みコールに再接続します。

表 C-6 コンサルタント会議（続き）

番号	テスト手順
F8	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。B がコンサルタント・コールを保留にします。A が会議を完了します。B が会議済みコールに再接続します。
F9	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答します。C がコンサルタント・コールを保留にします。B が会議を完了します。C が会議済みコールに再接続します。
F10	A が B をコールします。B が応答します。A がコール保留にします。B が C に対してコンサルタント・コールを実行します。C が応答します [NO C/ConferenceWithData]。B が会議を完了します。A が会議済みコールに再接続します。
F11	A が B をコールします。B が応答します。A がコール保留にします。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答します。B が会議を完了します。A が会議済みコールに再接続します。

表 C-7 コンサルタント（転送 / 会議）/ 取消

番号	テスト手順
G1	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答する前に X が切断します。B が応答します [B/TransferWithData]。
G2	A が X からインバウンド・コールを受信します。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答する前に A がコンサルタント・コールを切断します。A が X に再接続します。
G3	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します [B/TransferWithData]。A が転送または会議を完了する前に X が切断します。
G4	A が X からインバウンド・コールを受信します。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。A がコンサルタント・コールを切断します。A が X に再接続します。
G5	A が X からインバウンド・コールを受信します。A が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。B がコンサルタント・コールを切断します。A が X に再接続します。
G6	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答する前に X が切断します。Y が応答します。

表 C-7 コンサルタント（転送 / 会議）/ 取消（続き）

番号	テスト手順
G7	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答する前に A がコンサルタント・コールを切断します。A が X に再接続します。
G8	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A が転送または会議を完了する前に X が切断します。
G9	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A がコンサルタント・コールを切断します。A が X に再接続します。
G10	A が X からインバウンド・コールを受信します。A が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。Y がコンサルタント・コールを切断します。A が X に再接続します。
G11	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が B に対してコンサルタント・コールを実行します。B が応答する前に X が切断します。B が応答します。
G12	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が B に対してコンサルタント・コールを実行します。B が応答する前に A がコンサルタント・コールを切断します。A が X に再接続します。
G13	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。A が転送または会議を完了する前に X が切断します。
G14	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。A がコンサルタント・コールを切断します。A が X に再接続します。
G15	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が B に対してコンサルタント・コールを実行します。B が応答します。B がコンサルタント・コールを切断します。A が X に再接続します。
G16	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答する前に X が切断します。Y が応答します。
G17	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答する前に A がコンサルタント・コールを切断します。A が X に再接続します。
G18	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A が転送または会議を完了する前に X が切断します。

表 C-7 コンサルタント（転送 / 会議）/ 取消（続き）

番号	テスト手順
G19	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。A がコンサルタント・コールを切断します。A が X に再接続します。
G20	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が Y に対してアウトバウンドのコンサルタント・コールを実行します。Y が応答します。Y がコンサルタント・コールを切断します。A が X に再接続します。
G21	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答する前に A が切断します。C が応答します。
G22	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答する前に B がコンサルタント・コールを切断します。B が A に再接続します。
G23	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答します。B が転送または会議を完了する前に A が切断します。
G24	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答します。B がコンサルタント・コールを切断します。B が A に再接続します。
G25	A が B をコールします。B が応答します。B が C に対してコンサルタント・コールを実行します。C が応答します。C がコンサルタント・コールを切断します。B が A に再接続します。
G26	A が B をコールします。B が応答します。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答する前に A が切断します。X が応答します。
G27	A が B をコールします。B が応答します。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答する前に B がコンサルタント・コールを切断します。B が A に再接続します。
G28	A が B をコールします。B が応答します。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答します。B が転送または会議を完了する前に A が切断します。
G29	A が B をコールします。B が応答します。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答します。B がコンサルタント・コールを切断します。B が A に再接続します。
G30	A が B をコールします。B が応答します。B が X に対してアウトバウンドのコンサルタント・コールを実行します。X が応答します。X がコンサルタント・コールを切断します。B が A に再接続します。
G31	A が X からインバウンド・コールを受信します。A が応答します。A がビジー状態の B に対してコンサルタント・コールを実行します。A が X に再接続します。

表 C-7 コンサルタント（転送 / 会議）/ 取消（続き）

番号	テスト手順
G32	A が X からインバウンド・コールを受信します。A が応答します。A がビジー状態の Y に対してアウトバウンドのコンサルタント・コールを実行します。A が X に再接続します。
G33	A が X からインバウンド・コールを受信します。A が応答します。A が無効な内部番号に対してコンサルタント・コールを実行します。A が X に再接続します。
G34	A が X からインバウンド・コールを受信します。A が応答します。A が無効な外部番号に対してアウトバウンドのコンサルタント・コールを実行します。A が X に再接続します。
G35	A が X に対してアウトバウンド・コールを実行します。X が応答します。A がビジー状態の B に対してコンサルタント・コールを実行します。A が X に再接続します。
G36	A が X に対してアウトバウンド・コールを実行します。X が応答します。A がビジー状態の Y に対してアウトバウンドのコンサルタント・コールを実行します。A が X に再接続します。
G37	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が無効な内部番号に対してコンサルタント・コールを実行します。A が X に再接続します。
G38	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が無効な外部番号に対してアウトバウンドのコンサルタント・コールを実行します。A が X に再接続します。
G39	A が B をコールします。B が応答します。B がビジー状態の C に対してコンサルタント・コールを実行します。B が A に再接続します。
G40	A が B をコールします。B が応答します。B がビジー状態の X に対してアウトバウンドのコンサルタント・コールを実行します。B が A に再接続します。
G41	A が B をコールします。B が応答します。B が無効な内部番号に対してコンサルタント・コールを実行します。B が A に再接続します。
G42	A が B をコールします。B が応答します。B が無効な外部番号に対してアウトバウンドのコンサルタント・コールを実行します。B が A に再接続します。

表 C-8 ルート・ポイント

番号	テスト手順
K1	A が R をコールします。R が B にルーティングします [NO B/ExternalWithData]。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答します [NO C/TransferWithData]。A が転送を完了します。
K2	A が R をコールします。R が B にルーティングします。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答する前に A が転送を完了します。C が応答します [NO C/TransferWithData]。
K3	A が R をコールします。R が B にルーティングします。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答します [NO C/ConferenceWithData]。A が会議を完了します。
K4	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答します [NO C/TransferWithData]。B が転送を完了します。
K5	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングする前に B が転送を完了します。R が C にルーティングします。C が応答します [NO C/TransferWithData]。
K6	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答する前に B が転送を完了します。C が応答します [NO C/TransferWithData]。
K7	A が B をコールします。B が応答します。A が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答します [NO C/TransferWithData]。A が転送を完了します。
K8	A が B をコールします。B が応答します。A が R に対してコンサルタント・コールを実行します。R が C にルーティングする前に A が転送を完了します。R が C にルーティングします。C が応答します [NO C/TransferWithData]。
K9	A が B をコールします。B が応答します。A が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答する前に A が転送を完了します。C が応答します [NO C/TransferWithData]。
K10	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答します [NO C/ConferenceWithData]。B が会議を完了します。
K11	A が R をコールします。R が B にルーティングします。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答する前に B が切断します。C が応答します [NO C/ConferenceWithData]。

表 C-8 ルート・ポイント（続き）

番号	テスト手順
K12	A が R をコールします。R が B にルーティングします。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答する前に A がコンサルタント・コールを切断します。
K13	A が R をコールします。R が B にルーティングします。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答します。B が切断します。
K14	A が R をコールします。R が B にルーティングします。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答します。A がコンサルタント・コールを切断します。
K15	A が R をコールします。R が B にルーティングします。B が応答します。A が C に対してコンサルタント・コールを実行します。C が応答します。C がコンサルタント・コールを切断します。
K16	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングする前に A が切断します。R が C にルーティングします。C が応答します。
K17	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答する前に A が切断します。C が応答します。
K18	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングする前に B がコンサルタント・コールを切断します。
K19	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答する前に B がコンサルタント・コールを切断します。
K20	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答します。A が切断します。
K21	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答します。B がコンサルタント・コールを切断します。
K22	A が B をコールします。B が応答します。B が R に対してコンサルタント・コールを実行します。R が C にルーティングします。C が応答します。C がコンサルタント・コールを切断します。
K23	A が X からインバウンド・コールを受信します [A/ExternalWithData]。A が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングします。B が応答します [B/TransferWithData]。A が転送を完了します。

表 C-8 ルート・ポイント（続き）

番号	テスト手順
K24	A が X からインバウンド・コールを受信します。A が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングする前に A が転送を完了します。R が B にルーティングします。B が応答します [B/TransferWithData]。
K25	A が X からインバウンド・コールを受信します。A が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングします。B が応答する前に A が転送を完了します。B が応答します [B/TransferWithData]。
K26	A が X からインバウンド・コールを受信します。A が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングします。B が応答します [B/ConferenceWithData]。A が会議を完了します。
K27	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングします。B が応答します [B/TransferWithData]。A が転送を完了します。
K28	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングする前に A が転送を完了します。R が B にルーティングします。B が応答します [B/TransferWithData]。
K29	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングします。B が応答する前に A が転送を完了します。B が応答します [B/TransferWithData]。
K30	A が X に対してアウトバウンド・コールを実行します。X が応答します。A が R に対してコンサルタント・コールを実行します。R が B にルーティングします。B が応答します [B/ConferenceWithData]。A が会議を完了します。

表 C-9 DTMF トーン

番号	テスト手順
L1	A が X からインバウンド・コールを受信します。A が応答します。A がいくつかの数字を押して「ダイヤル」ボタンをクリックします。X には、A がダイヤルした数字の DTMF トーンが聞こえます。
L2	A が X に対してアウトバウンド・コールを実行します。X が応答します。A がいくつかの数字を押して「ダイヤル」ボタンをクリックします。X には、A がダイヤルした数字の DTMF トーンが聞こえます。

表 C-9 DTMF トーン（続き）

番号	テスト手順
L3	A が B をコールします。B が応答します。A がいくつかの数字を押して「ダイヤル」ボタンをクリックします。B には、A がダイヤルした数字の DTMF トーンが聞こえます。
L4	A が B をコールします。B が応答します。B がいくつかの数字を押して「ダイヤル」ボタンをクリックします。A には、B がダイヤルした数字の DTMF トーンが聞こえます。

D

サンプル・コード

サンプル・コードは、oracle/apps/cct/java/sample ディレクトリにある 3 つの Java ソース・ファイルで構成されています。

- SampleTeleDeviceFactory.java
- SampleTelesetDevice.java
- SampleRoutePointDevice.java

これは、ターゲット・スイッチおよび CTI ミドルウェアとしてスイッチ・シミュレータを使用する Oracle Telephony Adapter の実装例です。次の機能が実装されます。

- エージェントのログイン / ログアウト
- エージェント通話可 / 通話不可
- コールの実行
- コールへの応答
- コールの解放
- コールのルーティング

D.1 スイッチ・シミュレータの概要

スイッチ・シミュレータは、実際のスイッチを利用できないデモンストレーション環境で使用される Interaction Center サーバー・プロセスの 1 つです。スイッチ・シミュレータでは、テレセット機能とルート・ポイント機能がサポートされ、OpenTel Bean API（Java API、パッケージ oracle.apps.cct.openTel.bean）を使用してプログラムによりアクセスできます。スイッチ・シミュレータにより、テレセットとルート・ポイントの各回線が内線オブジェクトとしてモデル化されます。内線オブジェクトには、OpenTelExtensionBean を使用してアクセスできます。イベントは、スイッチ・シミュレータにより OpenTel イベントとして送信されます。

詳細は、oracle.apps.cct.openTel.bean パッケージの Javadoc を参照してください。

D.2 サンプル・アダプタ設計

サンプル・アダプタの主な目的は、スイッチ・シミュレータに接続して Adapter SDK のメソッド・コールを OpenTel Bean API メソッド・コールに転送し、OpenTel イベントを処理し、それを Adapter イベントに変換することです。次に Interaction Center サーバー間の関連を示します。

（Interaction Center）<---> （ OTAS <--> サンプル・アダプタ）<---> （スイッチ・シミュレータ）

SampleTeleDeviceFactory

SampleTeleDeviceFactory は、init() メソッドのハッシュテーブルから IP アドレスとポートを読み込みます。

詳細は、ソース・コードと Javadoc を参照してください。

SampleTelesetDevice

SampleTelesetDevice は、回線ごとに OpenTelExtensionBean オブジェクトを 1 つ保持します。Adapter SDK のメソッドは、指定された回線インデックスに基づいて適切な OpenTelExtensionBean オブジェクトに送られます。SampleTelesetDevice は、スイッチ・シミュレータ・イベントを処理するために、OpenTelExtensionBean オブジェクトごとに専用 OpenTelEventListener を登録します。

詳細は、ソース・コードと Javadoc を参照してください。

SampleRoutePointDevice

SampleRoutePointDevice は、1 つの OpenTelExtensionBean オブジェクトを保持します。Adapter SDK のメソッドは、ルート・ポイントの OpenTelExtensionBean オブジェクトに送られます。SampleRoutePointDevice は、スイッチ・シミュレータ・イベントを処理するために、それ自体を OpenTelEventListener として登録します。

詳細は、ソース・コードと Javadoc を参照してください。

用語集

ACD

自動コール分配装置（Automatic Call Distribution）。Oracle Interaction Center エージェントへの着信に自動的に応答し、キューに入れ、ルーティングするように設計されているシステム。PBX の場合はユーザーが共有できる電話回線の数が限られていますが、ACD の場合はエージェントごとに 1 つ以上の電話回線があります。

ANI

自動番号識別（Automatic Number Identification）。通話者 ID と同様に、長距離通信事業者が通話者の請求番号を識別するために提供するサービス。

API

アプリケーション・プログラミング・インタフェース（Application Programming Interface）。ソフトウェア・アプリケーションでオペレーティング・システムや他のサービスへのアクセスに使用される呼出し規則。

CTI

コンピュータ・テレフォニ統合（Computer Telephony Integration）。コンピュータからスイッチに電話のダイレクト方法に関する指示が送信されるように、コンピュータを電話交換機（PBX または ACD）に接続するシステム。

DNIS

ダイヤル番号識別サービス（Dialed Number Identification Service）。コールを適切な内線にルーティングするテレフォニ・システムに対してコールされた番号を識別する、800 および 900 回線の機能。

IDE

対話型開発環境（Interactive Development Environment）。ソフトウェア記述プロセスを支援するシステム。IDE には、構文指向のエディタ、プログラム入力用のグラフィカル・ツール、プログラムをコンパイルして実行し、コンパイル・エラーをソースに関連付けるための統合サポート機能が含まれる場合があります。IDE の例には、Visual C++ や Visual Basic があります。

Inbound Telephony サーバー（Inbound Telephony Server: ITS）

インバウンドのテレフォニ顧客対応を処理する Oracle Interaction Center サーバー。ITS でサポートされる機能は、次のとおりです。

- （アクティブ・モードのみ）ITS は、CTI ミドルウェアからのルート問合せをリスニングし、それに応答してコールをルーティングするスイッチを指示することで、エンタープライズ・データ・ベースのルーティングを使用可能にします。
- ITS はルート・ポイントに着信するコールをモニターします。
- ITS はルート・ポイントで放棄されるコールを検出します。
- ITS は IVR からの IVR データ・パケットを受信します。

Interaction Center サーバー（interaction center server）

Oracle Interaction Queueing and Distribution、Oracle Universal Work Queue、Oracle Routing サーバーおよび Oracle Inbound Telephony サーバーなど、任意の Interaction Center サーバー。中間層サーバー・プロセス（mid-tier server process）およびサーバー・プロセス（server process）と同義。

Interaction Center サーバー・マネージャ（Interaction Center Server Manager: ICSM）

各ターゲット・システム上で明示的に起動する必要のある唯一のサーバー・プロセス。ICSM は、他のすべての Oracle Advanced Inbound サーバー・プロセスの起動、停止およびモニターを受け持ちます。ICSM サーバー・プロセスは、Interaction Center サーバーの HTML 管理機能により制御されます。

IVR

音声自動応答装置（Interactive Voice Response）。着信電話に応答し、通話者に対して電話のボタンを押してコールを 1 つ以上の内線にルーティングするオプションを提供する録音メッセージを再生するように自動化されたシステム。

Java Development Kit（JDK）

Java プログラミングに必要な環境を提供する Sun 社の製品。

Javadoc

プログラムから HTML ドキュメントを生成する Java Development Kit 内の機能。Javadoc はソース・コードを読み込み、特別な書式設定と位置設定を持つコメントをドキュメントへと解析します。

Java ネイティブ・インタフェース（Java Native Interface: JNI）

Java 固有のプログラミング・インタフェース。このインタフェースにより、C や C++ など、他のプログラム言語で記述されたアプリケーションおよびライブラリを、Java Virtual Machine 内で実行される Java コードで操作できます。

JDBC

Java Database Connectivity。Java Development Kit のうち、Java プログラムからデータベースへの標準的な SQL アクセスを行うための Java 用アプリケーション・プログラミング・インタフェースを定義する部分。

Observer API

Oracle Interaction Center に着信を通知し、ソフトフォン表示、スクリーンポップおよびコール・ルーティングに使用されるコール・データと通話者データを配信するために、アダプタで使用される API。Oracle Interaction Center により Observer API が実装されます。

Oracle Advanced Inbound

Oracle E-Business パッケージ内のビジネス・アプリケーションをテレフォニを使用可能にするために必要な Oracle E-Business アプリケーション。Oracle Advanced Inbound のサーバー・アーキテクチャは、単一の物理サイトまたは複数サイトで Interaction Center を実行するように拡張可能です。Oracle Advanced Inbound バンドル製品には、Call Center Technology、Oracle Universal Work Queue、Oracle Telephony Manager、Oracle Call Center Connectors および Oracle Interaction Blending があります。

Oracle Advanced Outbound

Oracle Advanced Inbound に対応するアウトバウンド・テレフォニ機能を提供する Oracle E-Business アプリケーション。

Oracle Interaction Center（OIC）

Oracle の E-Business Suite アプリケーションのテレフォニを使用可能にする基盤となるサーバー・プロセスのグループ。

Oracle Telephony Adapter サーバー（Oracle Telephony Adapter Server: OTAS）

Oracle Call Center Connectors にかわる CTI アダプタ・サーバー。Oracle Telephony Adapter サーバーには、スイッチごとに 1 つのテレフォニ・アダプタが組み込まれています。

Oracle Telephony Manager（OTM）

メディア項目のキューイング、ルーティングおよび分配を実行する Oracle Interaction Center アプリケーション。

PBX

構内交換機（Private Branch eXchange）。会社のユーザー間でコールを切り替えて多数の外部電話回線を共有可能にする、会社や他の組織内の電話システム。パッシブ・モードでは、コールは PBX ルートのコールによりルーティングされます。

Provider API

コールの実行、転送、ルーティングおよびエージェント・モードの変更など、テレフォニおよび Interaction Center 機能の要求をサポートする API。Provider API Provider はアダプタにより実装されます。

Software Development Kit

（SDK）ソフトウェア・ベンダーが自社製品を他のソフトウェア・ベンダーの製品とともに使用できるように提供するソフトウェア。

アクティブ・モード（active mode）

Oracle Advanced Inbound で構成されているビジネス・データとルールを使用して、コール・センター・エージェントへの着信のルーティングと分配が制御されるルーティング・モード。モニター対象の PBX/ACD ルート・ポイントに達した時点で、インバウンド・コールのすべての制御を Oracle Advanced Inbound に付与するには、特定の PBX/ACD 構成が必要です。

アダプタ（adapter）

Oracle Interaction Center を特定のスイッチおよび CTI ミドルウェア・プラットフォームに統合するために、特別に開発された Oracle Telephony Adapter サーバーのテレフォニ・ドライバ。オラクル社は、保証済みのスイッチとミドルウェアの組合せについてアダプタを開発しています。サード・パーティは、Oracle Telephony Adapter SDK を使用して、オラクル社が保証していないスイッチとミドルウェアの組合せ用にアダプタを開発できます。通常、各アダプタは特定の製造業者のテレフォニ・システムとのみ統合するように開発されます。

拡張性（scalability）

ソフトウェアまたはハードウェア製品の、将来のビジネス・ニーズに対する適合可能性を示す尺度。

拡張パッシブ・モード（enhanced passive mode）

PBX/ACD のルート・ポイントをモニターする Oracle Advanced Inbound でコール・センター・エージェントへのコールの標準的 PBX/ACD ルーティングおよび分配が発生するルーティング・モード。このモードでは、Oracle Interaction Center Intelligence によるレポート処理用に、ターゲットとなるスクリーンポップ用のコールの分類、インバウンド・コールのキュー件数およびルート・ポイントで放棄されるコールの追跡ができます。インバウンド・コールが Oracle Advanced Inbound によりモニターされている PBX/ACD ルート・ポイントを確実に通過するようにするには、特定の PBX/ACD 構成が必要です。

サーバー・ステータス（server status）

サーバー・プロセスが実行中かどうか、サーバーの実行期間などに関する情報。

サーバー・プロセス（server process）

Oracle Interaction Queueing and Distribution、Oracle Universal Work Queue、Routing サーバー、Oracle Inbound Telephony サーバーおよび Oracle Telephony メディア・コントローラなど、任意の Interaction Center サーバー。中間層サーバー・プロセス（mid-tier server process）および Interaction Center サーバー（interaction center server）と同義。

サイト（site）

Interaction Center の単一の所在地。通常、サイトには PBX と CTI ミドルウェアがインストールされています。

スーパー・グループ（super group）

サーバー・グループ階層の最上位にある親サーバー・グループ。

スイッチ・シミュレータ（switch simulator）

Intel CT Connect ミドルウェアを使用して、Nortel スイッチと Oracle Call Center Connectors サーバーの接続およびメッセージ動作をシミュレートするプロセス。スイッチ・シミュレータにより、現実のスイッチに接続せずに Interaction Center をセットアップできます。

スキル・ベースのルーティング（skill-based routing）

インバウンド・コールを、通話者のニーズを満たす適切なスキルを持ったエージェントに配信する動的コール・ルーティング・インテリジェンス。

スクリーンポップ（screen pop）

顧客の着信電話と同時に Interaction Center エージェントのモニターに表示される、顧客データと製品およびサービス情報のユーザー・インタフェース表現。

静的ルート（static route）

キャッシュされたデータに基づくルート。

ソフトフォン（softphone）

顧客対応エージェントのモニターに表示される電話を機能的な GUI で表したもの。

ターゲット・マシン（target machine）

中間層サーバー・プロセスが実行されるマシン。ノード（node）と同義。

中間層サーバー・プロセス（mid-tier server process）

Oracle Interaction Queueing and Distribution、Oracle Universal Work Queue、Oracle Routing サーバー、Oracle Inbound Telephony サーバーおよび Oracle Telephony メディア・コントローラなど、任意の Interaction Center サーバー。サーバー・プロセス（server process）および Interaction Center サーバー（interaction center server）と同義。

テレフォニ・システム（telephony system）

PBX、ACD、IVR、プレディクティブ・ダイヤラおよび CTI ミドルウェアなど、テレフォニおよび CTI メッセージ機能を提供する任意のハードウェアおよびソフトウェア・コンポーネント。

テレフォニ使用可能（telephony enabled）

インバウンド・コールまたはアウトバウンド・コール、あるいはその両方に関して、電話交換機とユーザーのアプリケーションの間でメッセージを処理する CTI ミドルウェアを介して電話システムと通信するアプリケーションの機能。

テレフォニ・モデル（telephony model）

特定のテレフォニ機能について、予期されるコールの動作を記す使用例。たとえば、あるテレフォニ・モデルでは転送済みコールのコール ID がオリジナル・コールと同じになり、別のテレフォニ・モデルではコンサルタント・コールと同じになります。さらに、第 3 のテレフォニ・モデルでは、オリジナル・コールともコンサルタント・コールとも異なるまったく新しいコール ID になります。

動的ルート（dynamic route）

PL/SQL 問合せに基づくルート。

パッシブ・モード（passive mode）

コールセンター・エージェントに対するコールが標準的な PBX/ACD によりルーティングされ分配されるルーティング・モード。Oracle Advanced Inbound では、コールがエージェントのテレセットで呼び出すと、CTI を介してコールが認識されます。このモードでは、PBX/ACD ルート・ポイントは Oracle Advanced Inbound によりモニターまたは制御されません。

パッケージ（package）

プロシージャ、ファンクション、変数および SQL 文が 1 単位としてグループ化されたもの。

ブラインド転送（blind transfer）

通話者が相手に自分の識別情報を伝えることなく 2 者間で転送されるコール。

マルチ PBX（multi-PBX）

同じサーバーで複数のスイッチおよびミドルウェア構成をサポートする機能。

マルチサイト（multi-site）

複数の物理位置にまたがって連動する Interaction Center。

マルチサイト・キューイングおよび分配（multi-site queuing and distribution）

エージェントのキューを複数サイトにまたがって格納し、保持する単一システム。

マルチサイト・ルーティング（multi-site routing）

複数のサイトにまたがるエージェントにコールをルーティングする機能。

メディア・キュー（media queue）

インバウンド・メディア項目をキューに入れて分配するための Interaction Center コンポーネント。メディア・キューにより、電話や E メールなどのインバウンド項目がキューに格納され、項目を一連のエージェントに送信できるようにルーティング・モジュールと統合されます。メディア・キューにより、キュー内の項目を問い合せたり操作できるように、Oracle Universal Work Queue などの他のモジュールへの API が提供されます。

メディア項目（media item）

電話、E メール、Web コールバックなど、メディアのタイプを表す項目。

メディア・コントローラ（media controller）

他のシステムやソフトウェアを PBX などの基礎となるメディア・ハードウェアとつなぐソフトウェア。

モニター（monitoring）

サーバーのステータスを表示する機能。

ルート・ポイント（route point）

コールのキューイングとルーティングの開始ポイント。自動コール分配機能（automatic call distributor: ACD）。ルート・ポイントは、Alcatel ではパイロットとも呼ばれます。

索引

数字

A

B

C

D

E

F

H

I

J

L

M

N

O

P

R

S

T

U

V

W

あ

い

え

お

か

き

く

け

こ

さ

し

す

そ

ち

て

な

は

ふ

み

め

よ

ら

る

ろ