Oracle® Telephony Adapter SDK

Developers Reference Guide

Release 11*i*

Part No. A97207-05

February 2005

ORACLE®

Oracle Telephony Adapter SDK Developers Reference Guide, Release 11*i*

Part No. A97207-05

Primary Author: Michael Phelan

Contents

Send Us Your Comments

Oracle Telephony Adapter SDK Developers Reference Guide, Release 11*i*

Part No. A97207-05

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appsdoc_us@oracle.com
- FAX: 650-506-7200 Attn: Oracle Supply Chain Management Documentation Manager
- Postal service:
  Oracle Supply Chain Management Documentation Manager
  Oracle Corporation
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Intended Audience

Welcome to Release 11*i* of the *Oracle Telephony Adapter SDK Developers Reference Guide*.

See Related Documents on page x for more Oracle Applications product information.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/ .

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

1 Introduction

2 Concepts

Related Documents

Oracle Interaction Center Server Manager Implementation Guide

Oracle Advanced Inbound Telephony Implementation Guide

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Applications tables are interrelated, any change you make using an Oracle Applications form can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

1 Introduction

This chapter covers the following topics:

- Oracle Interaction Center and Oracle Advanced Inbound
- Overview of Oracle Telephony Adapter SDK
- Advanced Integration
- Standard Integration
- Basic Integration
- New in this Release
- Approaching an SDK Project

Oracle Interaction Center and Oracle Advanced Inbound

Oracle Interaction Center is a family of products that serve as the telephony-enabling foundation of Oracle's eBusiness Suite of applications. Oracle Advanced Inbound is an Oracle Interaction Center product specifically for inbound interaction centers, and which has the following primary functions:

- Agent softphone
- Agent screen pop
- Virtual queuing
- Call routing
- Call and data transfer

Oracle Advanced Inbound consists of a group of server processes, including:

- Inbound Telephony Server, which monitors incoming calls arriving at ACD queues and route points
- Interaction Queuing and Distribution, which maintains a virtual queue of media items for each agent
- Routing Server, which routes and classifies calls
- Oracle Telephony Adapter Server, which normalizes telephony platform-specific messages and events, and provides a single interface between the Oracle Interaction Center and the underlying CTI platform
- Oracle Telephony Manager, which monitors and controls agent extensions

- Oracle Universal Work Queue server, which maintains a virtual work queue for each agent, and provides a single interface between the Oracle Interaction Center and the Oracle E-Business Suite applications

Overview of Oracle Telephony Adapter SDK

Oracle Advanced Inbound features out-of-the-box screen pops of customer data in the Oracle eBusiness Suite, call routing and classifications based on IVR and business data, a feature-rich soft phone and a software development kit (SDK). With the SDK, consultants can make custom integrations to virtually any PBX or CTI middleware that support CTI interfaces or that have the capability to pass data between their telephony client using XML and HTTP.

By way of its ACD/PBX and CTI Middleware Integration Program, Oracle provides Oracle-certified telephony adapters, enabling out-of-the-box integration to a selection of leading PBX/CTI middleware combinations. Integrations using the SDK can be either server based or client based. Server-based integrations offer the richest set of CTI functionality, whereas client-based integrations offer basic login, dialing and application screen pops.

Oracle Telephony Adapter SDK provides a method for consultants and partners to integrate telephony platforms that are not supported directly by Oracle Advanced Inbound. Field teams can use Oracle Telephony Adapter SDK to develop an adapter that plugs into the Telephony Adapter Server and supports CTI integration for the customer’s telephony platform.

The Oracle Telephony Adapter SDK consists of the following components:

- Oracle Telephony Adapter API, the interfaces that the adapter must implement and are called by Oracle Advanced Inbound, and the interfaces that Oracle Advanced Inbound implements and are called by the adapter.
- Oracle Telephony Adapter SDK Programmers Reference Guide, (this document) which explains how to use the API to implement and deploy the adapter, test the adapter implementation, and so on.
- Test Utility for testing and validating the adapter implementation.

Oracle Telephony Adapter SDK has three levels of integration, which are described in the following sections:

- Advanced Integration
- Standard Integration
- Basic Integration

Advanced Integration

Advanced Integration is the full implementation of all the functions and features required by Oracle Advanced Inbound and Oracle Advanced Outbound. Advanced Integration requires implementing all the methods and interfaces specified by the Telephony Adapter SDK. The runtime system requires configuring and running all Interaction Center server processes: Oracle Universal Work Queue, Interaction Queuing and Distribution, Oracle Telephony Manager, Inbound Telephony Server, Oracle Telephony Adapter Server and Oracle Routing Server.

The following table lists the Advanced Integration features.

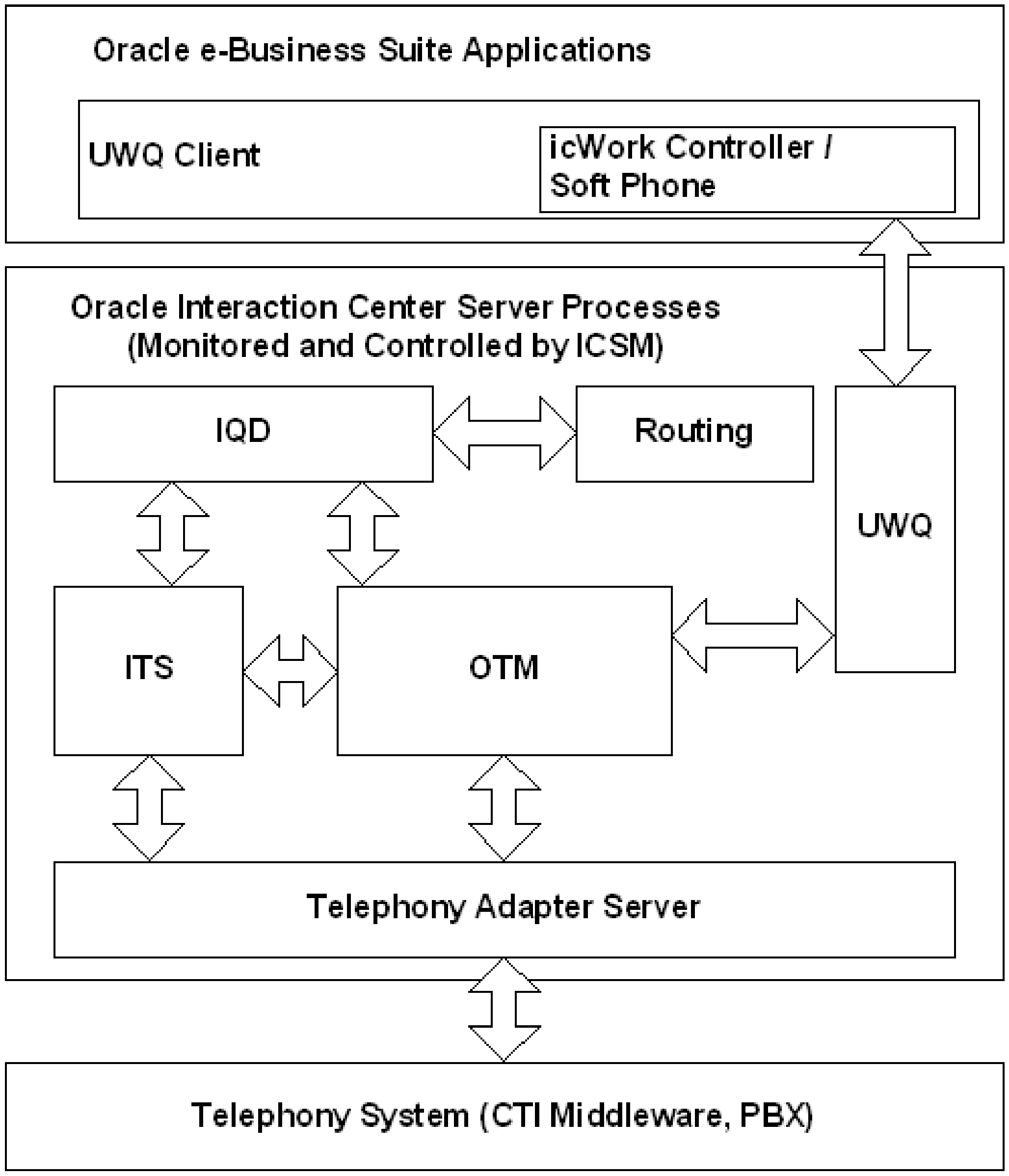
Advanced Integration Features

Feature	Advanced Integration
Summary	• Screen pop • Active/Enhanced Passive/Passive • Full Softphone • IVR Integration • Advanced Outbound
Screen Pop	Screen pop based on data from: • ANI • DNIS • PBX application data • IVR data
CTI Functionality	• Answer Call Release Call Hold / Retrieve • Transfer, Conference • Send DTMF Digits • Route Point Monitoring and Control
Softphone	Full Softphone: • Three Line buttons • Hold button • Release button
Call Routing	Oracle Advanced Inbound does the routing (Active Mode)
Call Classification	Classification based on: • DNIS • IVR Data • PBX application data • IVR data
Outbound Dialing Modes	Choice of Advanced Outbound dialing modes: • Predictive • Progressive • Preview

Feature	Advanced Integration
Interaction History	Full Interaction History: • Life cycle segments (IVR, in queue, with agent) • Abandoned calls
Web Callback	Fully integrated

As the following figure illustrates, in Advanced Integration, the Telephony Adapter Server is the intermediary between the telephony system and Oracle Advanced Inbound. Oracle Universal Work Queue is the intermediary between Oracle Advanced Inbound and the agent's client system, including the icWork Controller and Advanced Integration softphone.

Advanced Integration Architecture

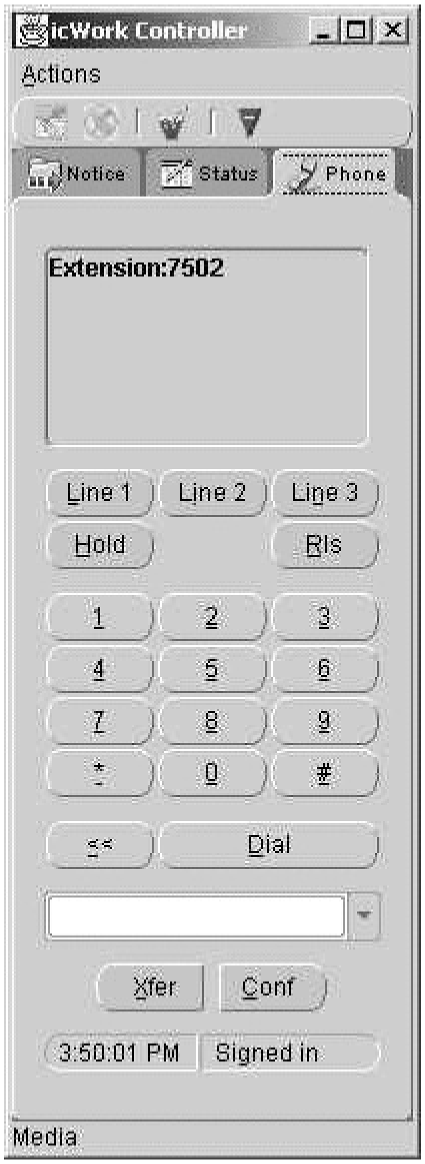

Oracle e-Business Suite Applications
UWQ Client
icWork Controller / Soft Phone
Oracle Interaction Center Server Processes (Monitored and Controlled by ICSM)
IQD
Routing
UWQ
ITS
OTM
Telephony Adapter Server
Telephony System (CTI Middleware, PBX)

ITS = Oracle Inbound Telephony Server
IQD = Interaction Queuing and Distribution
OTM = Oracle Telephony Manager
UWQ = Oracle Universal Work Queue
ICSM = Interaction Center Server Manager

Advanced Integration Softphone

The following illustration shows the softphone with full features, displaying buttons for three lines, hold, release, dial, transfer, conference and numbered dialing buttons.

Advanced Integration Softphone



Standard Integration

Standard Integration requires implementing a subset of the methods and interfaces that are specified by the Telephony Adapter SDK. The runtime system requires configuring and running all Interaction Center server processes required for Oracle Advanced Inbound *except* for Inbound Telephony Server and Interaction Queuing and Distribution. Oracle Routing Server does classifications.

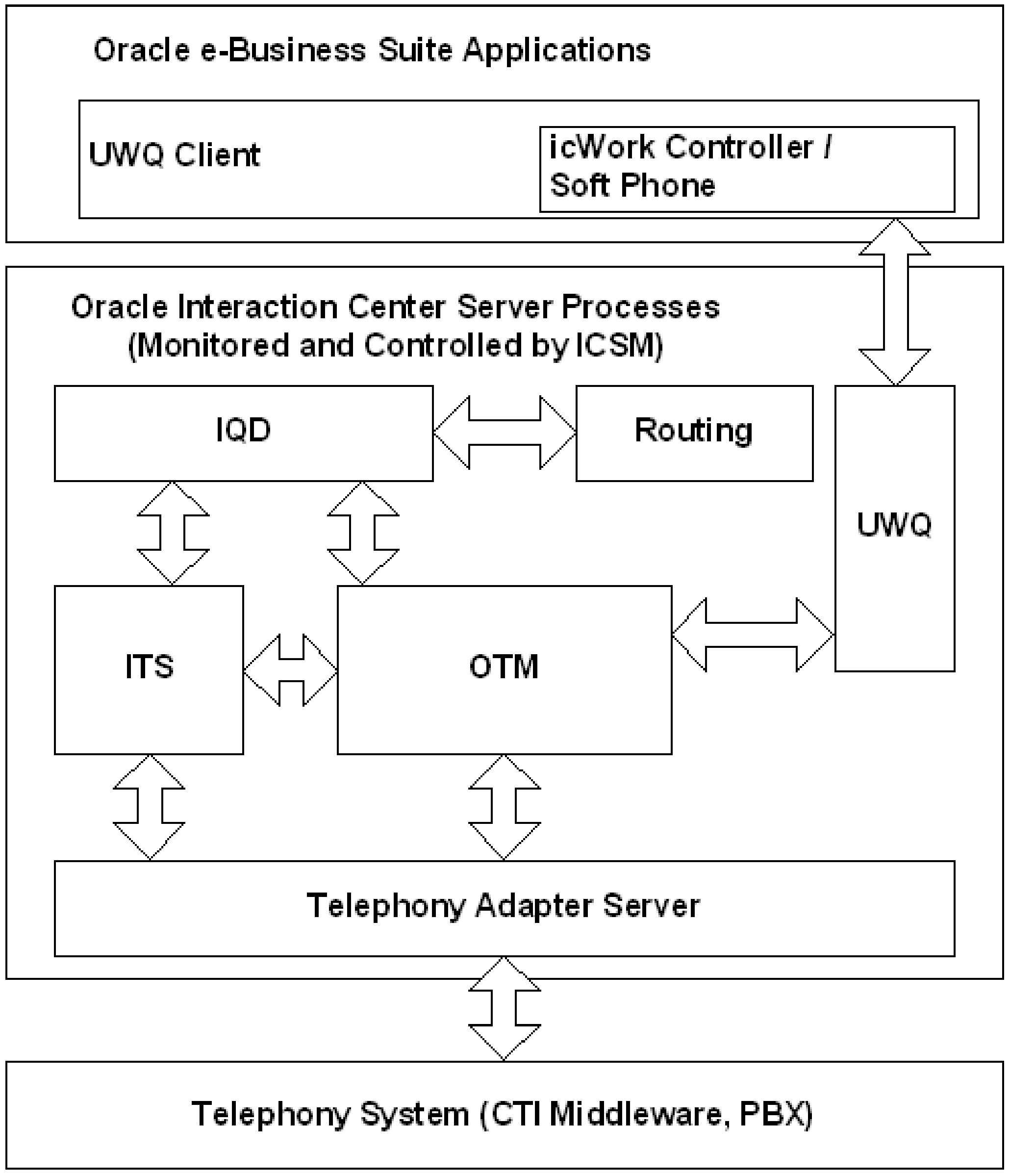
The following table lists and describes the features of Standard Integration.

Standard Integration Features

Feature	Standard Integration
Summary	• Screen pop • Softphone Lite • Passive (ACD routing)
Screen pop	Screen pop based on: • PBX Application Data • ANI • DNIS
CTI Functionality	• Transfer • Conference Use the physical telephone to do all other telephony functions.
Softphone	Softphone Lite: • Transfer button • Conference button
Call Routing	ACD does the routing (Passive Mode)
Call Classification	Classification based on: • ANI • DNIS • PBX Application Data
Outbound Dialing Modes	• Progressive • Preview
Interaction History	Moderate Interaction History: Media LCS for Trn/Conf
Web Callback	Optional

As the following figure illustrates, in Standard Integration the Telephony Adapter Server is the intermediary between the telephony system and Oracle Advanced Inbound. Oracle Universal Work Queue is the intermediary between Oracle Advanced Inbound and the agent's client system, including the icWork Controller and the softphone.

Standard Integration Architecture

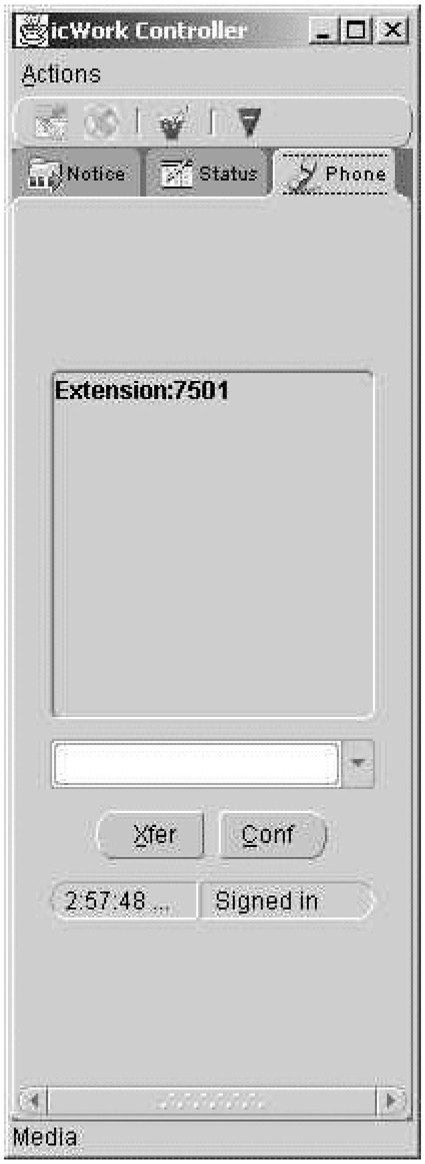

Oracle e-Business Suite Applications
UWQ Client
icWork Controller / Soft Phone
Oracle Interaction Center Server Processes (Monitored and Controlled by ICSM)
IQD
Routing
UWQ
ITS
OTM
Telephony Adapter Server
Telephony System (CTI Middleware, PBX)

ITS =	Oracle Inbound Telephony Server
IQD =	Interaction Queuing and Distribution
OTM =	Oracle Telephony Manager
UWQ =	Oracle Universal Work Queue
ICSM =	Interaction Center Server Manager

Standard Integration Softphone

The following illustration shows the icWorkController softphone with lightweight functions, displaying the extension number and buttons for transfer and conference.

Standard Integration Softphone



Extension:7501

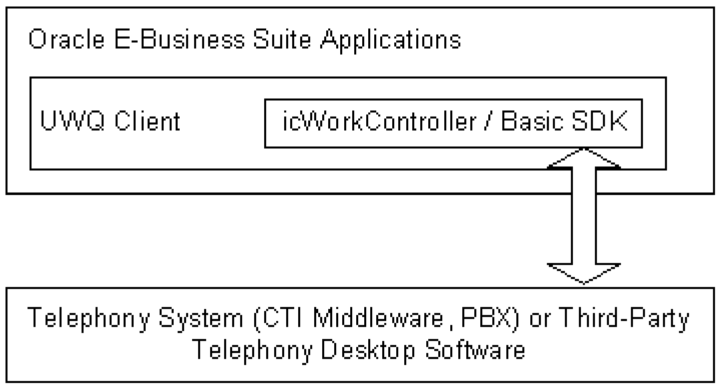
2:57:48 ... Signed in

Basic Integration

Basic Integration is the lightest level of integration to Oracle Advanced Inbound. The API and the underlying infrastructure for Basic Integration are different from Standard Integration and Advanced Integration.

As the following figure illustrates, unlike the Standard and Advanced Integrations, Basic Integration does not require any Advanced Inbound or Universal Work Queue server processes. Oracle e-Business Applications and the Telephony System communicate through an Oracle Universal Work Queue Client using HTTP and XML.

Basic Integration Architecture


Oracle E-Business Suite Applications
UWQ Client
icWorkController / Basic SDK
Telephony System (CTI Middleware, PBX) or Third-Party Telephony Desktop Software

The following table lists and describes the features of Basic Integration.

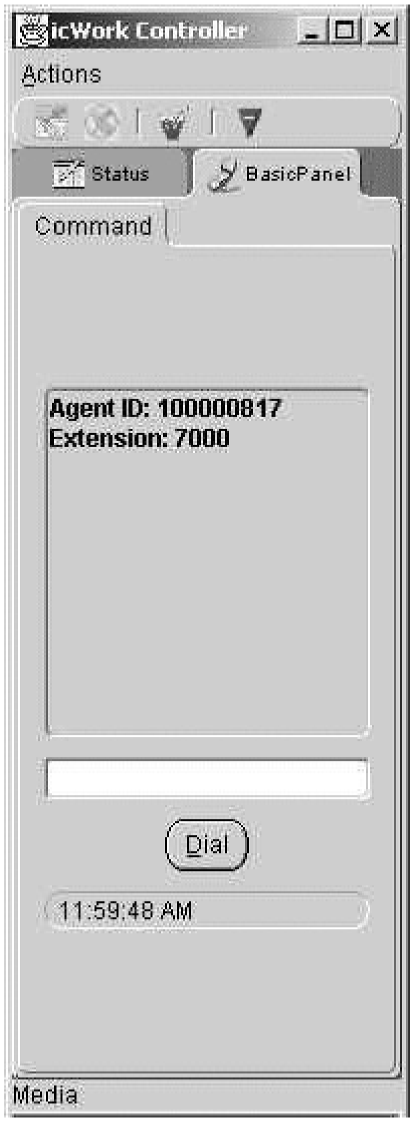
Basic Integration Features

Features	Basic Integration
Screen pop	Screen pops based on: • ANI • DNIS • Other interaction keys. See Additional Interaction Keys.
CTI Functionality	• Login / Logout • Ready / Not Ready • Wrap Up • Make Call Use the physical telephone to do all other telephony functions.
Softphone	Basic Panel
Call Routing	ACD does the routing (Passive Mode)
Call Classification	(Optional) By way of third-party field passed to Basic client
Outbound Dialing Modes	• Manual. • Preview mode is supported in implementations in which the option for Advanced Outbound in preview mode is elected. Support is by way of the Oracle Universal Work Queue server and the Interaction Center Server Manager. With Basic Integration, Advanced Outbound in preview mode is the only dialing mode available. Starting the Universal Work Queue server for preview mode does not result in progressive or predictive mode.
Interaction History	Limited Interaction History: • Number of calls per site • Number of calls per agent • Talk time
Basic Web Callback	• Preview Dialing • Timed Preview Dialing

Basic Integration Softphone

The Basic Integration softphone consists of the Basic Panel, a tab on icWork Controller that agents can use to make calls, view the current telephony event, and optionally configure to display detailed events and logs. Basic Panel is shown in the following illustration.

Basic Integration Softphone



Agent ID: 100000817
Extension: 7000

11:59:48 AM

Basic Web Callback

Basic Integration can be extended with support for Basic Web Callback, which enables interaction center agents to service Web callback requests that are made through Oracle *i*Store or Oracle *i*Support. Web callback requests are queued in first in first out (FIFO) order and can be routed to any eligible agent. When an eligible agent does an Oracle Universal Work Queue function Get Work for Basic Web Callback, a Web callback request is assigned from the queue to the agent, who receives a screen pop for the callback.

In Basic Web Callback, all requests are routed to all agents. Because Oracle Routing Server is not involved in Basic Integration, you cannot set up route rules for actively routing Basic Web Callback requests.

New in this Release

This release of Oracle Telephony Adapter SDK does not include any new features or functionality.

Approaching an SDK Project

Use the following outline in planning an SDK project.

1. Qualify a project for SDK implementation. See SDK Qualification Worksheet.
2. Scope the extent of the project and brief the programmer. See SDK Scope Analysis.
3. Develop the Telephony Adapter. See Developing the Telephony Adapter for the C and Javadoc APIs.
4. Unit test the adapter and perform end-to-end integration testing. See Telephony Adapter Test Cases and Testing the Telephony Adapter.
5. Analyst review of the deliverables and sign off.
6. Deploy the Telephony Adapter, see Deploying the Telephony Adapter.

2 Concepts

This section explains key concepts of Oracle Telephony Adapter SDK.

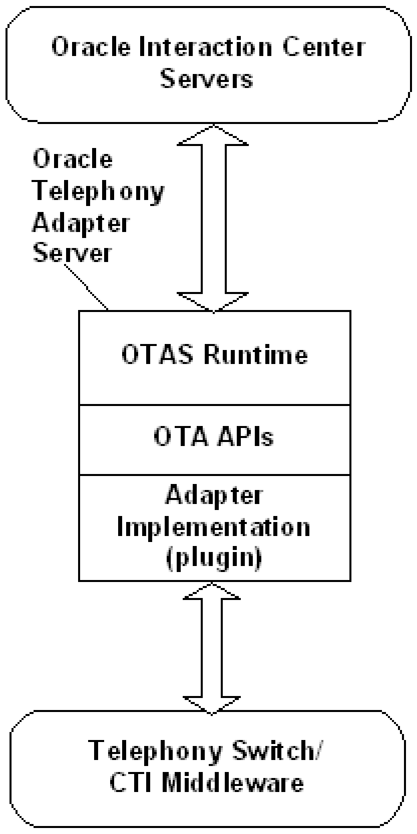
This chapter covers the following topics:

- Architecture
- Adapter Implementation
- Oracle Telephony Adapter API Objects
- TeleDeviceFactory
- TelesetDevice
- RoutePointDevice
- Call Data
- Media Item ID
- Media Item ID Model

Architecture

The Oracle Telephony Adapter Server consists of a communications core (Oracle Telephony Adapter Server Runtime), the Oracle Telephony Adapter APIs, and the adapter code that implements and writes to the Oracle Telephony Adapter APIs.

The following figure illustrates Oracle Telephony Adapter Server Architecture.


Oracle Interaction Center Servers
Oracle Telephony Adapter Server
OTAS Runtime
OTA APIs
Adapter Implementation (plugin)
Telephony Switch/ CTI Middleware

As the previous figure illustrates, the Oracle Telephony Adapter Server consists of Oracle Telephony Adapter Server Runtime, Oracle Telephony Adapter Server APIs, and the adapter implementation (plug-in). The Oracle Telephony Adapter Server connects the telephony switch and CTI middleware with Oracle Interaction Center.

Adapter Implementation

The adapter implementation is platform-specific custom code that maps telephony platform (ACD/PBX and CTI middleware) APIs to Oracle Telephony Adapter APIs.

Oracle Telephony Adapter API Objects

Oracle Telephony Adapter APIs specify three objects that the adapter code must implement and through which it must communicate. The objects are TeleDeviceFactory, TelesetDevice and RoutePointDevice.

TeleDeviceFactory

Oracle uses the TeleDeviceFactory object to allocate and deallocate switch resources. In the current release, only the Teleset and Route Point resources can be allocated.

TelesetDevice

A teleset device represents a physical telephone (currently only digital telesets, not analog telesets) that is used by an agent. Agents log into their telesets and use this object to perform various call control functions, such as placing a call, putting a call on hold and hanging up a call. The teleset device also relays switch events signifying state changes and call progression from the telephony platform to Oracle Interaction Center.

For Advanced Integration, the teleset device has the following characteristics:

- Each teleset is displayed as three lines on the Oracle softphone.
- Each line can hold one call.
- Depending on the telephony platform, one key on the physical teleset maps to one or more lines on the Oracle softphone. For example, for Avaya Definity one physical teleset is programmed with three keys that share the same "station number." In this case, on the Oracle softphone one line represents one key of the Avaya physical teleset. For Nortel Meridian, one physical teleset is programmed with two keys, Personal DN and ACD DN. In this case, on the softphone, line one represents the Personal DN key, line two represents the ACD DN key, and line three represents a consultation call placed from either key.
- Most events that are fired to the Oracle Interaction Center must specify the line on which the event occurs (line indexes 0, 1 or 2).
- Most call control commands are performed on the line that is specified by the API call, such as holdCall. The API calls makeCall and consultationCall can return events on a different line.
- Only one consultation call can be placed at a time. Therefore, only one transfer or conference can be pending.

RoutePointDevice

A route point device represents a switch queue that can be either a controlled queue, in which Oracle Advanced Inbound actively routes calls, or a monitored queue, in which Oracle Advanced Inbound only keeps track of the queued calls while the switch does the actual call routing.

The route point device has the following characteristics:

- Each route point has a queue of pending calls.
- Each call in the queue is identified by a call identifier, usually the call ID that is assigned by the telephony platform.
- The call identifier is used by call control commands. Any events that are fired must contain the call identifier.
- If the call ID changes due to a blind transfer, then the corresponding call identifier should not change.

Call Data

Call data is any data that is associated with a call, specifically the ANI, DNIS and any IVR data. This data is used by Oracle Interaction Center to properly classify and route the call and is also the basis for a screen pop. The adapter expects call data from the telephony platform at the start of a call or soon after the start.

Media Item ID

Oracle Interaction Center keeps track of logical interactions that are identified by a Media Item Identifier (Media Item ID). Although a switch-specific call ID may change as the call is transferred, the Media Item ID must remain the same. When the Media Item ID is assigned by Oracle, it is the responsibility of the implementer to ensure that it is propagated as the call moves from agent to agent. If the call ID on a line changes due to a transfer or conference completion, then the corresponding Media Item ID does not change.

Media Item IDs are assigned by Oracle and passed to the Adapter by way of the TelesetDevice and RoutePointDevice methods. Maintaining the correct Media Item ID in call events is essential to ensure that transfer and conference scenarios are processed correctly.

Media Item ID Model

The Media Item ID model has the following characteristics.

Note: The following examples do not show all events.

Oracle Interaction Center Actions

Oracle Interaction Center Actions	Adapter Server Actions/ States	Oracle Interaction Center States
A: makeCall (MI: 1, Dest: B)4	A: dial (Dest: B) map resultant call id 1 to MI 11	
	A: beginCallEvent (MI: 1, line: 0)2	A: line 0 has MI 11
	B: beginCallEvent (MI: 1, line: 0)2	B: line 0 has MI 11
A: releaseCall (line: 0)	A: hangup (call id 1)	
	A: callReleasedEvent (line: 0)	A: line 0 is clear
	B: callReleasedEvent (line: 0)	B: line 0 is clear

Customer Actions

Customer Actions	Adapter Server Actions/ States	Oracle Interaction Center States
X: calls A	A: beginCallEvent (MI: n/ a, line:0) Oracle Telephony Adapter Server does not know MI1	A: line 0 has MI 24: OIC assigns Media Item ID to call
X: hangup call	A: callReleasedEvent (line:0)	A: line 0 is clear

Customer/ Oracle Interaction Center Actions

Customer/OIC Actions	Adapter Server Actions/ States	Oracle Interaction Center States
X: calls A	A: beginCallEvent (MI: n/ a, line:0) // call id 2, Oracle Telephony Adapter Server does not know MI1	A: line 0 has MI 33: OIC assigns Media Item ID to call
A: consult (MI1: 3, line: 0, Dest: B)4	A: consult (call id: 2, Dest: B) // map call id 2 to MI1 3 // map consult call id 3 to MI 3	
	A: beginCallEvent (MI: 3, line: 1)2	A: line 1 has MI 3
	B: beginCallEvent (MI: 3, line: 0)2	B: line 0 has MI 3
A: transfer (line: 1, old line: 0)	A: transfer (call ID 3, call ID 2)	
	A: callReleasedEvent (line: 0)	A: line 0 is clear
	A: callReleasedEvent (line: 1)	A: line 1 is clear
	B call ID changes (ABA model)5	No event to OIC

Footnotes

1. Media Item ID2. If an incoming call already has a Media Item ID, then it must be passed to the receiving device.3. If an incoming call does not have a Media Item ID, then Oracle Interaction Center assigns a new Media Item ID and uses it with any future consultation calls.4. New calls are assigned a new Media Item ID when the dial command is made.5. There is a one-to-many relationship between Media Item ID and switch-specific call IDs. If the call ID changes while the actual call is transferred or conferenced, then the Media Item ID should remain the same.

3 Development Environment

This chapter covers the following topics:

- Minimum Hardware Requirements
- Minimum Software Requirements

Minimum Hardware Requirements

Oracle Telephony Adapter SDK has the following hardware requirements.

Development Machine

- CPU: 500MHz
- Memory (RAM): 256MB
- Disk space: 10G
- Operating Systems:
  - C API: Windows NT 4.0 SP6 or Windows NT 2000
  - Java API: Windows NT 4.0 SP6 or Windows NT 2000, Linux and UNIX
- Plus any required hardware for telephony connectivity

Telephony Hardware

- PBX
- Telesets

Minimum Software Requirements

Oracle Telephony Adapter SDK has the following software requirements.

General

- Java Development Kit version 1.1.8, 1.2 or 1.3
  - (Optional) An Integrated Development Environment (IDE) for Java such as Oracle JDeveloper

SDK for Java

- Any third-party CTI Java libraries

SDK for C

- Any third-party CTI software for Windows NT and Windows 2000
    - C compiler and make utility
    - (Optional) An Integrated Development Environment (IDE) for Windows such as Visual C++

4 Oracle Telephony Adapter SDK Components

This chapter covers the following topics:

- Installing Oracle Telephony Adapter SDK
- Package Contents
- SDK for Java
- SDK for C
- Oracle Telephony SDK Integrated Test Utility

Installing Oracle Telephony Adapter SDK

Oracle Telephony Adapter SDK provides APIs for the C and Java programming languages.

Oracle Telephony Adapter Java APIs provide an open interface for consultants and switch and CTI middleware vendors to implement an integration to Oracle Interaction Center (OIC) for any telephony platforms using the Java programming language.

Oracle Telephony Adapter C APIs enable programmers to integrate switches and CTI middlewares using the C programming language.

The Oracle Telephony Adapter C API consists of the following APIs and functions:

- Header files for TeleDeviceFactory (TeleDeviceFactoryAPI.h), TelesetDevice (TelesetDeviceAPI.h) and RoutePointDevice (RoutePointDeviceAPI.h).
- Event Listener API (TelesetEventListenerAPI.h and RouteEventListenerAPI.h)
- Common utilities functions, typedefs and constants (occt_pub.h , occt_md.h)
- OHashtable functions (Hashtable.h)

The C adapter implementer implements all methods that are defined in the Header files.

Use the following instructions to install the Oracle Telephony Adapter SDK.

Prerequisite

Java Development Kit or Java Runtime Environment must be installed before you install Oracle Telephony Adapter SDK. Oracle recommends that you use JDK version 1.3.

Downloading Oracle Telephony Adapter SDK

The "Patch for Telephony Adapter Software Development Kit (SDK)" is available in Oracle*MetaLink* as Patch 2621791.

Use the following steps to download Oracle*MetaLink* Patch 2621791.

1. In the side navigation bar, select **Patches**.

   The Select a Patch Search Area page appears.

2. Click **New MetaLink Patch Search**.

   The Simple Search page appears.

3. Enter 2621791 in the blank text field.
4. Click **Go**.
5. The Patch 2621791 page appears.
6. Click **Download**.

   The download process begins.

> **Note:** Before you extract the downloaded Zip file, in the WinZip Extract window make sure to check "Use folder names" (or a similar option) so that the files extract into the specified directory structure. See the following section "Unpacking."

Unpacking

Use a file extraction program to unzip p2621791_11i_WINNT.zip to a directory path that does not contain white spaces. After you unzip p2621791_11i_WINNT.zip, edit the file oracle/apps/cct/bin/sdkenv.cmd and modify the value of the SDK_JRE_HOME environment variable to the location where JDK is installed. The directory path where the JDK is installed should not contain blank spaces.

Package Contents

Oracle Telephony Adapter SDK package p2621791_11i_WINNT.zip contains the following files.

Contents of Zip File

Directory	Description
bin	Scripts for starting Adapter Server and Test Tools
lib	Java and C runtime libraries supplied by Oracle
docs	Documentation
c	SDK for C
c/include	SDK for C: C include files
c/lib	SDK for C: C libraries supplied by Oracle for C adapter development
c/impl	SDK for C: Directory for storing C adapter implementation
c/sample	SDK for C: Sample C code
java	SDK for Java
java/javadoc	SDK for Java: Javadoc
java/classes	SDK for Java: Directory for storing Java adapter implementation class files
java/sample	SDK for Java: Sample code
3rdParty	Storage for Java and C libraries provided by switch and CTI middleware vendor
c/ao/include	AO SDK Extension: C include files
c/ao/lib	AO SDK Extension: C libraries supplied by Oracle for AO SDK Extension
c/ao/sample	AO SDK Extension: Sample AO All C Implementation
java/sample/advanced	SDK for Java: Sample Code for Advanced Integration
java/sample/ao	AO SDK Extension: Sample AO All Java Implementation
java/sample/basic	SDK for Java: Sample Code for Basic Integration
java/sample/ standard	SDK for Java: Sample Code for Standard Integration

SDK for Java

Developers can use the Oracle Telephony Adapter SDK for Java to implement the adapter by using the Java programming language. The SDK for Java should be used when the Java API is available from the target switch and CTI middleware to be integrated, such as CT Connect and JTAPI.

The adapter implemented using SDK for Java is packaged as JAR files.

SDK for C

Developers can use the Oracle Telephony Adapter SDK for the C programming language to implement the adapter on only the Windows NT and Windows 2000 platforms. The SDK for C should be used when the C API is the only available option from the switch and middleware to be integrated, such as Cisco ICM and TAPI. SDK for Windows NT and Windows 2000 is a set of C functions implementing integrations to the switch and CTI middleware. These functions are collectively packaged as an adapter.

The adapter implemented using SDK for Windows NT and Windows 2000 is packaged as a Windows DLL (Dynamically Loaded Library) file.

Oracle Telephony SDK Integrated Test Utility

Oracle Telephony SDK Integrated Test Utility is a user interface for running, configuring and testing telephony adapters. Developers can use this utility to test their adapter implementations in offline mode, which does not require an Oracle Application Database or Oracle Interaction Center servers. However, the Test Utility does require that the telephony adapter be connected to the telephony platform. The Test Utility is primarily used in the first phase of adapter development where developers can work on their own development workstations without connecting to an Oracle Application Database. The integrated test utility includes four components: Oracle Telephony Adapter Server, TelesetDevice Test Utility, Telephony Adapter Verification Tool, and the switch simulator.

Oracle Telephony Adapter Server

The Test Utility contains the Oracle Telephony Adapter Server. Developers can use Oracle Telephony Adapter Server to run and test adapter implementations.

- TeleDevice Test Utility: TeleDevice Test Utility provides a user interface for testing TelesetDevice and RoutePointDevice objects. Developers can use this utility to launch the softphone and to drive test cases.
- Telephony Adapter Verification Tool: Telephony Adapter Verification Tool provides developers with a means to verify their adapter implementation against a predefined set of telephony test cases.
- Switch Simulator: Developers can use the switch simulator to practice developing an adaptor without having to use a physical telephony switch.

5
Testing the Telephony Adapter

This chapter covers the following topics:

- Starting and Stopping the Test Utility
- Using Log Windows
- Oracle Telephony Adapter Server
- Configuring the Oracle Telephony Adapter Server Log
- Configuring Middleware
- Starting and Stopping Oracle Telephony Adapter Server
- Verification Tool
- Configuring the Verification Process
- Configuring Agent Information
- Starting and Stopping the Verification Tool
- TeleDevice Test Utility
- TeleDevices
- Assigning TeleDevices
- Connecting and Disconnecting a TeleDevice
- Deassigning a TeleDevice
- Viewing Events
- Invoking API Methods
- Disconnecting a TeleDevice
- Launching the Softphone
- Closing the Softphone
- Switch Simulator
- Starting and Stopping the Switch Simulator
- Configuring Switch Simulator Server Settings
- Configuring Simulated Extensions and Route Points
- Configuring the Switch Simulator for Standard or Advanced Integration
- Configuring the Switch Simulator for Basic Integration

- Javadoc

Starting and Stopping the Test Utility

The Oracle Telephony SDK Integrated Test Utility is a tool for testing the Telephony Adapter implementation, which is loaded and run by the Oracle Telephony Adapter Server. The Telephony SDK Integrated Test Utility contains user interfaces for running the Oracle Telephony Adapter Server, the Verification Tool, the TeleDevice Test Utility, Switch Simulator, and the Javadoc. Developers can use the Oracle Telephony SDK Integrated Test Utility to test the full functions of the TeleDevice implementations.

The Oracle Telephony SDK Integrated Test Utility interface has five tabs:

- Oracle Telephony Adapter Server, for configuring and running the Oracle Telephony Adapter Server
- Verification Tool, for configuring and running the TeleDevices
- TeleDevice Test Utility, for configuring and running the TeleDevice and softphone
- Switch Simulator, for configuring and running the switch simulator
- Help, which displays the Javadoc

Use the following procedures to start and stop the Oracle Telephony SDK Integrated Test Utility.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

1. Obtain the Oracle Telephony SDK Zip file.
2. Update the SDK_JRE_HOME environment variable in the file sdkenv.cmd to point to a JDK installation, such as D:/jdk1.2.2.

Steps:

1. Navigate to the directory oracle/apps/cct/bin.
2. Run the file sdkrun.cmd.

   The Oracle Telephony Adapter SDK Integrated Test Utility appears, opened to the Oracle Telephony Adapter Server tab > Server sub tab.
3. To stop the Oracle Telephony SDK Integrated Test Utility, close the Oracle Telephony Adapter SDK Integrated Test Utility window.

Using Log Windows

By default each main tab (excluding Help) of the Oracle SDK Integrated Test Utility includes a Log window that shows the progress and completion of configurations. To view the log messages in a separate, individual window, right click in the Log window and select **Undock**. The Log window detaches as a smaller, separate window that contains the log messages.

To close the separate log window and restore the log file to the main Log window, right click in the separate window and select **Dock**.

To clear log messages, right click in the selected log window and select **Clear**.

Oracle Telephony Adapter Server

The Oracle Telephony SDK Integrated Test Utility's Oracle Telephony Adapter Server user interface consists of the Server tab, to stop Oracle Telephony Adapter Server and to configure and view logs, and the Middleware tab, and configure CTI middleware.

> **Note:** Oracle Telephony Adapter Server cannot be started in the Middleware tab.

Configuring the Oracle Telephony Adapter Server Log

Use the following procedure to configure the Oracle Telephony Adapter Server log.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Install and configure Oracle Telephony Adapter SDK.

Steps

1. Select the Oracle Telephony Adapter Server tab > Server sub tab.
2. From the Log Level list, select one of the four log levels:
   1. error, to print only error messages
   2. warning, to print error and warning messages
   3. info, to print error, warning and information messages
   4. verbose, to print all messages
3. In the Log Directory field, enter the directory path of the log file. The log file created by Oracle Telephony Adapter Server is named using the pattern OTASyyyymmdd_hhmmss.log. The default directory is oracle/apps/cct/bin/log.
4. Select **File > Save** to save the configuration.

Configuring Middleware

In the Oracle Telephony SDK Integrated Test Utility, the Oracle Telephony Adapter Server middleware configuration consists of three types of data:

- Data that Oracle Telephony Adapter Server requires to load the correct type of adapter, which includes the Middleware Type, Library Name and Factory Class Name.
- Custom data that the adapter requires to be connected to the third-party CTI middleware systems, which includes IP addresses, ports and custom attributes. This data is passed to the TeleDeviceFactory.init() method.
- Dialing data in the Dialing section that Oracle Telephony Manager requires to generate the local dial string based on the location from which the call is placed. Typically, developers are not required to use the data in the Dialing section.

Use the following procedure to configure middleware.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Install and configure Oracle Telephony Adapter SDK.

Steps

1. In the Oracle Telephony SDK Integrated Test Utility, select the Oracle Telephony Adapter Server tab > Middleware sub tab.

   The Middleware page appears.
2. Do one of the following:

   For an adapter developed in Java:
   1. From the Middleware Type list, choose **Java Adapter**.
      1. In the Factory Class Name field, enter the TeleDeviceFactory class name.

         For an adapter developed in C:
      2. From the Middleware Type list, choose **C Adapter.**
      3. In the Library Name field, enter the DLL library name.
3. All other fields are passed to the TeleDeviceFactory.init() method as key-value pairs stored in a Hashtable. The hashtable keys used for each field are defined in interface Constants (for Java) and occt_pub.h (for C) and are listed in the following table.

Oracle Telephony Adapter Server Field and Hashtable Keys

Field	Hashtable Key (Java)
Passive mode	KEY_PASSIVE_MODE
CTI Server IP Address 1	KEY_CTI_SERVER_IP_1
CTI Server IP Address 2	KEY_CTI_SERVER_IP_2
CTI Server Port 1	KEY_CTI_SERVER_PORT_1
CTI Server Port 2	KEY_CTI_SERVER_PORT_2
Adapter Server Info 1	KEY_ADAPTER_SERVER_INFO_1
Adapter Server Info 2	KEY_ADAPTER_SERVER_INFO_2
Adapter Server Info 3	KEY_ADAPTER_SERVER_INFO_3
Adapter Server Info 4	KEY_ADAPTER_SERVER_INFO_4
Adapter Server Info 5	KEY_ADAPTER_SERVER_INFO_5
Adapter Server Info 6	KEY_ADAPTER_SERVER_INFO_6
Domestic Dialing Prefix	KEY_DOMESTIC_PREFIX
International Dialing Prefix	KEY_IDD_PREFIX
Outgoing Prefix	KEY_OUTGOING_PREFIX
Site Overlay	KEY_SITE_OVERLAY
Site Area Code	KEY_SITE_AREA_CODE
Site Country Code	KEY_SITE_COUNTRY_CODE
Site Internal Number Length	KEY_SITE_INTERNAL_NUM_LENGTH
Site Local Number Maximum Length	KEY_SITE_LOCAL_NUM_MAX_LENGTH

4. Select **File > Save** to save the configuration.

Starting and Stopping Oracle Telephony Adapter Server

Use the following procedures to start and stop the Oracle Telephony Adapter Server of the Oracle Telephony SDK Integrated Test Utility.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

1. Install and configure Oracle Telephony SDK.
2. Configure the CTI middleware.

Steps

To start Oracle Telephony Adapter Server, use the following procedure.

1. In the Oracle Telephony Adapter Server window select the Server tab.
2. Click **Start**.

   Oracle Telephony Adapter Server launches as a background Java process and uses the current middleware configuration to start. A log file appears in the Oracle Telephony Adapter Server Log window.

   > **Note:** Any configuration changes made while Oracle Telephony Adapter Server is running will not take effect until Oracle Telephony Adapter Server is stopped and restarted.

To stop Oracle Telephony Adapter Server, use the following procedure.

1. In the Oracle Telephony Adapter Server window select the Server tab.
2. Click **Stop**.

   The Oracle Telephony Adapter Server background process terminates. The log file remains open in the Oracle Telephony Adapter Server Log window.

Verification Tool

The Verification Tool automates testing the adapter implementation by performing the following functions:

- Runs a series of common test cases and validates an implementation
- Reads a test case file written in proprietary Oracle internal syntax
- Executes the series of steps in the test case sequentially
- Waits and collects the events after each step in the test case
- Verifies the collected set of events with the expected set of events (from the solutions file) for every test step
- Prints the result of the verifications in the Results file and in the log window

Configuring the Verification Process

Use the following procedure to configure the settings in the Verification Tool Control tab, including starting and stopping the verification process, and configuring input and output files and log levels.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

Implement the telephony adapter.

Steps

1. Select the Verification Tool tab > Control sub tab.

   The Control page appears.

2. In the Solutions File field, enter or browse for the default solution file Solutions.txt. The path is oracle/apps/cct/scripts directory. To use a different solutions file, enter the full path and file name of the file in the Solutions File box. The solutions file contains the expected events and event details for every SDK API invocation and compares the collected events at runtime with the expected events.
3. In the Test Cases File field, enter or browse for the default test case file TestCases.txt in the directory oracle/apps/cct/scripts. The Test Cases file contains the listing of all the test cases that the Verification Tool will execute. Each line in this file points to a specific script file. Each script file name should be either the absolute file name or relative to the bin directory of the installation. Optionally, to use a different test case file, enter the full path and file name of the file in the Test Cases File field. You can edit the TestCases.txt file to modify the order of Script files to execute, and add or drop a script file to execute.
4. In the Results File field, enter or browse for the default results file output.txt. The Results File is the output file to which the Verification Tool writes results. Optionally, to use a different results file, enter the full path and file name of the file in the Results File field.
5. From the Log Level list, select one of the four levels of logging sensitivity.
   1. Error: Error messages only
   2. Warning: Error and warning messages
   3. Info: Error, warning and informational messages
   4. Verbose: All messages
6. From the Event Delay Threshold list select the number of seconds that the Verification Tool must wait between executing every two steps in a script. Set the value of the EventDelay Threshold to correspond to greater than or equal to the time delay between command execution and event reception for your telephony platform.
7. Select **File > Save** to save the configuration.

Configuring Agent Information

Use the following procedure to configure the Verification Tool settings in the Configure tab, including the agent and teleset and route point numbers that are required to run the tool.

Prerequisites

- Implement the telephony adapter.
- Configure the Verification Tool Control tab settings.

Steps

1. Select the Verification Tool tab > Configure sub tab.

   The Configure page appears.
2. The Verification Tool uses three telesets (Agent A, Agent B and Agent C) for most of the test cases. In the Line*n* Extn# fields, enter the line extension number of each teleset.
3. For each teleset, enter the AgentLoginId, AgentPassword, and AgentQueue. The Verification Tool uses these configuration values to log in to each teleset before executing the scripts.
4. Some test cases involve outbound calls to external numbers and inbound calls from external numbers. In the External Numbers fields, enter external telephone numbers which the Verification Tool can use to make calls and answer calls to verify the ANI and DNIS values for the events collected in those test cases.
5. Some test cases involve making calls to Busy Numbers and Invalid Numbers. In the Busy & Invalid Numbers fields enter the telephone numbers that the Verification Tool can use to verify the ANI and DNIS values for the events collected in those test cases.
6. The Verification Tool uses two route points, one active and one passive, for testing the RoutePointDevice implementations. In the Active RP # field, enter the extension number of an active route point. In the Passive RP # field, enter the extension number of a passive route point.
7. Select **File > Save** to save the configuration.

Starting and Stopping the Verification Tool

To start the Verification Tool, use the following procedure.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Install the Verification Tool.

Steps

1. Select the Verification Tool tab > Control sub tab.
2. Click **Start**.

   The Verification Tool uses the current configuration and starts as a background process. The log file appears in the Verification Tool Log window.

   > **Note:** Any changes you make while the Verification Tool is running will not take effect until you stop and restart the Verification Tool.

To stop the Verification Tool, use the following procedure.

1. Select the Verification Tool tab > Control sub tab.
2. Click **Stop**.

   The background process terminates.

TeleDevice Test Utility

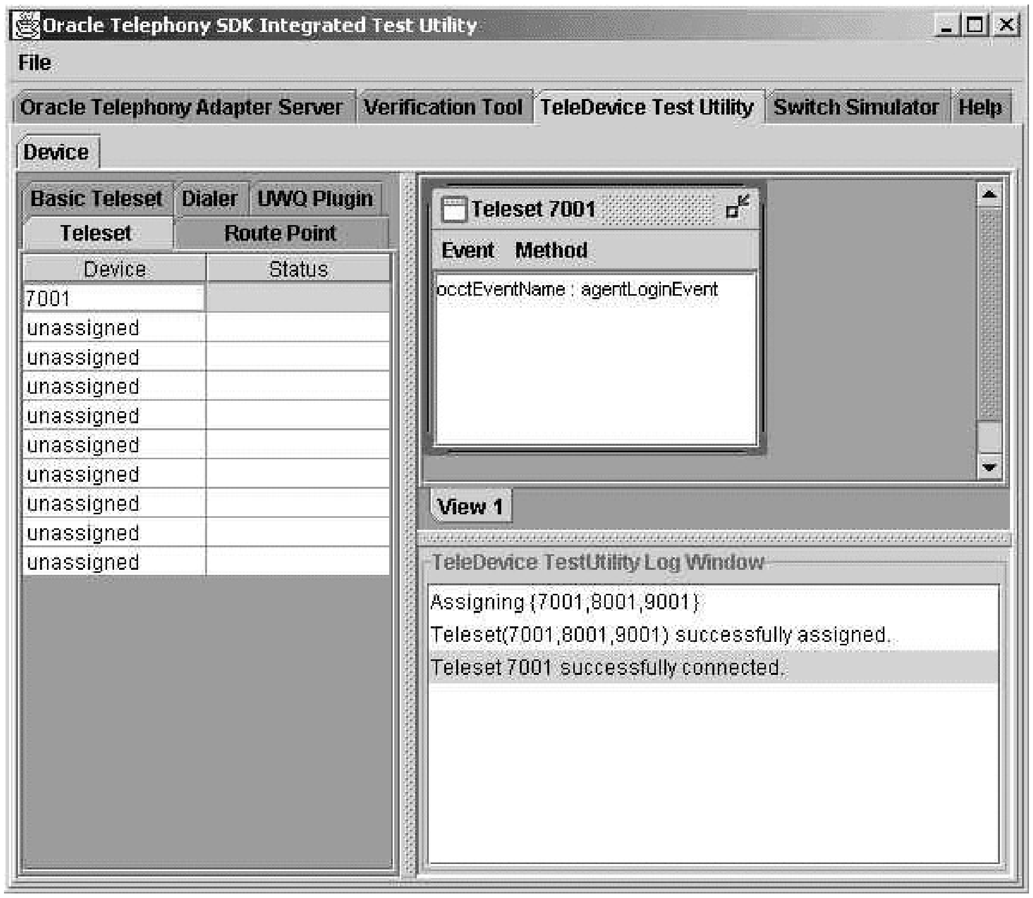
TeleDevice Test Utility is an interface for controlling and monitoring TeleDevices. Developers can use TeleDevice Test Utility as a client to Oracle Telephony Adapter Server, and invoke API methods on TeleDevice implementations. Events that are sent by TeleDevice implementations are received by TeleDevice Test Utility and displayed on the user interface. Additionally, developers can use TeleDevice Test Utility to launch the softphone and perform unit testing for TelesetDevice implementations.

The user interface consists of the following components:

- Left device tab for configuring and starting TeleDevice clients
- Top right side contains multiple internal frames (one per Device)
- Bottom right side message window for displaying messages

These components are illustrated in the following figure.

TeleDevice Test Utility Interface



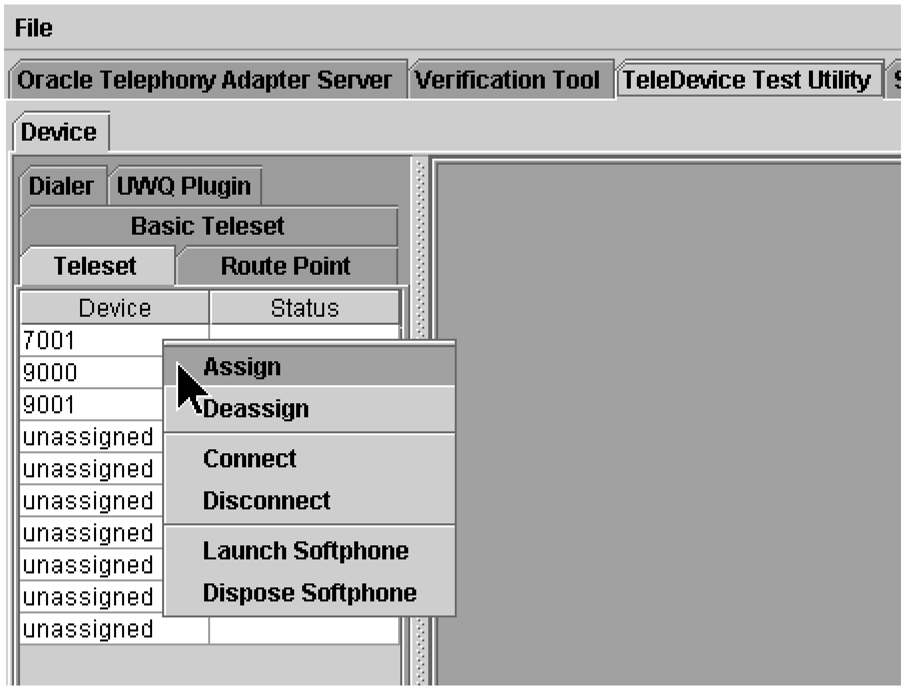
Device	Status
7001	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	

```
occtEventName : agentLoginEvent
```

```
Assigning {7001,8001,9001}
Teleset(7001,8001,9001) successfully assigned.
Teleset 7001 successfully connected.
```

To configure a device, right click on it to open a menu and select an option: Assign, Deasign, Connect, Disconnect, Launch Softphone, or Dispose Softphone, as shown in the following illustration.

TeleDevice Configuration Options



Device	Status
7001	
9000	
9001	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	
unassigned	

TeleDevices

Each TeleDevice simulates a communication channel between an Oracle component and the third-party component, through the different SDK interfaces.

Teleset

Represents a Teleset device in Oracle Telephony Adapter Server, and simulates the channel between Oracle Telephony Manager and Oracle Telephony Adapter Server. Use this TeleDevice only for Standard or Advanced Integration.

Route Point

Represents a Route Point device in Oracle Telephony Adapter Server, and simulates the channel between Inbound Telephony Server and Oracle Telephony Adapter Server. This TeleDevice should only be used for Advanced Integration.

Dialer

Represents a Dialer device in Oracle Telephony Adapter Server, and simulates the channel between Advanced Outbound Dial Server and Oracle Telephony Adapter Server. Use this TeleDevice only for Standard or Advanced Integration with Oracle Advanced Outbound.

Basic Teleset

Represents a Teleset device in a third party provider's process, and simulates the channel between Media Provider Plug-in (in the Universal Work Queue Client process space) and the third party provider. Use this TeleDevice only for Basic Integration.

UWQ Plug-in

Represents an Oracle Universal Work Queue Client plug-in, and simulates the channel between Oracle Universal Work Queue Client, and the third-party provider. Use this TeleDevice only for Basic Integration.

Assigning TeleDevices

Each of the Oracle SDK Integrated Test Utility TeleDevice tabs has a table of ten rows. Each row represents a TeleDevice object. The TeleDevice objects must be associated to specific configurations to be able to connect to Oracle Telephony Adapter Server. This association is called "Assign" in the TeleDevice Test Utility. Developers can assign up to ten devices.

Assigning TelesetDevice

Use the following procedure to assign a TelesetDevice.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

Successfully start Oracle Telephony Adapter Server.

Steps

1. Select the TeleDevice Test Utility tab.

   The Device page appears.
2. Select the Teleset sub tab.
3. Select the Device or Devices to assign.
4. Right click on the selected rows.

   A menu appears.
5. Choose **Assign**.

   The Assign Teleset window appears.
6. Enter the three extension numbers.
7. Click **OK**.

   The TeleDevice Test Utility Log window displays the progress and completion of the assignments.

Assigning Route Point Device

Use the following procedure to assign a Route Point Device.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Successfully start Oracle Telephony Adapter Server.

Steps

1. Select the TeleDevice Test Utility tab.

   The Device page appears.
2. Select the Route Point sub tab.
3. Select the Device or Devices to assign.
4. Right click on the selected rows.

   A menu appears.
5. Choose **Assign**.

   The Assign RoutePoint window appears.
6. Enter the Route Point Number.
7. Optionally, if you want this route point to receive route requests, click **Route Requested**.
8. For Nortel Meridian with Intel CT Connect/NetMerge Call Processing Software only, in the Immediate Treatment field specify the immediate treatment of inbound calls arriving at this route point CDN (Control Directory Number). Enter ##R for ringback, ##M for music, or ##S for silence.
9. For Nortel Meridian with Intel CT Connect/NetMerge Call Processing Software only, if music treatment (##M) is specified in step 7, in the Music Route Number field specify the route number of a music source that is configured in the Meridian PBX. Enter # followed by a two-digit route number specified in hexidecimal. For example, if the music route number is 10, then enter #0A in the Music Route Number field.
10. Click **OK**.

    The TeleDevice Test Utility Log window displays the progress and completion of the assignments.

Assigning Dialer

Use the following procedure to assign a Dialer.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Successfully start Oracle Telephony Adapter Server.

Steps

1. Select the TeleDevice Test Utility tab.

   The Device page appears.
2. Select the Dialer sub tab.
3. Select the Device or Devices to assign.
4. Right click on the selected rows.

   A menu appears.
5. Choose **Assign**.

   The Assign Dialer window appears.
6. Enter the Dialer ID and Vdu ID. Dialer ID is the ID to uniquely identify this Dialer Device. Vdu ID is the ID to identify the Vdu that the Dialer Device will control.
7. Press **OK**.

   The TeleDevice Test Utility Log window displays the progress and completion of the assignments.

Assigning Basic Teleset

Use the following procedure to assign a Basic Teleset.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Successfully start Oracle Telephony Adapter Server.

Steps

1. Select the TeleDevice Test Utility tab.

   The Device page appears.
2. Select the Basic Teleset sub tab.
3. Select the Device or Devices to assign.
4. Right click on the selected rows.

A menu appears.

5. Choose **Assign**.

   The Assign Basic Teleset window appears.

6. Enter the 3rd Party url, Agent ID, Teleset ID and Listener Port.

7. Click **OK**.

   The TeleDevice Test Utility Log window displays the progress and completion of the assignments.

Assigning UWQ Plug-in

Use the following procedure to assign a UWQ Plug-in.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Successfully start Oracle Telephony Adapter Server.

Steps

1. Select the TeleDevice Test Utility tab.

   The Device page appears.

2. Select the UWQ Plug-in sub tab.

3. Select the Device or Devices to assign.

4. Right click on the selected rows.

   A menu appears.

5. Choose **Assign**.

   The Assign UWQ Plug-in window appears.

6. Enter the Agent ACD ID, Agent ACD Password, Agent ACD Queue, Listener Port, 3rd Party url, Reconnect Interval(sec), and Log Level.

   1. Agent ACD ID: The Agent ACD ID to log in to the switch

   2. Agent ACD Password: The Agent ACD Password to login into the switch

   3. Agent ACD Queue: The Agent ACD Queue to log in to the switch

   4. Listener Port: The Listener port for Basic SDK HTTP Listener

   5. 3rd Party url: Third-party URL implementing the Callout handling

   6. Reconnect Interval: Number of seconds before the Basic SDK Plug-in tries to ping or reconnect to the third-party URL

7. Log Level: Error, warning, info, or verbose

7. Click **OK**.

   The TeleDevice Test Utility Log window displays the progress and completion of the assignments.

Connecting and Disconnecting a TeleDevice

When a TeleDevice is assigned, the next step is to connect it to Oracle Telephony Adapter Server. Use the following procedure to connect a TeleDevice.

> **Note:** Check that Oracle Telephony Adapter Server is started and running before connecting a TeleDevice.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

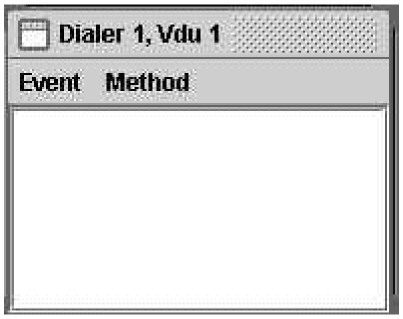
Assign at least one TeleDevice.

Steps

1. Select the appropriate TeleDevice tab: Dialer, UWQ Plug-in, Basic Teleset, Teleset, or Route Point.
2. Right click on the TeleDevice that you want to connect.

   A menu appears.
3. Choose **Connect**.
4. If the TeleDevice has connected successfully, a TeleDevice internal window appears, as shown in the following illustration.

TeleDevice Window



5. The message window shows that "RoutePoint X successfully connected" or "Teleset X successfully connected." Other TeleDevices display similar windows and messages.

Deassigning a TeleDevice

Removing a device assignment is called *deassigning*. Use the following procedure to deassign a TeleDevice.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Assign at least one TeleDevice.

Steps

1. Select the appropriate TeleDevice tab (Dialer, UWQ Plug-in, Basic Teleset, Teleset, or Route Point).
2. Right click on the TeleDevice that you want to deassign.

   A menu appears.
3. Select **Deassign**.

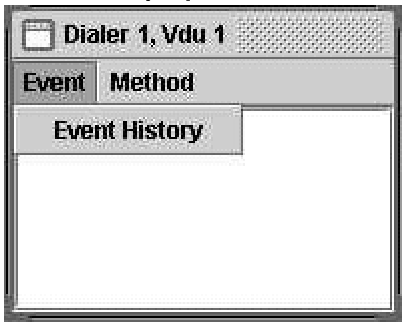
   The configuration associated with the row is deassigned and removed.

Viewing Events

In the Event History window you can view up to thirty events that are sent by TeleDevice object implementations and cached.

To open the Event History window, open a device message window (see Connecting and Disconnecting a TeleDevice) and select **Event** > **Event History**, as shown in the following illustration.

Event History Option



The Event History window opens, showing any event messages.

Invoking API Methods

You can invoke API methods for TelesetDevice, RoutePointDevice, DialerDevice, BasicTelesetDevice, and UWQPluginDevice.

Use the following procedures to invoke API methods for TeleDevices.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

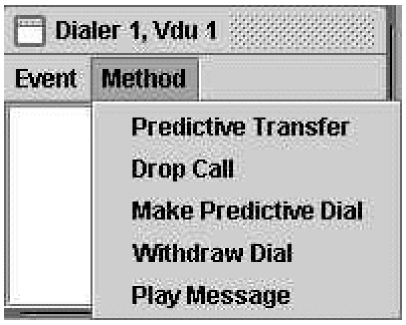
Connect TeleDevice.

Steps

1. In an opened TeleDevice internal message window, select **Method**.

   The Methods menu appears, as shown in the following illustration for the Dialer method.

   Methods Options

2. Select a method.

   For some methods, no result is apparent. For other methods, a window appears.
3. If a window appears, then enter the required information in the fields.
4. Click **OK**.

Route Point

RoutePointDevice invokes the following methods:

- assignMediaItemId
- routeCall

Teleset

TelesetDevice invokes the following methods:

- makeCall
- answerCall
- releaseCall
- holdCall
- retrieveCall
- loginAgent
- logoutAgent
- agentReadyOn
- agentReadyOff
- blindTransfer
- consultationCall(Transfer)
- completeTransfer
- consultationCall(Conference)
- completeConference

Dialer

DialerDevice invokes the following methods:

- Predictive Transfer
- Drop Call
- Make Predictive Dial
- Withdraw Dial
- Play Message

UWQ Plug-in

UWQ Plug-in invokes the following methods:

- Make Call
- Next Work
- Cancel Work Request
- Login Agent
- Logout Agent

Basic Teleset

Basic Teleset invokes the following methods:

- Make Call
- Agent Ready
- Agent Not Ready
- Login Agent

- Logout Agent
- Register Agent
- Unregister Agent

Disconnecting a TeleDevice

Use the following procedure to disconnect a TeleDevice.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Connect a TeleDevice.

Steps

1. Select the appropriate TeleDevice tab (Dialer, UWQ Plug-in, Basic Teleset, Teleset, or Route Point).
2. Right click on the TeleDevice that you want to disconnect.

   A menu appears.
3. Choose **Disconnect**.

Launching the Softphone

To verify the softphone functionality, use the following procedures to launch and close the softphone.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Assign a Teleset Device.

Steps

1. Select the TeleDevice Test Utility tab > Teleset sub tab.
2. In the Device column, right click on the applicable, assigned teleset device number.

   A menu appears.

3. Select **Launch Softphone**.

   The loginAgent window appears.

4. Enter the acdAgentId, acdAgentPassword and acdQueue.
5. Click **OK**.

   The softphone interface opens.

Closing the Softphone

Use the following procedure to close the softphone.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

None

Steps

1. Select the TeleDevice Test Utility tab > Teleset sub tab.
2. In the Device column, right click on the applicable teleset device number.

   A menu appears.

3. Select **Dispose Softphone**.

   The Agent Logout window appears.

4. Click **OK**.

   The softphone interface closes.

Switch Simulator

The Switch Simulator is one of the Interaction Center server processes that is used in demonstration environments where a real switch is not available. The switch simulator supports teleset and route point functions, and three integration levels: Basic, Standard and Advanced.

In Standard and Advanced Integration, the Switch Simulator acts as a simulated switch that listens for Oracle Telephony Adapter's Server's teleset and route point commands (makeCall) and fake events (callRinging) to the appropriate teleset or route point extensions.

In Basic Integration, the Switch Simulator acts as a third party HTTP listener that listens for callouts from the Basic Telephony SDK plug-ins and fake HTTP and XML events to the appropriate Basic Telephony SDK plug-ins.

The Switch Simulator interface consists of two tabs:

- Server tab, to start and stop the Switch Simulator server, and to configure and view logs.
- Configure tab, the extension configuration for starting the Switch Simulator server.

Starting and Stopping the Switch Simulator

Use the following procedures to start or stop the Oracle SDK Integrated Test Utility Switch Simulator.

Log in

Not applicable

Responsibility

Not applicable

Prerequisite

Configure the Switch Simulator.

Steps

1. In the Oracle SDK Integrated Test Utility, select the Switch Simulator tab > Server sub tab.

   The Server page opens.
2. Do one of the following:
   1. To start the Switch Simulator, click **Start**.

      The Switch Simulator launches as a background Java process, and starts by using the current extension configuration.
   2. To stop the Switch Simulator, click **Stop**.

      The Switch Simulator background process terminates.

Configuring Switch Simulator Server Settings

Use the following procedure to configure the Switch Simulator server settings.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

None

Steps

1. In the Oracle SDK Integrated Test Utility, select the Switch Simulator tab > Server sub tab.

   The Server page opens.

   > **Note:** Any changes you make while the Switch Simulator is running do not take effect until you stop and restart the Switch Simulator.

2. Choose the extent of detail in the Switch Simulator Log by selecting a logging level:

   1. error: Print only error messages.
   2. warning: Print error and warning messages.
   3. info: Print error, warning and informational messages.
   4. verbose: Print all messages.

3. Optionally, you can configure the directory of the log file by typing the directory path into the Log Directory text box. To name the log file, use the pattern swsimyyyymmdd_hhmmss.log. The default directory is oracle/apps/cct/bin/log.
4. Configure the port number of the Switch Simulator by entering the port number in the Server Port field. This is the port number that Oracle Telephony Adapter Server uses to communicate with the Switch Simulator for Standard and Advanced Integration.
5. Configure the port number of the Switch Simulator HTTP listener by entering the port number in the HTTP Server Port text box. This is the port number that UWQ Client uses to communicate with the Switch Simulator in Basic Integration.
6. Select **File** > **Save** to save the configurations.

Configuring Simulated Extensions and Route Points

Use the following procedure to configure simulated extensions and route points.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

None

Steps

1. Select the Switch Simulator tab > Configure sub tab.

   The Extensions and Route Points page appears.

2. In the Extension Range fields, enter extension ranges 1 through 3 for simulated telesets.

3. In the three Route Point fields, enter route point extensions for simulated route points.
4. Select **File** > **Save** to save the configurations.

Configuring the Switch Simulator for Standard or Advanced Integration

Use the following procedure to configure the Oracle SDK Integrated Test Utility's Oracle Telephony Adapter Server settings for Standard Integration or Advanced Integration.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

- Configure Oracle Telephony Adapter Server to connect to the Switch Simulator.
- Configure the Switch Simulator.

Steps

1. Select the Oracle Telephony Adapter Server tab > Middleware sub tab.

   The Middleware page opens.
2. From the Middleware Type list, select **Java Adapter**.
3. In the Factory Class Name field, enter

   `oracle.apps.cct.sdk.sample.advanced.SampleTeleDeviceFactory`
4. In the CTI Server IP Address 1 field, enter the IP Address of the machine on which the Switch Simulator starts.
5. In the CTI Server IP Port 1 field, enter the Server Port (*not* the HTTP Server Port) of the Switch Simulator.
6. Select the TeleDevice Test Utility tab > Device sub tab.

   The Device page opens.
7. In Teleset Device fields 1 through 3, assign extensions within the range that are the same as those configured in the Switch Simulator.
8. In the Route Point Status fields, assign Route Point Number to be one of the Route Point extensions that are configured in the Switch Simulator.
9. Select **File** > **Save** to save the configuration.

Configuring the Switch Simulator for Basic Integration

When developing Basic Integration, the Switch Simulator acts as the third-party provider that listens for callouts from UWQ Client. The Basic Teleset and UWQ Plug-in

TeleDevices can be used to simulate connectivity between UWQ Client and the Switch Simulator.

Use the following procedure to configure the Oracle SDK Integrated Test Utility's Switch Simulator for Basic Integration.

Log in

Not applicable

Responsibility

Not applicable

Prerequisites

Configure the Switch Simulator.

Steps

1. Select the TeleDevice Test Utility tab > Device sub tab.

   The Device page opens.

2. Do *either* a or b:

   1. Select the Basic Teleset sub tab.

      1. Right click on a Device that you want to configure.

         A menu appears.

      2. Select **Assign**.

         The Assign Basic Teleset window opens.

      3. In the 3rd Party url field, enter

         `http://<IP Address of Switch Simulator>:<HTTP Server Port of Switch Simulator>/`

   2. Select the UWQ Plug-in tab.

      1. Right click on a Device that you want to configure.

         A menu appears.

      2. Select **Assign**.

         The Assign UWQ Provider Plug-in window opens.

      3. In the 3rd Party url field, enter

         `http://<IP Address of Switch Simulator>:<HTTP Server Port of Switch Simulator>/`

Javadoc

Javadoc is a facility provided within the Java Development Kit that produces HTML documentation from a program. Javadoc reads the source code and parses specially formatted and positioned comments into documentation.

To access the Javadoc for Oracle Telephony Adapter SDK, select the Help tab in the Oracle Telephony SDK Integrated Test Utility.

6 Deploying the Telephony Adapter

This chapter covers the following topics:

- Introduction

Introduction

This section describes the steps involved in packaging and uploading Oracle Telephony Adapter to deploy the implementation in a live interaction center environment. The following steps outline the deployment.

- Package the Telephony Adapter.
- Upload the Telephony Adapter to Interaction Center Server Manager.
- Configure CTI Middleware, Telesets and Route Points.
- Configure the Server Group.
- Test with Interaction Center, softphone and Oracle Universal Work Queue.

Prerequisites

- Install Oracle eBusiness Suite.
- Install Oracle Interaction Center Server Manager.

Steps:

1. Package the telephony adapter. The type of file package for Oracle Telephony Adapter depends on the development language of the adapter implementation.

   Do one of the following:

   - For an adapter developed in Java, all Java classes should be packaged into a single jar file using the Java JAR utility, such as impl.jar.
     - For an adapter developed in C, compile all code into a single DLL file, such as impl.dll.
2. Identify the node where Oracle Telephony Adapter Server will be deployed.
3. Copy the jar or dll file to the directory "3rdParty" that is under the directory where Interaction Center Server Manager is installed.
4. Configure CTI middleware, telesets and route points. Define a CTI middleware configuration before Oracle Telephony Adapter Server can start loading the adapter implementation. Do one of the following:

- For a Java adapter implementation, use the Custom Java Adapter middleware type.
    - For a C adapter implementation, use the Custom C Adapter middleware type.

5. Configure the server group. Define Oracle Telephony Adapter Server and related servers before starting Oracle Telephony Adapter Server.

Related Topics

- *Oracle Interaction Center Implementation Guide*
- *Oracle Advanced Inbound Implementation Guide*

7 Basic Integration

This chapter covers the following topics:

- Introduction
- Architecture
- Understanding Basic Integration
- User Interface
- User Actions and Corresponding Basic Telephony Integration Actions
- Software Requirements and Development Strategies
- Implementing Basic Integration
- Implementing Callouts
- Error Reporting
- Implementing Events
- Additional Interaction Keys
- Deploying Basic Integration
- Media Action Configuration
- User Profile Configuration
- Implementing Basic Web Callback
- Testing Basic Implementation
- Testing with SDK Integrated Test Utility
- Testing with Oracle eBusiness Suite Application
- Sample Code
- Switch Simulator
- Design

Introduction

This chapter describes Oracle Telephony Adapter SDK Basic Integration, and explains how to use the infrastructure and facilities of Basic Integration to integrate the Oracle eBusiness Suite with a third-party telephony system.

This information is intended for developers and consultants who intend to use Oracle Telephony Adapter SDK Basic Integration to integrate Oracle eBusiness Suite with a third-party telephony vendor.

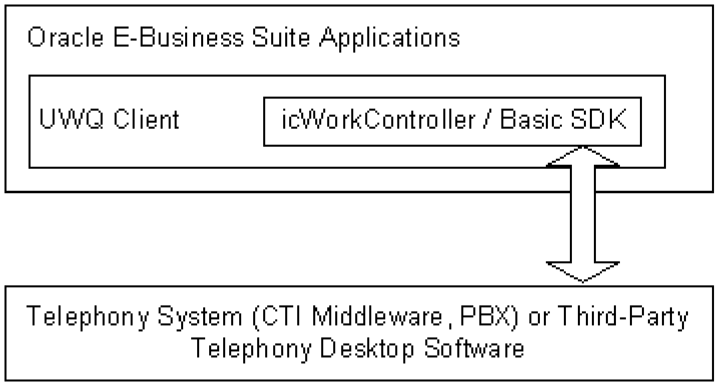
Architecture

Basic Integration of Oracle Telephony Adapter SDK consists of the following:

- Oracle Universal Work Queue Client
- Basic Telephony SDK plug-in

As the following diagram shows, Basic Integration integrates the telephony system (middleware, PBX), or third-party desktop software, with the Oracle Universal Work Queue client in the Oracle eBusiness Suite.

Basic Integration Architecture

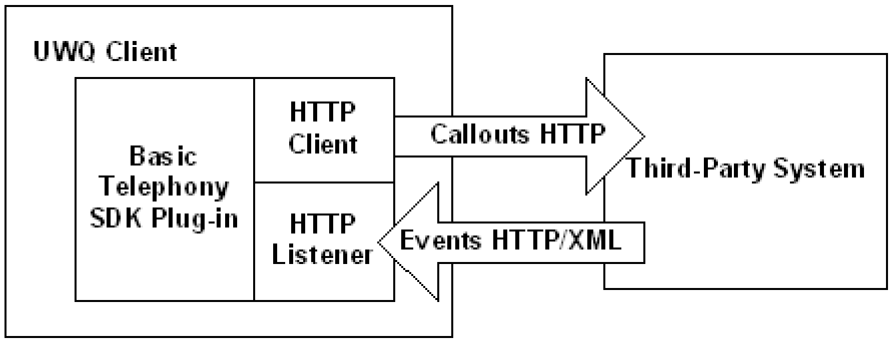

Oracle E-Business Suite Applications
UWQ Client
icWorkController / Basic SDK
Telephony System (CTI Middleware, PBX) or Third-Party Telephony Desktop Software

The Basic Telephony SDK plug-in enables third-party telephony systems to communicate with the Oracle eBusiness Suite by using one of two types of HTTP messages:

- Callout: Messages are sent from Oracle eBusiness Suite to the third-party system using a HTTP GET request.
- Event: Messages are sent from a third-party system using an HTTP POST request.

The following diagram illustrates sending callouts from the HTTP client to the third-party telephony system, and events from the third-party telephony system to the HTTP listener.

Callouts and Events



Understanding Basic Integration

The components of Basic Integration are explained in the following sections.

User Interface

The Basic Integration user interface includes the following components.

Oracle Universal Work Queue

On the Oracle Universal Work Queue form, "Basic Telephony" appears on the left Queue Selector Pane whenever Basic Telephony is enabled. Selecting Basic Telephony on the Queue Selector Pane displays the right Queue Details Pane. A row with the title "<ANY>" appears. When in the Basic Telephony mode, the Get Work button does not appear. Double clicking in the <ANY> row triggers Get Work.

icWork Controller

icWork Controller is a floating window that is used to Stop Media and to get Next Media.

User Actions and Corresponding Basic Telephony Integration Actions

The following paragraphs explain the Basic Integration actions and their corresponding user actions.

User Opens the Oracle Universal Work Queue Form

When a user opens the Oracle Universal Work Queue form, the basic framework elements set up and prepare Oracle Universal Work Queue for use. The Basic SDK client plug-in is loaded and performs the following actions:

- Loads the user profile values from the database and determines how to connect to the third-party system.
- Starts the HTTP listener on a randomly-allocated port or the optionally-specified port.
- Sends Register and Login callouts to the third-party system to establish a session.
- Starts the reconnect checking process.

User Selects the Basic Telephony Work Queue

The right Queue Details Pane displays <ANY>, and the Basic SDK client plug-in takes no action.

User Selects the Basic Telephony <ANY> Record

Basic SDK client plug-in sends Ready callout to the third-party system and sets itself in Get Work mode if there is no prior ScreenPop event received. Otherwise, the ScreenPop event is delivered.

User Selects End Interaction on eBusiness Suite Form

Basic SDK client plug-in sends Ready callout to the third-party system and sets itself in Get Work mode if there is no prior ScreenPop event received. Otherwise, the ScreenPop event is delivered.

User Selects Stop Media Button on icWork Controller

Basic SDK client plug-in sends NotReady callout to the third-party system and sets itself in non-Get Work mode.

User Closes Oracle Universal Work Queue Form

- The Basic SDK client plug-in sends Logout and Unregister to the third-party system.
- Basic SDK client plug-in stops HTTP listener and frees any resources.

User Receives an Inbound Call

- Basic SDK integration sends a ScreenPop event to the Basic SDK client plug-in.
- Basic SDK client plug-in delivers the data to Oracle Universal Work Queue for screen pop if it is in Get Work mode. Otherwise, it puts the event in its FIFO queue.
- Basic SDK client plug-in creates a media item record in Oracle Customer Interaction History.

User Answers an Inbound Call

- Basic SDK integration sends a CallEstablished event to the Basic SDK client plug-in.
- Basic SDK client plug-in creates a media item life cycle segment in Oracle Customer Interaction History.

User Releases an Inbound Call

- Basic SDK integration sends a CallReleased event to the Basic SDK client plug-in.
- Basic SDK client plug-in updates the media item life cycle segment in Oracle Customer Interaction History.

User Selects Dial on the Basic Panel

The dial string entered in the text field on Basic Panel is sent with MakeCall callout to a third-party system.

Software Requirements and Development Strategies

The following software requirements and development strategies apply to the Oracle Telephony Adapter SDK Basic Integration.

Minimum Patch Level

The minimal patch level required by Basic Telephony SDK is Release 11.5.8 + Interaction Center Family Pack P.

Programming Languages

Because the Basic Telephony SDK is defined in HTTP and XML, developers can use any programming language in which HTTP request processing can be implemented easily.

Development Strategies

The following two examples demonstrate different ways of developing Basic Telephony Integration.

- Client-based (recommended): In the client-based strategy, integration is done on client-side third-party software, such as a softphone, and is installed on each desktop machine. The Basic SDK communicates locally with the third-party software.
- Server based: In the server-based strategy, custom integration is done on a server-based platform and is installed centrally on the server. No client software is installed on the desktop machine. Basic SDK communicates with third-party software through the LAN.

Implementing Basic Integration

This section describes the implementation of Basic Integration.

Implementing Callouts

Callouts are messages sent from Oracle eBusiness Suite to the third-party system using HTTP GET requests. Developers must implement an HTTP listener that is capable of handling the HTTP GET requests specified in this document. The URL to the HTTP listener is then provided to the Basic SDK client plug-in through the use of a profile option. Parameters are passed to the URL using GET request semantics (CGI parameters). All callouts are handled by one single URL with the FunctionName parameter being a different value.

Register

Definition: Establish a session with the telesetNumber specified and notify third-party system on the IP address and port to which the Basic SDK client plug-in is listening.

Parameter Name	Value
FunctionName	Register
telesetNumber	telesetNumber entered by agent
agentID	ACD agent ID
ipAddress	ipAddress of the Basic SDK client plug-in HTTP Listener
port	Port of the Basic SDK client plug-in HTTP Listener

URL example:

```
http://host:port/?FunctionName=Register&telesetNumber=1001&agentI
D=10001&ipAddress=130.46.1.1&port=8088
```

Unregister

Definition: End the session with the telesetNumber specified.

Parameter Name	Value
FunctionName	Unregister
telesetNumber	the telesetNumber entered by agent
agentID	ACD agent ID

URL example:

```
http://host:port/?FunctionName=Unregister&telesetNumber=1001&agen
tID=10001
```

Login

Definition: (Optional) Log in the agent to the specified Teleset Number. The third-party system or developer can decide whether to react to this callout, if manual login is desired.

Parameter Name	Value
FunctionName	Login
telesetNumber	the telesetNumber entered by agent
agentID	ACD agent ID
agentPassword	ACD password of the agent
agentQueue	ACD queue the agent should login to

URL example:

```
http://host:port/?FunctionName=Login&telesetNumber=1001&agentID=1
0001&agentPassword=123&agentQueue=1234
```

Logout

Definition: (Optional) Log out the agent to the specified Teleset Number. The third-party system or developer can decide whether to react to this callout.

Parameter Name	Value
FunctionName	Logout
telesetNumber	the telesetNumber entered by agent
agentID	ACD agent ID
agentPassword	ACD password of the agent
agentQueue	ACD queue the agent should login to

URL example:

`http://host:port/?FunctionName=Logout&telesetNumber=1001&agentID=10001&agentPassword=123&agentQueue=1234`

Ready

Definition: Agent is ready for work.

Parameter Name	Value
FunctionName	Ready
telesetNumber	TelesetNumber entered by agent
agentID	ACD agent ID

URL example:

`http://host:port/?FunctionName=Ready&telesetNumber=1001&agentID=10001`

NotReady

Definition: Agent is not ready for work.

Parameter Name	Value
FunctionName	NotReady
telesetNumber	TelesetNumber entered by an agent
agentID ACD	ACD agent ID

URL example:

`http://host:port/?FunctionName=NotReady&telesetNumber=1001&agentID=10001`

MakeCall

Definition: Make a call.

Parameter Name	Value
FunctionName	MakeCall
telesetNumber	TelesetNumber entered by agent
agentID ACD	ACD agent ID
dialString	Destination number to dial

URL example:

```
http://host:port/?FunctionName=MakeCall&telesetNumber=1001&agentI
D=10001&dialString=6509991111
```

CheckStatus

Definition: Check if the third-party system is up and running. This callout is called periodically by the Basic SDK client plug-in to detect lost connection and reconnect. You can configure the interval between each callout.

Parameter Name	Value
FunctionName	CheckStatus

URL example:

```
http://host:port/?FunctionName=CheckStatus
```

Error Reporting

For each callout the developer must return a 200 HTTP response code indicating that the callout has processed the request. Successful callouts return an HTTP response with the 200 response code and no content. Errors in the HTTP response message may be reported using the following XML format:

```
<CCTSDK>


<event name="Error">
```

```
<data name="occtErrorMsg" value="error message here"/>

</event>

</CCTSDK>
```

Implementing Events

Events are messages sent from the third-party system to the Basic SDK client plug-in using HTTP POST request. You can construct the URL of the Basic SDK client plug-in by using the IP address and port parameters from Register Callout. The request body must be an XML formatted message that is defined as follows:

> **Note:** Formatting is based on standard XML conventions, for example white spaces are ignored. The value is case sensitive.

```
<CCTSDK>

<event name="eventName">

<data name="name1" value="value1"/>

<data name="name2" value="value2"/>

<data name="name3" value="value3"/>

    ...

</event>

</CCTSDK>
```

ScreenPop

Definition: ScreenPop event triggers a screen pop in the Oracle eBusiness Suite. Developers must determine when to send a ScreenPop event, typically when the agent gets an inbound call. Developers must decide whether to send a ScreenPop event for every call or only for certain types of calls. For example, internal or outbound calls are not often considered for screen pops.

Parameter Name	Value
occtSourceID	Logical ID used to connect between multiple events of the same physical call. Multiple events generated from the same physical call must have the same occtSourceID value, that is, telephony middleware call IDs. SourceID values must be a number. Valid values are 0 and positive and negative numbers with magnitude 1.0E-130 to 9.99..E125
occtANI	ANI. Must be <= 30 characters.
occtDNIS	DNIS. Must be <= 30 characters.
occtClassification (Optional)	Field passed to Oracle Universal Work Queue for determining the appropriate media action for this ScreenPop event. Must be <= 64 characters.

XML example:

```
<CCTSDK>

<event name="ScreenPop">

<data name="occtSourceID" value="1"/>

<data name="occtANI" value="50666546"/>

<data name="occtDNIS" value="7404"/>

</event>

</CCTSDK>
```

CallEstablished (Optional)

Definition: Indicates that a call is established. CallEstablished event triggers the creation of a life cycle segment in Oracle Customer Interaction History. The start timestamp of the segment will be logged. This event must be sent after ScreenPop event.

Parameter Name	Value
occtSourceID	Logical ID used to connect between multiple events of the same physical call. Multiple events generated from the same physical call must have the same occtSourceID value, that is, telephony middleware call IDs. This value must be a number. Valid values are 0, and positive and negative numbers with magnitude 1.0E-130 to 9.99..E125

XML example:

```
<CCTSDK>

<event name="CallEstablished">

<data name="occtSourceID" value="1"/>

</event>

</CCTSDK>
```

CallReleased (Optional)

Definition: Indicates a call is released. CallReleased event triggers the life cycle segment that is being updated in Oracle Customer Interaction History. The end timestamp and the duration of the segment are logged. This event must be sent after CallEstablished.

Parameter Name	Value
occtSourceID	Logical ID used to connect between multiple events of the same physical call. Multiple events generated from the same physical call must have the same occtSourceID value, that is, telephony middleware call IDs. This value must be a number. Valid values are 0, and positive and negative numbers with magnitude 1.0E-130 to 9.99..E125

XML example:

```
<CCTSDK>

<event name="CallReleased">

<data name="occtSourceID" value="1"/>
```

```
</event>

</CCTSDK>
```

Additional Interaction Keys

In addition to the keys specified in the ScreenPop event section, the following table lists keys that may also be sent by the third-party system as name/value pairs as part of the ScreenPop event. These keys may be used for out-of-the-box screen pops by Oracle Customer Care and Oracle TeleSales.

Interaction Key	Description
ContactNum	Oracle TeleSales Key 3
PromotionCode	Oracle TeleSales Key 4
AccountCode	Oracle TeleSales Key 5
QuoteNum	Oracle TeleSales Key 6
CustomerID	Oracle TeleSales Key 7
Product	Oracle TeleSales Key 8
SystemName	Oracle TeleSales Key 9
ContractNum	Oracle TeleSales Key 10
PreferredID	Oracle TeleSales Key 11
ScreenPopType	Oracle TeleSales Key 12
occtCallBirthTime	Oracle TeleSales Key 13
occtCallAnswerTime	Oracle TeleSales Key 14
CustomerProductID	Oracle TeleSales Key 15
InventoryItemID	Oracle TeleSales Key 16
InvoiceNum	Oracle TeleSales Key 17
LotNum	Oracle TeleSales Key 18
OrderNum	Oracle TeleSales Key 19
ProductName	Oracle TeleSales Key 20
PurchaseOrderNum	Oracle TeleSales Key 21
ReferenceNum	Oracle TeleSales Key 22

Interaction Key	Description
RevisionNum	Oracle TeleSales Key 23
RMANum	Oracle TeleSales Key 24
SerialNum	Oracle TeleSales Key 25
ServiceRequestNum	Oracle TeleSales Key 26
CustomerNum	Oracle TeleSales Key 27
CustomerName	Oracle TeleSales Key 28
occtANI	Oracle TeleSales Key 29
occtClassification	Oracle TeleSales Key 31
LanguageCompetency	Language Competency
KnowledgeCompetency	Knowledge Competency
ProductCompetency	Product Competency
occtMediaType	Media Type
employeeID	Employee ID
occtMediaItemID	Media Item ID
AccountNum	Account Number
SiteNum	Site Number
RepairNum	Repair Number
DefectNum	Defect Number
CustomerStatus	Customer Status
EventCode	Event Registration Code
CollateralReq	Collateral Request Number
SocialSecurityNumber	Social Security Number
CAMPAIGN_SCHEDULE_NAME	Campaign Schedule Name
CompetencyType	Competency Type
Competency	Competency
occtCreationTime	Time of the Day
MarketingPIN	Marketing PIN

Interaction Key	Description
ServiceKey	Service Key
ContractNumModifier	Contract Number Modifier

Example of XML for ScreenPop Event with CustomerID:

```
<CCTSDK>

<event name="ScreenPop">

<data name="occtSourceID" value="134"/>

<data name="CustomerID" value="1234567"/>

<data name="occtANI" value="40666546"/>

<data name="occtDNIS" value="7404"/>

</event>

</CCTSDK>
```

Related Topics

- *Oracle Advanced Inbound Implementation Guide*
- *Oracle Customer Care User Guide*
- *Oracle TeleSales User Guide*

Deploying Basic Integration

This section includes the following topics.

Media Action Configuration

Media Action must be defined in the Oracle Universal Work Queue Media Action HTML administration page for the proper Oracle eBusiness Suite Application to pop on the ScreenPop event. The following is an example to pop Customer Care Form on any classification and Oracle Universal Work Queue Diagnostics Form on Diagnostics classification.

Media Type	Classification	Media Action
Basic Telephony		Customer Care Media Function
Basic Telephony	Diagnostics	Universal Work Queue Media Diagnostics Function
Basic Web Callback		Web callback integration with Oracle *i*Store and Oracle *i*Support

User Profile Configuration

Note: Basic Telephony is typically implemented with inbound telephony and Web callback as the only media. In these cases, Basic Telephony and Basic Web Callback are enabled and all other media queues are disabled.

If your system supports Basic Integration, the only supported media types are Basic Telephony and Basic Web Callback. Therefore, enable only those profile options for media. Set user profile options (site-level or user-level) as shown in the following table.

Profile	Value
IEU: Queue: Basic Telephony	Yes
IEU: Queue: Inbound Telephony	No
IEU: Queue: Advanced Outbound	No
IEU: Queue: Inbound E-mail	No
IEU: Queue: Acquired E-mail	No
IEU: Queue: Web Collaboration	No
IEU: Queue: Web Callback	Yes

Set additional profile options at the user-level as shown in the following table.

Profile	Meaning	Example
CCT: Basic Telephony: ACD ID	<agent acd id>	
CCT: Basic Telephony: ACD Password	<agent acd password>	
CCT: Basic Telephony: ACD Queue	<agent acd queue>	

CCT: Basic Telephony: Listener Port	(Optional) Listener port for the Basic SDK HTTP Listener. The port where the Universal Work Queue client listens for incoming events. Each instance of a basic teleset or Universal Work Queue plug-in must have a unique listener port. If you do not specify a value, then the UWQ client allocates the port randomly, determined by the Java Socket API which guarantees availability, that is, the port already in use will not be allocated again. It does not use a specific range. Formatting is based on standard XML syntax: • White spaces are ignored. • Values are case sensitive. • Character set is US-ASCII.	1234
CCT: Basic Telephony: Reconnect Interval	(Optional) Number of seconds of the reconnect interval when the Universal Work Queue plug-in tries to ping the basic SDK implementation. If the Basic SDK implementation does not respond to the ping request, then an error event is thrown.	10
CCT: Basic Telephony: Third Party URL	(Required) Third-party URL implementing callout handling. The HTTP listener where the Basic SDK implementation waits for callouts from the Universal Work Queue client.	http://localhost:9099
CCT: Basic Telephony: Log Level	(Optional) Error, warning, info or verbose	error
CCT: Basic Telephony: IC Plugin: Enable Log Window	(Optional) Yes/no, enable Log tab on Basic Panel	yes
CCT: Basic Telephony: IC Plugin: Enable Event Window	(Optional) Yes/no, enable Event tab on Basic Panel	yes
CCT: Basic Telephony: IC Plugin: Enable ScreenPop Window	(Optional) Yes/no, enable ScreenPop tab on Basic Panel	yes

Implementing Basic Web Callback

To enable Basic Web Callback, extend Basic Integration by implementing the dialCanonical method. In the DialCanonical method, third party integrators must use an agent's telephone to dial a customer's telephone number that is in the canonical format. One of the parameters sent in DialCanonical is application data (appData), which must be sent back to the client SDK plugin in the Call Established event for the physical call made in DialCanonical. The appData must be returned as the key value pair {occtAppData,appData}.

Example callout URL:

```
http://<dest-server-url>?FunctionName=DialCanonical&telesetNumber
=70011&agentID=7011&dialString=1234567&appData=10325
```

Parameters

The following parameters apply to Basic Web Callback.

- FunctionName = DialCanonical
- telesetNumber = TelesetNumber entered by the agent.
- agentID = ACD agent ID
- dialString = Dial string in the canonical form
- appData = Application data attached to the callout.

Web Callback requests are made by way of Oracle *i*Store and Oracle *i*Support. Administrators of these applications need to enable Web callback by selecting the default seeded server group BASIC_SDK in Oracle *i*Store and Oracle *i*Support.

Basic Web Callback Modes

The following modes are supported with Basic Web Callback:

Restrictions

Basic Web Callback

This is the default mode. In this mode, when agents have selected GetWork for Basic Web Callback, the system successively assigns Web callback requests to the agent. If no Web callback request is currently available, then the system polls until a Web callback request becomes available. The polling interval can be set as the profile value CCT: Basic WebCallback: Polling Interval in seconds. The default polling interval is 30 seconds.

When a Web callback request is assigned to an agent, a screen pop occurs at the agent's desktop and the callback phone call is immediately dialed out automatically.

Basic Web Callback with Preview Dialing

In this mode, when a Web callback request is assigned to an agent, a screen pop occurs, but the callback phone call is not dialed out. Instead, the callback phone number is displayed and the agent is required to click Dial on the phone for the dial out to occur. Agents should use this mode when they need time to read the customer information in the screenpop before dialing out.

To set up preview dialing, set the profile option CCT: Basic WebCallback: Enable Preview to Yes.

Basic Web Callback with Timed Preview Dialing

In this mode, when a Web callback request is assigned to an agent, a screen pop occurs, but the callback phone call is not made for a specified time, that is, the preview time interval. When the preview time interval expires, the callback phone call automatically dials out. Use this mode when you need to set a cap on the amount of time an agent can use to preview customer information.

To set up Timed Preview Dialing, first enable Preview Dialing, and then specify the preview time interval in the profile option CCT: Basic WebCallback: Maximum Preview Interval.

Testing Basic Implementation

Basic Implementation can be tested in one of two ways:

- Testing with SDK Integrated Test Utility
- Testing with Oracle eBusiness Suite Application

Testing with SDK Integrated Test Utility

The Basic SDK Implementation can be tested using the SDK Integrated Test Utility. This is useful in the development stage of the Basic SDK Implementation when access to Oracle eBusiness Suite is not available. The SDK Integrated Test Utility emulates the Basic SDK client plug-in and enables developers to invoke callouts to their Basic SDK implementations and receive HTTP/XML events.

1. Start the SDK Integrated Test Utility
2. Run sdkrun.bat.
3. In the TeleDevice Test Utility tab, select the Basic Teleset tab.
4. Select a row and right click.

   A menu appears.
5. Select **Assign**.

   A dialog box appears.
6. Enter the URL, agent ID, teleset ID and listener port.
7. Click **OK**.
8. The selected row is filled with agentID and telesetID information.
9. Right click the selected row.
10. Choose **Connect**.

    An internal window (the Basic Teleset Watch window) appears.

The Basic Teleset Watch Window consists of the following components:

- Pull Down menu Event and Method
- Current Event Display

Developers can use the pull down menu Method to invoke HTTP callouts to their own implementation. HTTP/XML events received by the Basic Teleset Watch Window can be viewed in the Current Event Display area. Developers can also use the Event pull down menu to display the list of events received by the Basic Teleset.

Testing with Oracle eBusiness Suite Application

Use the configuration described in Deploying Basic Integration to create a user with the appropriate Oracle eBusiness Suite responsibility. After you launch the Universal Work Queue form, perform the inbound call test listed in the following table.

#	Procedure	Expected Outcome	Result	Comments
A1	Launch Universal Work Queue form	Extension Number Dialog Appears		
A2	Enter Extension Number	Universal Work Queue Client should be launched without error message. Basic SDK Integration should receive Register and Login callouts.		
A3	Double click Basic Telephony <ANY> record	Basic SDK Integration should receive Ready callout		
A4	Generate an inbound call to the agent logged on	Basic SDK Integration should send ScreenPop event. A screen pop should occur.		
A5	Agent answer the call	Basic SDK Integration should send CallEstablished event.		
A6	Agent release the call	Basic SDK Integration should send CallReleased event		
A7	Press End Interaction	Basic SDK Integration should receive Ready callout		

#	Procedure	Expected Outcome	Result	Comments
A8	Press Stop Media button on icWork Controller	Basic SDK Integration should receive NotReady callout		
A9	Close from Universal Work Queue	Oracle Universal Work Queue form should be successfully closed, Basic SDK Integration should receive Logout and Unregister callouts		

Sample Code

The Basic Integration sample code consists of four Java source files in the directory oracle/apps/cct/java/sample/basic.

- BasicSDKException.java
- BasicSDKManager.java
- BasicSDKBootstrap.java
- SampleBasicSDKManager.java

In the following example implementation of the Basic SDK integration with the Switch Simulator, the following functions are implemented.

- Callouts: Register, Unregister, MakeCall
- Events: Screenpop, CallEstablished, CallReleased

The following table describes the location of the Basic Integration sample code and the functionality supported by the corresponding sample implementation.

Location	Sample Code	Supported Functions
oracle/apps/cct/java/sample/basic	BasicSDKBootsrap.java, BasicSDKException.java, BasicSDKManager.java, SampleBasicSDKManager.java	Basic SDK Integration supports all functions.

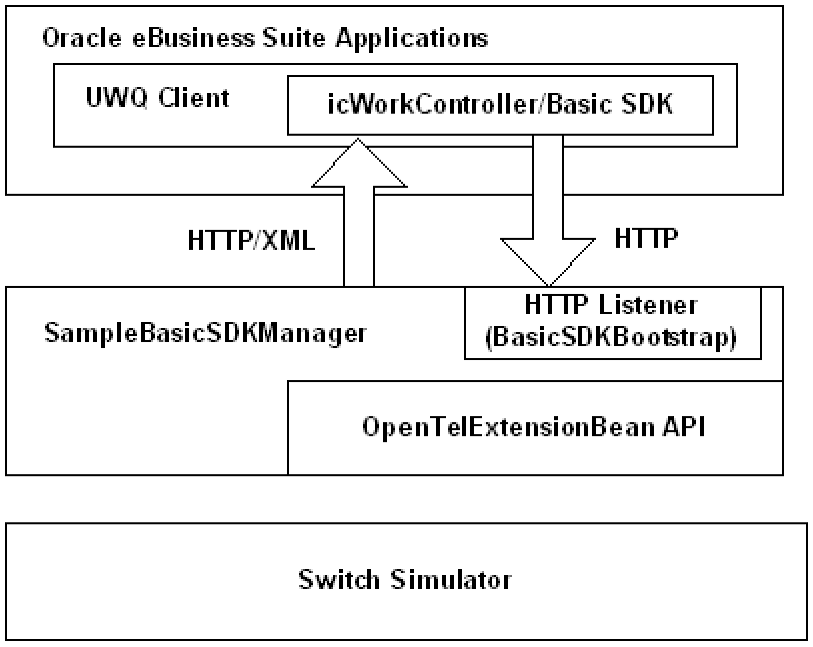
Switch Simulator

The Switch Simulator is one of the Interaction Center server processes that is used in demonstration environments where a real switch is not available. The simulator supports teleset and route point functions. Access the Switch Simulator programmatically by using the OpenTel Bean API (a Java API, package oracle.apps.cct.openTel.bean). The simulator models each line of the teleset and route point as extension objects, which you can access by using the OpenTelExtensionBean. The Switch Simulator sends OpenTel events.

Design

In the following example implementation, Basic SDK Integration acts as a bridge between the Basic SDK client plug-in residing in Oracle eBusiness Suite and the Switch Simulator. SampleBasicSDKManager handles HTTP callouts, which are translated to OpenTel Bean API. OpenTel events sent by the Switch Simulator are received and translated into XML, and then sent to the Basic SDK client plug-in using HTTP.

Basic Integration Example Scenario


Oracle eBusiness Suite Applications
UWQ Client
icWorkController/Basic SDK
HTTP/XML
HTTP
SampleBasicSDKManager
HTTP Listener (BasicSDKBootstrap)
OpenTelExtensionBean API
Switch Simulator

The following descriptions explain how each function is implemented.

Register Callout

When SampleBasicSDKManager receives a Register request, it creates an OpenTelExtensionBean object that connects to the specified extension running in Switch Simulator. An event listener is added to the OpenTelExtensionBean object so that OpenTel events can be received.

Unregister Callout

When SampleBasicSDKManager receives an Unregister request, it disposes of the OpenTelExtensionBean object and removes any event listener from it.

MakeCall Callout

When SampleBasicSDKManager receives a MakeCall request, it looks up the specified OpenTelExtensionBean object and invokes the makeCall API.

ScreenPop Event

ScreenPop event is sent when an OpenTel event of INBOUND_CALL and state RECEIVE is received.

CallEstablished Event

CallEstablised event is sent when an OpenTel event of TP_ANSWERED and state CONNECT is received. The primary call ID of the current call is retained.

CallReleased Event

CallReleased event is sent when an OpenTel event of state NULL is received and there is a current call.

8 Oracle Advanced Outbound Extension

This chapter covers the following topics:

- Advanced Outbound Extension Architecture
- Programming Languages
- Implementing AnalogExtensionDevice
- Implementing VduDevice
- Testing Advanced Outbound Extension
- Deploying Advanced Outbound Extension

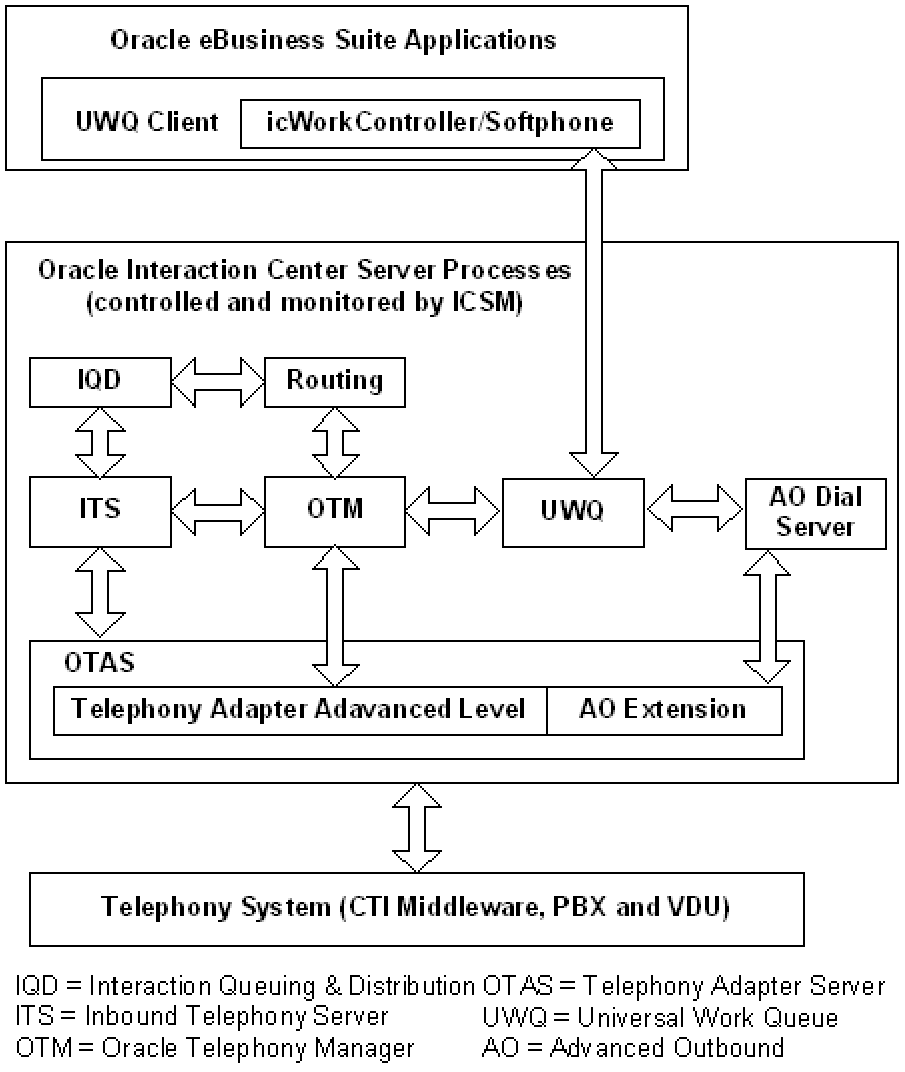
Advanced Outbound Extension Architecture

The Advanced Integration of Oracle Telephony Adapter SDK supports preview and progressive calls for Oracle Advanced Outbound. The Advanced Outbound Extension of Oracle Telephony Adapter SDK supports predictive calls in the Oracle Telephony Adapter Server framework. Predictive calls are dialed automatically by a predictive dialer, such as a voice detection unit (VDU), and transferred programmatically to an interaction center agent. Oracle Advanced Outbound requires access to analog extensions and VDU boards. The Oracle Telephony Adapter SDK Advanced Outbound Extension extends the Oracle Telephony Adapter Server framework to support development of telephony adapters that are capable of accessing these two types of devices.

The architecture of the Oracle Telephony Adapter SDK Advanced Outbound Extension consists of all the components that are required by the Advanced Integration level of Oracle Advanced Inbound plus the Advanced Outbound Dial Server. Advanced Outbound Dial Server is a server process that integrates list management and a predictive dialer. Oracle Telephony Adapter Server provides the Advanced Outbound Dial Server with the abstraction that is necessary to access and control predictively-dialed calls.

As the following diagram illustrates, in Advanced Integration, the Telephony Adapter Server is the intermediary between the telephony system and Oracle Advanced Inbound. Oracle Universal Work Queue is the intermediary between Oracle Advanced Inbound and the agent’s client system, including the icWork Controller and the softphone. The Oracle Telephony Adapter Server is extended to handle requests from Advanced Outbound Dial Server.

Advanced Outbound Extension Architecture

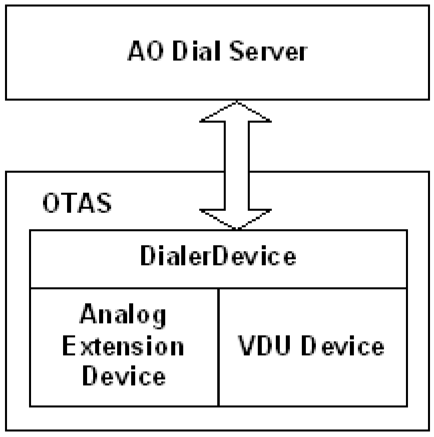

Oracle eBusiness Suite Applications
UWQ Client
icWorkController/Softphone
Oracle Interaction Center Server Processes (controlled and monitored by ICSM)
IQD
Routing
ITS
OTM
UWQ
AO Dial Server
OTAS
Telephony Adapter Adavanced Level
AO Extension
Telephony System (CTI Middleware, PBX and VDU)
IQD = Interaction Queuing & Distribution
ITS = Inbound Telephony Server
OTM = Oracle Telephony Manager
OTAS = Telephony Adapter Server
UWQ = Universal Work Queue
AO = Advanced Outbound

The Advanced Outbound Extension consists of two major devices:

- VDU: Predictive dialer hardware
- Analog Extension: Extension for analog lines. The analog extension is similar to the TelesetDevice but has only one line extension.

Oracle Telephony Adapter Server provides abstraction interfaces for these types of devices so that adapter developers can develop specific implementations of these devices to their target switch, middleware and VDU hardware. The Advanced Outbound Dial Server accesses these devices by using the dialer object hosted by Oracle Telephony Adapter Server. The following diagram illustrates the architecture.

Advanced Outbound Extension and Oracle Telephony Adapter Server



As the previous diagram illustrates, Advanced Outbound Dial Server communicates with AnalogExtensionDevice and VDUDevice through DialerDevice running in Oracle Telephony Adapter Server.

DialerDevice controls and receives events from AnalogExtensionDevice and VduDevice to perform the functions required by the Advanced Outbound Dial Server. DialerDevice is developed Oracle and is shipped with Oracle Advanced Inbound. Adapter developers are not required to develop DialerDevice.

The Advanced Outbound Extension adds the following to the Oracle Telephony Adapter SDK package.

- Java interface: Defined in package oracle.apps.cct.sdk.ao
- C header files: Located in directory oracle/apps/cct/c/ao

Programming Languages

Developers can implement Advanced Outbound Extension in the Java or C programming languages. The choice of programming language is determined by the underlying API provided by the third-party vendor of AnalogExtension and the VDU board. Typically, the AnalogExtension API is included in the same API that is necessary for Advanced Integration (that is, TelesetDevice and RoutePointDevice) while the VDU API is available separately.

Developers may implement Advanced Outbound Extension in the same programming language as the Advanced Integration or in a different programming language. For example, developers can implement Advanced Integration in Java and implement the Advanced Outbound Extension in C. The following table lists the supported combinations of programming languages and implementation types.

Supported Programming Languages and Implementation Types

Implementation Type	Description	Coding	Packaging
All Java	Implement all interfaces in Java	TeleDeviceFactory Impl implements TeleDeviceFactory, AnalogExtensionDeviceFactory, VduDeviceFactory	One JAR file
All C	Implement all interfaces in C	Implement all C functions	Two DLL files: one for Advanced Integration, one for Advanced Outbound Extension
Advanced Integration in Java, Advanced Outbound Extension completely in C	Implement TelesetDevice and RoutePointDevice in Java Implement AnalogExtensionDevice and VduDevice in C	TeleDeviceFactory Impl implements TeleDeviceFactory	One JAR file and one DLL file
VDU in C only	Implement TelesetDevice RoutePointDevice and AnalogExtensionDevice in Java Implement VduDevice in C	TeleDeviceFactory Impl implements TeleDeviceFactory, AnalogExtensionDeviceFactory	One JAR file and one DLL file

Advanced Integration is always packaged separately from Advanced Outbound Extension except for the all-Java implementations.

Implementing AnalogExtensionDevice

AnalogExtensionDevice consists of three interfaces:

- AnalogExtensionDeviceFactory: Factory interface that creates and destroys AnalogExtensionDevice objects
- AnalogExtensionDevice: Interface for all methods of Analog Device
- AnalogExtensionEventListener: Interface for sending events required by Analog Device

AnalogExtensionDeviceFactory

Developers must implement the AnalogExtensionDeviceFactory interface in their TeleDeviceFactory implementations. The TeleDeviceFactory implementation must implement both the TeleDeviceFactory and AnalogExtensionDeviceFactory interfaces. For example,

```
public class TeleDeviceFactoryImpl implements TeleDeviceFactory,
AnalogExtensionDeviceFactory
```

createAnalogExtensionDevice

Definition: Create an AnalogExtension object.

createAnalogExtensionDevice Parameters

Parameter Name	Type Java or C	Value
extension	String/const char*	Number of the analog extension

Returns: AnalogExtensionDevice object

destroyAnalogExtensionDevice

Definition: Destroy an AnalogExtension object.

Parameter Name: extension

Type Java or C: String/const char*

Value: Number of the analog extension

Returns: AnalogExtensionDevice object

init

Definition: Initialize AnalogExtensionDeviceFactory.

init Parameters

Parameter Name	Type Java or C	Value
config	Hashtable / OHashtable	Configuration parameters in a Hashtable

Returns: Not applicable

AnalogExtensionDevice

AnalogExtensionDevice is the interface that represents an extension for analog lines. AnalogExtensionDevice is similar to the TelesetDevice, but has only one line extension. It is used by the VDU to make outbound calls for Advanced Outbound Integration.

addAnalogExtensionEventListener

Definition: Add AnalogExtensionEventListener

addAnalogExtensionEventListener Parameters

Parameter Name	Type Java or C	Value
listener	AnalogExtensionEventListener, N/A	The listener to add

Returns: Not applicable

completeTransfer

Definition: Complete transfer.

Parameters: Not applicable

Returns: Not applicable

destroy

Definition: Destroy the AnalogExtensionDevice

Parameters: Not applicable

Returns: Not applicable

makeCall

Definition: Make a call.

makeCall Parameters

Parameter Name	Type: Java or C	Value
destNumber	String/const char*	The destination number to dial
appData	String/const char*	Application data

Returns: Not applicable

releaseCall

Definition: Release a call.

Parameters: Not applicable

Returns: Not applicable

removeAnalogExtensionEventListener

Definition: Remove AnalogExtensionEventListener

removeAnalogExtensionEventListener Parameters

Parameter Name	Type Java or C	Value
listener	AnalogExtensionEventListener / N/A	The listener to remove

Returns: Not applicable

transferScreened

Definition: Initialize a two-step transfer.

transferScreened Parameters

Parameter Name	Type Java or C	Value
destNumber	String / const char	Destination number to dial
appData	String / const char	Application data

Returns: Not applicable

transferUnscreened

Definition: Initialize a single-step transfer.

transferUnscreened Parameters

Parameter Name	Type: Java or C	Value
destNumber	String / const char *	Destination number to dial
appData	String / const char *	Application data

Returns: Not applicable

AnalogExtensionEventListener

Oracle Telephony Adapter Server internally implements AnalogExtensionEventListener. AnalogExtensionEventListener is associated with AnalogExtensionDevice by calling addAnalogExtensionEventListener and removeAnalogExtensionEventListener. Developers must keep a list of listeners in the AnalogExtensionDevice implementation and send events to Oracle Telephony Adapter Server by calling the method defined in AnalogExtensionEventListener.

callEstablishedEvent

Definition: Call established event.

Parameters: Not applicable

Returns: Not applicable

callReleasedEvent

Definition: Call released event

Parameters: Not applicable

Returns: Not applicable

callTransferredEvent

Definition: Call transferred event.

Parameters: Not applicable

Returns: Not applicable

Implementing VduDevice

VduDevice is the interface that represents predictive dialer hardware.VduDevice consists of three interfaces:

- VduDeviceFactory: Factory interface that creates and destroys VduDevice objects
- VduDevice: Interface for all methods of VDU Device
- VduEventListener: Interface for sending events required by VDU Device

Developers may implement these interfaces in Java or C. For Java implementations, developers must implement this interface in their TeleDeviceFactory implementations similar to AnalogExtensionDeviceFactory. For C implementations, developers must implement the function that is defined in the VduDevice.h and VduFactory.h header files and create a DLL. In these cases, VduDeviceFactory is not required by TeleDeviceFactory implementations.

> **Note:** You can implement VduDevice in C and implement the rest of the adapter in Java. The reason for this is that VDU is likely to be a different system than the CTI middleware with which the developer is integrating. For example, Java APIs may be available for the CTI middleware but not for the VDU board.

VduDeviceFactory

createVduDevice

Returns: Create a VduDevice object.

createVduDevice Parameters

Parameter Name	Type: Java or C	Value
vduId	String / const char*	Vdu ID
config	Hashtable / OHashtable	Configuration parameters in a Hashtable

Returns: Vdu Device object

destroyVduDevice

Definition: Destroy a VduDevice object and free any allocated resources.

destroyVduDevice Parameters

Parameter Name	Type: Java or C	Value
config	VduDevice / OcctVduDevice*	Object to be destroyed

Returns: Not applicable

init

Definition: Initialize VduDeviceFactory.

VduDeviceFactory Parameters

Parameters Name	Type: Java or C	Value
config	Hashtable / OHashtable	Configuration parameters in a Hashtable

Returns: Not applicable

VduDevice

addVduDeviceListener

Definition: Add VduEventListener

addVduDeviceListener

Parameter Name	Type: Java or C	Value
listener	VduEventListener / N/A	Listener to add

Returns: Not applicable

close

Definition: Close the Vdu port.

Parameters: Not applicable

Returns: Not applicable

dropCall

Definition: Drop the current call on the VDU port, and stop call progress detection.

Parameters: Not applicable

Returns: Not applicable

makeCall

Definition: Make a call through this VDU port. This call sets the port offHook before making the call.

makeCall Parameters

Parameter Name	Type: Java or C	Value
destNumber	String / const char*	Destination number, if null, this call will do call progress detection only
cpd	boolean / oboolean	Call progress detection. If false, all followed parameters are ignored.
sit	boolean / oboolean	STI tone detection
amd	boolean / oboolean	Answering machine detection
amdMethod	int / oint	Answering machine detection method
narc	int / oint	No answer ring count

Returns: Not applicable

offHook

Definition: "offhook" the associated analog extension. After successfully executing this method, the extension remains in the "offhook" state for one or two seconds.

Parameters: Not applicable

Returns: Not applicable

open

Definition: Open the Vdu port.

open Parameters

Parameter Name	Type: Java or C	Value
portNumber	int / oint	vdu port number
vduType	int / oint	vdu type
voiceBoard	int / oint	voice board number
dtiBoard	int / oint	dti board number

Returns: Not applicable

playMessage

Definition: Play voice message.

***playMessage* Parameters**

Parameter Name	Type: Java or C	Value
fileName	String / const char*	Message file name
repeatCount	int / oint	Repeat count

Returns: Not applicable

removeVduEventListener

Definition: Remove VduEventListener.

***removeVduEventListener* Parameters**

Parameter Name	Type: Java or C	Value
listener	VduEventListener / N/A	Listener to remove

Returns: Not applicable

setConfiguration

Definition: Update configuration.

***setConfiguration* Parameters**

Parameter Name	Type: Java or C	Value
config	Hashtable / OHashtable	Configuration parameters in a Hashtable

Returns: Not applicable

withdrawCall

Definition: Withdraw the current call on the VDU port. If an outcome had already been reached, then "withdraw" does nothing.

Parameters: Not applicable

Returns: Not applicable

VduEventListener

VduEventListener is implemented internally by Oracle Telephony Adapter Server. VduEventListener is associated with VduDevice by calling addVduEventListener and removeVduEventListener.

Developers must keep a list of listeners in the VduDevice implementation and send events to Oracle Telephony Adapter Server by calling the method that is defined in VduEventListener.

callOutcomeEvent

Definition: Event reported by Vdu device.

callOutcomeEvent Parameters

Parameter Name	Type: Java or C	Value
vduState	int / oint	vdu state
callOutcome	int / oint	call outcome

Returns: Not applicable

Testing Advanced Outbound Extension

The Advanced Outbound Extension has two testing methods:

- SDK Integrated Test Utility
- Oracle eBusiness Suite Advanced Outbound

The TeleDevice component of the SDK Integrated Test Utility provides a user interface for creating and controlling Dialer devices. Developers can use the Dialer device UI for invoking Dialer methods and monitor Dialer Events.

Developers should set up an environment for Oracle Advanced Outbound and perform integration testing.

Related Topics

Oracle Advanced Outbound Implementation Guide

Deploying Advanced Outbound Extension

Implementations that are done exclusively in Java require no extra steps to deploy the Advanced Outbound Extension because the Adapter Jar file contains the extension. For partial Java and C implementations, perform the following procedures:

- Package the C implementation in a DLL file, such as ao_extension.dll.
- Copy the file ao_extension.dll to the third-party directory where Interaction Center Server Manager is installed and where Oracle Telephony Adapter Server will run.
- Update the Server Option of the target Oracle Telephony Adapter Server with -vdu_dll ao_extension.dll.

9

Business Application Considerations

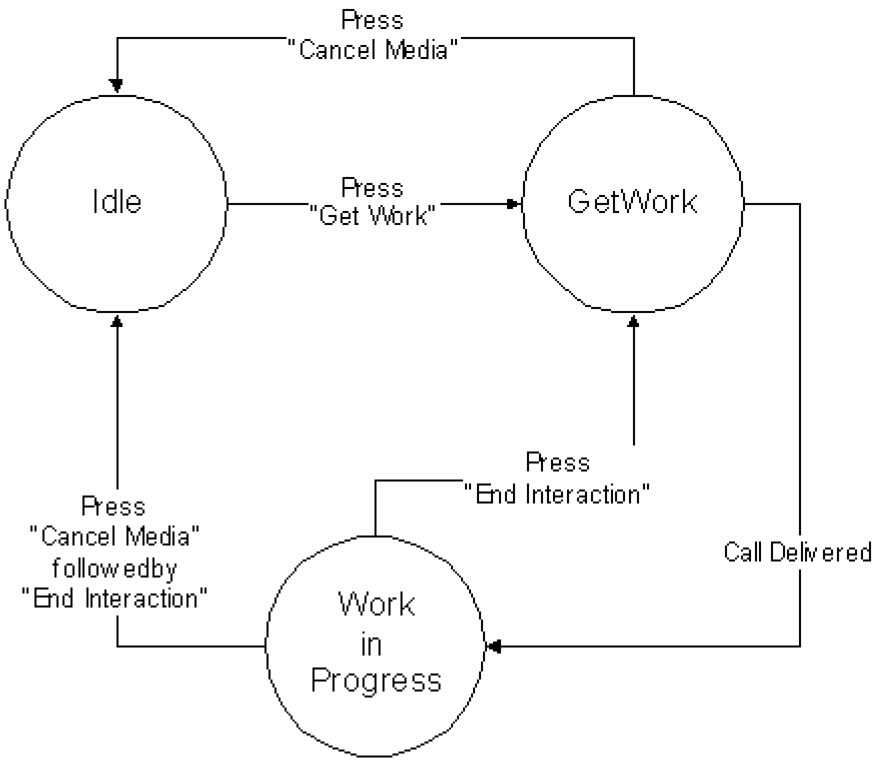
This chapter covers the following topics:

- Special Ready/Not Ready State Requirement for Oracle Universal Work Queue Get Work Model

Special Ready/Not Ready State Requirement for Oracle Universal Work Queue Get Work Model

Some special considerations are necessary when implementing Oracle Telephony Adapter SDK with certain Oracle eBusiness Suite Applications. Refer to the following instructions whenever the implementation involves any of the applications listed in this section.

When an agent logs in to Oracle Universal Work Queue, the agent is put into an Idle state in which the agent receives only unsolicited calls, such as directly transferred calls. The agent can change from the Idle state to the Get Work state by selecting Get Work. In the Get Work state, the agent is available for the next work item, and remains in this state until receiving a work item, such as an inbound call. When the agent receives the call, the state is changed to Work in Progress. The Work in Progress state is similar to Idle in that the agent receives only unsolicited work. From the Work in Progress state, the agent can either change to Idle by first using Cancel Media and then End Interaction, or change to the Get Work state by using End Interaction. While in the Get Work state, if the agent completes the state change before receiving a work item, then the agent can change to Idle by using Cancel Media.



Oracle Universal Work Queue requires that when an inbound call is released, the agent must not receive another inbound call until the agent selects End Interaction / Next Call. This requirement can be fulfilled in one of the following ways.

Programmatic

For ringing or established events, set the agent to NotReady (or an equivalent state), while making sure that the call does not get hung up because of an agent state change. Make sure the agent remains in NotReady (or an equivalent state) after the call is released.

Switch and Middleware Configuration

Configure the switch and middleware to automatically put an agent to NotReady (or an equivalent state) immediately after a call is released. Make sure the agent is *not* automatically set back to Ready after a specified timeout. The switch AutoReady is not compatible with the Oracle Universal Work Queue GetWork model.

Combined

Configure the switch and middleware combination to automatically put an agent in the NotReady state (or an equivalent state) immediately after a call is released. If this switch and middleware feature also automatically sets the agent back to Ready after a specified timeout, then the adapter code should also set the agent to NotReady (or an equivalent state) on released events so that the timer is canceled and the agent is *not* automatically set back to Ready.

10
Diagnostics and Troubleshooting

This chapter covers the following topics:

- Upgrading from Release 11.5.6 or 11.5.7
- SDK Integrated Test Utility
- Verification Tool
- Oracle Telephony Adapter Server Logging
- Oracle Telephony Adapter Server Startup Sequence
- Database Layer
- Socket Layer
- Adapter Layer
- Successful Startup
- Tracking SDK APIs and Events
- SDK API Traces
- SDK Exception Traces
- SDK Event Traces
- Sample Traces
- Interruption of OTAS and CTI Connection
- Common Adapter Errors
- Compiling Sample Code

Upgrading from Release 11.5.6 or 11.5.7

This document describes how to troubleshoot Oracle Advanced Inbound implementations using Oracle Telephony Adapter SDK.

Telephony SDK is part of Oracle Application Release 11.5.8 (Interaction Center Family Pack O).

When upgrading from Release 11.5.6 or 11.5.7 to Release 11.5.8 SDK, review the following documents:

- Metalink Note: 204766.1: Oracle Advanced Inbound Feature Support Changes for Oracle Interaction Center Family Pack-O

- MetaLink Note 204767.1: Oracle Advanced Inbound Certified Switches/Middleware for Oracle Interaction Center Family Pack-O
- Interaction Center Family Pack O 2374818 Readme
- Upgrade Patch 2435611 Readme: 115.6 to 11.5.8 SDK UPGRADE

SDK Integrated Test Utility

The following tables lists common issues in using the SDK Integrated Test Utility and recommended actions for resolving these issues.

SDK Integrated Test Utility

Common Issues	Recommended Action
SDK Test Utility does not start.	Check JDK path and SDK_JRE_HOME and SDK_JRE. environment variable in sdkenv.cmd. Make sure no spaces are in the PATH or CLASSPATH environment variable.
Oracle Telephony Adapter Server does not start.	Refer to Oracle Telephony Adapter Server Startup Sequence for more details.
Verification Tool does not start.	Check Verification Tool configuration. Make sure Oracle Telephony Adapter Server is running before Verification Tool.

Verification Tool

The Verification Tool is a development tool to automate Oracle Telephony Adapter SDK API testing. The Verification Tool's main use is to run SDK adapter test cases and to generate outputs easily. The scripts and solution files are written for a limited set of switches and middlewares. The supplied solution files do not cover all possible valid outcomes. Therefore, an adapter that passes all verifications might not necessarily work properly when integrated with Oracle Interaction Center. Likewise, an adapter that fails to pass some verifications might not necessarily work when integrated with Oracle Interaction Center.

Note: The Verification Tool is not a certification tool. Do not use the Verification Tool as the final check for Adapter implementation or for live customer diagnostics. Adapter developers should always run and test their implementations fully integrated with Interaction Center before going into production. Support should always use Oracle Telephony Adapter Server logging as the diagnostic tool.

Oracle Telephony Adapter Server Logging

Oracle Telephony Adapter Server logs runtime and diagnostics information into a log file. You can configure the content and the level of details logged in the file. This section describes how to configure and access Oracle Telephony Adapter Server logs.

Accessing Oracle Telephony Adapter Server Logs

Every time Oracle Telephony Adapter Server starts, it creates a log file in the directory <ICSM_TOP>/admin/scripts/<SERVER_NAME>, in which ICSM_TOP is the path where Interaction Center Server Manager is installed, and SERVER_NAME is the server name of Oracle Telephony Adapter Server that is defined in the Interaction Center Server Manager HTML Administration.

The Oracle Telephony Adapter Server log file has the naming convention

`<OTAS_server_name>mmddyyyy_hhMMss.log`

where " mmddyyyy" is the date when the log file is created, and "hhMMss" is the time when the log file is created.

Use the following procedure to view and download Oracle Telephony Adapter Server log files by way of the Interaction Center Server Manager HTML Administration.

Log In

HTML Login URL

Responsibility

Interaction Center Server Manager

Prerequisites

Create a user in Interaction Center Server Manager HTML Administration.

Steps

1. Log in to the Interaction Center Server Manager HTML Admin.
2. Locate the Server Group / Node where the target Oracle Telephony Adapter Server is running.
3. Select **OTAS**.
4. Go to Server Details > Advanced.

   The Advanced page opens.
5. View and download any of the listed log files.

   Note: You can view log files when Interaction Center Server Manager is running on the target node. When running Oracle Telephony Adapter Server by using the SDK Integrated Test Utility, the Oracle Telephony Adapter Server log file is in the directory that is specified in the SDK test utility.

You can also access Oracle Telephony Adapter Server log files locally on the node where Oracle Telephony Adapter Server runs.

Configuring Oracle Telephony Adapter Server Logs

Use the following procedure to configure Oracle Telephony Adapter Server to log different levels of detail.

Log In

HTML Login URL

Responsibility

Interaction Center Server Manager

Prerequisites

Create a user in Interaction Center Server Manager HTML Administration.

Steps

1. Locate the Server Group / Node where the target Oracle Telephony Adapter Server is running.
2. Select **ICSM**.
3. Go to Server Details > Server Parameter page.
4. Edit the Trace Level Parameter.

Oracle Telephony Adapter Server classifies different log entries into modules and levels of details. You can configure Oracle Telephony Adapter Server to log all modules with a specific level of details or log only specific modules with a specific level of details. Each log entry has the following format:

[timestamp] - [module: level] - message

For example,

```
[Fri Jul 19 12:18:05 PDT 2002] - [adapter:verbose] -
SampleTeleDeviceFactory.createTelesetDevice(7001, 8001, 9001);
```

A message could span multiple lines, as in the following example.

```
[Fri Jul 19 12:18:06 PDT 2002] - [device:info] - [API]

device = TelesetDevice[7001,8001,9001]

object = oracle.apps.cct.sdk.sample.SampleTelesetDevice@1ece00

method = loginAgent

acdAgentId = 7001

acdAgentPassword = 8001

acdQueue = 9001
```

Log Detail Levels

The following table lists the level of details that are supported by Oracle Telephony Adapter Server, in the order of descending severity.

Log Levels

Level	Meaning
fatal	Fatal and unrecoverable error, Oracle Telephony Adapter Server will shutdown
error	Major error, affects some Oracle Telephony Adapter Server functions
warning	Minor error, does not affect functionality
info	Informational only, for diagnostics purpose
verbose	Extended diagnostics information, use only when instructed by development

The following table lists the log modules that are defined in Oracle Telephony Adapter Server.

:

Log Modules

Module	Meaning
otas	Generic Oracle Telephony Adapter Server messages
comm	Communication layer messages
db	Database access layer messages
adapter	Messages generated by Telephony Adapter
device	Telephony API and Events diagnostics messages

Trace Level Format

By default, all modules are set to error level, so that logging includes only messages that are tagged at the error level or a higher level (error + fatal). Users can change the level by using the Trace Level server parameter.

The Trace Level server parameter uses the following format. Users can globally set the level for all modules or set the level individually.

- <level> or
- <module_1>=<level_1>,<module_2>=<level_2>,...

The following table lists and describes trace levels.

Trace Levels

Trace Level	Description
	(Default) Log error and fatal level messages for all modules
fatal	Log only fatal level messages for all modules
error	Log error and fatal level messages for all modules
warning	Log warning, error and fatal level messages for all modules
info	Log informational, warning, error and fatal level messages for all modules
device=info	• Log info level messages for device • Log error level messages for all others
device=info,adapter=verbose,otas=warning	• Log info level messages for device • Log verbose level messages for adapter • Log warning level messages for Oracle Telephony Adapter Server • Log error level messages for all others

Common and Recommended Usage

- info: Turn on everything from fatal to informational messages for all modules.
- device=info: For tracing Adapter API and Events while filtering out informational messages from other modules.
- device=info,adapter=verbose: For tracing Adapter API and Events and any internal messages generated by Adapter.

General Guidelines in Interpreting Oracle Telephony Adapter Server Logs

- Identify fatal and error messages.
- Info and verbose messages are not serious, even for an exception trace.

Oracle Telephony Adapter Server Log Header

The Oracle Telephony Adapter Server Log Header contains information about the release and build number and startup sequence information. This information is very useful in determining the release, patch level and adapter configuration at a particular site. The Oracle Telephony Adapter Server Log Header contains the following.

> **Note:** The following example shows the default logging configuration. If the log level is set to a higher error level, then the log displays more entries than those shown here.

- Server Wrapper Build and Release Information indicates the version of Interaction Center Server Manager.

```
[Tue Jul 23 14:37:15 PDT 2002] -

***************************************

Server Wrapper - Release 11.5.8

Build 219 on Mon Jul 15 15:54:52 PDT 2002

(c) Copyright 2002 Oracle Corporation. All rights reserved.

***************************************
```

- Oracle Telephony Adapter Server Build and Release Information indicates the version of Oracle Telephony Adapter Server.

```
[Tue Jul 23 14:37:17 PDT 2002] -

***************************************

Oracle Telephony Adapter Server - Release 11.5.8

Build 2547 on Thu Jul 18 18:36:26 PDT 2002

(c) Copyright 2002 Oracle Corporation. All rights reserved.

***************************************
```

Startup Sequence Information shows users which steps Oracle Telephony Adapter Server has gone through in startup. If there is an error in a particular step, then an error is logged and Oracle Telephony Adapter Server will not start. Adapter information is logged in this step:

```
[Tue Jul 23 14:37:18 PDT 2002] - Database Layer started

[Tue Jul 23 14:37:18 PDT 2002] - Socket Layer started

[Tue Jul 23 14:37:18 PDT 2002] - Adapter Layer started
```

The final status indicates that Oracle Telephony Adapter Server has been started successfully:

```
[Tue Jul 23 14:37:18 PDT 2002] - Oracle Telephony Adapter Server
[xxx_otas] started.
```

Oracle Telephony Adapter Server Startup Sequence

Oracle Telephony Adapter Server starts in the following sequence:

1. Database Layer: Establish connection to database and load configurations
2. Socket Layer: Create server TCP/IP socket for communication
3. Adapter Layer: Load and Initialize Adapter

If any part of the phase fails, then a fatal error is logged and Oracle Telephony Adapter Server shuts down.

Database Layer

The database layer phase establishes the database connection. Oracle Telephony Adapter Server executes this phase in the following order:

1. Load dbc file (passed by Interaction Center Server Manager).
2. Create database connection.
3. Load Server Configurations.
4. Load Middleware Configurations.
5. Update Server Status.

If any of the above procedures fail, Oracle Telephony Adapter Server logs the following:

```
[Tue Jul 23 15:33:04 PDT 2002] - Database Layer starting FAILED


[Tue Jul 23 15:33:04 PDT 2002] - [otas:fatal] - Nested Exception
:


< exception stack trace based on different errors >
```

The following table lists possible reasons for failure and describes recommended actions.

Database Layer Recommended Action

Reason	Recommended Action
dbc file does not exist (should not happen in general if Interaction Center Server Manager is running)	Check DBC file integrity
dbc file error (should not usually happen when Interaction Center Server Manager is running)	Check DBC file integrity
database unavailable	Contact Database Administrator
server configuration error	Check Server Configuration in Interaction Center Server Manager HTML Admin
middleware configuration error (missing Middleware Configuration Name in Server Parameter)	Check Server Configuration in Interaction Center Server Manager HTML Admin and Middleware Configuration in Call Center HTML Admin

If the database layer phase succeeds, Oracle Telephony Adapter Server logs the following message:

```
[Tue Jul 23 15:33:04 PDT 2002] - Database Layer started
```

Socket Layer

The socket layer phase creates a TCP/IP server socket. Oracle Telephony Adapter Server executes this phase in the following order.

1. Get the local IP address.
2. Create a server socket.
3. Update database with the port information.

If any of the step above fails, then Oracle Telephony Adapter Server will log the following message:

```
[Tue Jul 23 15:33:04 PDT 2002] - Socket Layer starting FAILED


[Tue Jul 23 15:33:04 PDT 2002] - [otas:fatal] - Nested Exception
:


< exception stack trace based on different errors >
```

The following table lists possible reasons for failure and describes recommended actions.

Socket Layer Recommended Action

Reason	Recommended Action
Network error	Contact Network Administrator
Unable to create server socket (address in use)	Check if -svr_port is used, contact network administrator if necessary
Unable to update database records	Contact Database Administrator

If the socket layer phase succeeds, Oracle Telephony Adapter Server logs the following message:

```
[Tue Jul 23 15:33:04 PDT 2002] - Socket Layer started
```

Adapter Layer

In the adapter layer phase, Oracle Telephony Adapter Server loads and initializes the Adapter in the following order:

1. Get Middleware Type and PBX Type from Middleware Configuration.
2. Derive the TeleDeviceFactory Java class name to be loaded.
3. Load the TeleDeviceFactory Java class.
4. Initialize the Adapter by invoking init() method.

If any of the above procedures fails, then Oracle Telephony Adapter Server logs the following message:

```
[Tue Jul 23 15:33:04 PDT 2002] - Adapter Layer starting FAILED


[Tue Jul 23 15:33:04 PDT 2002] - [otas:fatal] - Nested Exception
:


< exception stack trace based on different errors >
```

The following table lists possible reasons for failure and describes recommended actions.

Adapter Layer Recommended Action

Reason	Recommended Action
Wrong Middleware Configuration	Check Middleware Configuration
Unable to derive TeleDeviceFactory Java class name from Middleware Configuration	Check Middleware Configuration
Unable to load the TeleDeviceFactory Java class (class does not exist in clasps)	Check if adapter jar file exists in 3rdParty directory
Unable to instantiate the TeleDeviceFactory Java class (no default constructor)	Fix adapter source code
Unable to instantiate the TeleDeviceFactory Java class (constructor not public)	Fix adapter source code
TeleDeviceFactory Java class throws TeleDeviceException on init() (Runtime error checked by Adapter)	Check Middleware Configuration, Check Adapter Log, Consult Adapter Developer
Unable to load 3rd Party Java classes or libraries (NoClassDefFoundError)	Check if the 3rd Party libraries is placed in the 3rdParty directory

If the adapter layer phase succeeds, Oracle Telephony Adapter Server logs the following message:

```
[Tue Jul 23 15:33:04 PDT 2002] - Adapter Layer started
```

In the Adapter Phase, Oracle Telephony Adapter Server logs additional information about the type of adapter that is being configured, as shown in the following example.

```
[Tue Jul 23 15:21:00 PDT 2002] -


Middleware type =  ICC_CUSTOM_ADAPTER


PBX Type =  null


TeleDeviceFactory Class =  oracle.apps.cct.sdk.sample.SampleTeleD
eviceFactory


[Tue Jul 23 15:21:00 PDT 2002] -


Integration Level = <Advanced or Standard>
```

Based on the Middleware type and PBX type, users can determine whether the Oracle Telephony Adapter Server is running a custom adapter or an adapter that Oracle developed.

The following table lists and describes possible middleware types.

Possible Middleware Types

Middleware Type in Log File	Description
ICC_CUSTOM_ADAPTER	Custom Java Adapter
ICC_CUSTOM_C_ADAPTER	Custom C Adapter
ICC_CTC_ADAPTER (In-house)	Adapter for CT Connect
ICC_ASPECT_ADAPTER (In-house)	Adapter for Aspect
ICC_ICM_ADAPTER (In-house)	Adapter for Cisco ICM

The following table lists and describes possible PBX types (PBX types are applicable only to in-house adapters).

Possible PBX Types

PBX Type in Log File	Description
A	LUCENT_DEFINITY
C	ALCATEL_4400
E	ERICSSON_MD110
M	NORTEL_MERIDIAN
P	ASPECT
R	ROCKWELL_SPECTRUM
S	SIEMENS_HICOM
X	NORTEL_DMS100
Z	CALLMANAGER

Note: A value defined here does not indicate that switch certification is available for the platform. Please consult the Switch Certification support matrix for details.

Successful Startup

If Oracle Telephony Adapter Server starts successfully, then it logs the following message:

```
[Tue Jul 23 14:37:18 PDT 2002] - Oracle Telephony Adapter Server
[ <otas_server_name> ] started.
```

Tracking SDK APIs and Events

Oracle Telephony Adapter Server logs all API invoked, Events sent and Exceptions thrown by the Adapter at the info level of the device module. To enable this feature, set Trace Level to one of the following:

- info
- device=info

Oracle recommends that you use device=info so that log entries of other modules will not show.

SDK API and Event traces are useful for diagnosing Adapter behavior. When they are used in conjunction with the call event flow specification of the Telephony SDK, users will be able to determine if the Adapter is working properly.

SDK API Traces

SDK API Traces are Adapter methods, such as makeCall, that Interaction Center Servers invoke. Oracle Telephony Adapter Server logs the device, the method that is invoked and the parameters that are passed to that method.

SDK API Traces have the following format:

```
[<timestamp>] - [device:info] - [API]


device =  TelesetDevice[<extension1,extension2,extension3>] or Ro
utePointDevice[<extension>]


object =  <teledevice object>


method = <method name>


<parameter1 = value1 >


<parameter2 = value2 >


...
```

Example Log Entry

```
[Wed Jul 24 16:45:48 PDT 2002] - [device:info] - [API]


 device =  TelesetDevice[7001,8001,9001]


 object =  oracle.apps.cct.sdk.sample.SampleTelesetDevice@1ed043
 method = blindTransfer

mediaItemId =  7001

ineIndex =  8001

destinationNumber =  9001
```

SDK Exception Traces

SDK Exception Traces are TeleDeviceExceptions that the Adapter throws. The log records both the device that throws the exception and the exception details.

SDK Exception Traces have the following format:

```
[<timestamp>] - [device:info] - [EXCEPTION]

device =  TelesetDevice[<extension1,extension2,extension3>] or Ro
utePointDevice[<extension>]

exception :  <exception details>
```

Example Log Entry

```
[Wed Jul 24 16:45:48 PDT 2002] - [device:info] - [EXCEPTION]

device =  TelesetDevice[7001,8001,9001]

exception :  TeleDeviceException:ERROR_CODE_FUNCTION_NOT_SUPPORTE
D;blindTransfer NOT IMPLEMENTED
```

SDK Event Traces

SDK Event Traces are events that the Adapter sends. The log records both the device that sends the event and the event details.

SDK Event Traces have the following format:

```
[<timestamp>] - [device:info] - [EVENT]

device =  TelesetDevice[<extension1,extension2,extension3>] or Ro
utePointDevice[<extension>]
```

```
object =  <teledevice object>

event = <event name>

<parameter1 = value1 >

<parameter2 = value2 >

...
```

Example Log Entry

```
[Wed Jul 24 17:41:39 PDT 2002] - [device:info] - [EVENT]

device =  TelesetDevice[7002,8002,9002]

object =  oracle.apps.cct.sdk.sample.SampleTelesetDevice@1eca40

event = beginCallEvent

mediaItemId =  7002

mediaItemIdComplete =  true

ineIndex =  1

ani =  7001

dnis =  8002

data =  {MI=7002}

dataComplete =  true
```

Sample Traces

DEFINITION OF SAMPLE TRACES

Creation of TelesetDevice

```
[Wed Jul 24 17:41:24 PDT 2002] - [device:info] - [API]


createTelesetDevice


extension1 =  7001


extension2 =  8001


extension3 =  9001
```

Destruction of TelesetDevice

```
[Wed Jul 24 17:45:08 PDT 2002] - [device:info] - [API]


 destroyTeleDevice


 device =  T:7001
```

Simple Internal Call Scenario

The following table lists the complete SDK API and Event traces for a simple internal call scenario in which A calls B, B answers, and A hangs up.

Simple Internal Call SDK API and Trace Events

Sequence	Notes	TelesetDevice A: TelesetDevice[7001, 8001,9001]	TelesetDevice B: TelesetDevice[7002, 8002,9002]
1	A calls B on 8002 using line 0	**[Wed Jul 24 17:41:39 PDT 2002] - [device:info] - [API]**device = TelesetDevice[7001, 8001,9001]object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1ec457**method = makeCall**media ItemId = 7002lineIndex = 0destinationNumber = 8002	
2	B sends beginCallEvent on line 1		[Wed Jul 24 17:41:39 PDT 2002] - [device:info] - [EVENT]device = TelesetDevice[7002, 8002,9002] object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1eca40 event = beginCallEventmedia ItemId = 7002 media ItemIdComplete = true lineIndex = 1ani = 7001 dnis = 8002 data = {MI= 7002}dataComplete = true
3*	B sends callRingingEvent on line 1		[Wed Jul 24 17:41:39 PDT 2002] - [device:info] - [EVENT]device = TelesetDevice[7002, 8002,9002] object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1eca40 event = callRingingEventline Index = 1

Sequence	Notes	TelesetDevice A: TelesetDevice[7001, 8001,9001]	TelesetDevice B: TelesetDevice[7002, 8002,9002]
4	A sends beginCallEvent on line 0	[Wed Jul 24 17:41:39 PDT 2002] - [device:info] - [EVENT] device = TelesetDevice[7001, 8001,9001] object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1ec457 event = beginCallEvent mediaItemId = 7002 mediaItem IdComplete = true lineIndex = 0 ani = 7001 dnis = 8002 data = {MI=7002} dataComplete = true	
5*	A sends callDialingEvent on line 0	[Wed Jul 24 17:41:39 PDT 2002] - [device:info] - [EVENT] device = TelesetDevice[7001, 8001,9001]object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1ec457event = callDialingEventline Index = 0	
6	B answers call on line 1		[Wed Jul 24 17:41:56 PDT 2002] - [device:info] - [API]device = TelesetDevice[7002, 8002,9002] object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1eca40method = answerCallline Index = 1
7*	B sends callEstablishedEvent on line 1		[Wed Jul 24 17:41:56 PDT 2002] - [device:info] - [EVENT]device = TelesetDevice[7002, 8002,9002]object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1eca40event = callEstablishedEventline Index = 1

Sequence	Notes	TelesetDevice A: TelesetDevice[7001, 8001,9001]	TelesetDevice B: TelesetDevice[7002, 8002,9002]
8*	A sends callEstablishedEvent on line 0	[Wed Jul 24 17:41:56 PDT 2002] - [device:info] - [EVENT]device = TelesetDevice[7001, 8001,9001]object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1ec457event = callEstablishedEventline Index = 0	
9	A releases call on line 0	[Wed Jul 24 17:42:03 PDT 2002] - [device:info] - [API] device = TelesetDevice[7001, 8001,9001] object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1ec457 method = releaseCall lineIndex = 0	
10*	A sends callReleasedEvent on line 0	[Wed Jul 24 17:42:03 PDT 2002] - [device:info] - [EVENT] device = TelesetDevice[7001, 8001,9001] object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1ec457 event = callReleasedEvent lineIndex = 0	
11*	B sends callReleasedEvent on line 1		[Wed Jul 24 17:42:03 PDT 2002] - [device:info] - [EVENT] device = TelesetDevice[7002, 8002,9002] object = oracle.apps. cct.sdk.sample. SampleTelesetDevice@1eca40 event = callReleasedEvent lineIndex = 1

*The following sequences can be of any order: 3 and 5, 7 and 8, 10 and 11.

Simple Inbound Call Scenario

For inbound calls that reach the realm of monitored devices for the first time, the adapter should fire a beginCallEvent with the following attributes:

- mediaItemId should be left blank.
- mediaItemIdComplete should be set to true.

For implementing Oracle Advanced Inbound in enhanced passive mode only - A two-second delay is required to keep inbound calls at the route point before the calls are routed to agents. This delay is necessary to allow time for Oracle to assign a mediaItemId to an inbound call and notify the RoutePointDevice in the assignMediaItemId method. You may implement this delay by adding a wait step in the call processing script of the PBX/ACD or CTI middleware.

The adapter must keep track of the mediaItemId assigned by Oracle for each call and pass the same mediaItemId in all events for the entire life cycle of the call, including transferred and conferenced calls.

RoutePointDevice should not fire duplicated beginCallEvent and callQueueEvent for the same inbound call.

The following table describes a sample trace scenario for a simple inbound call for implementing Oracle Advanced Inbound Telephony in enhanced passive mode.

Sequence	Description	Events

1	An inbound call arrives at a route point.	`.`
		`[Fri Jun 25 14:39:42 PDT 2004] - [device:info]`
		`device =  RoutePointDevice[7710]`
		`object =`
		`oracle.apps.cct.sdk.sample.advanced.SampleRout`
		`event = beginCallEvent`
		`mediaItemId =`
		`mediaItemIdComplete =  true`
		`callId =  61696`
		`ani =  6501112222`
		`dnis =  7710`
		`data =  {AccountCode=12345}`
		`dataComplete =  true`
		`.`
		`.`
		`[Fri Jun 25 14:39:42 PDT 2004] - [device:info]`
		`device =  RoutePointDevice[7710]`
		`object =`
		`oracle.apps.cct.sdk.sample.advanced.SampleRout`
		`event = callQueuedEvent`
		`callId =  61696`
		`.`

Sequence	Description	Events
2	Oracle assigns a mediaItemId to the call.	
3	The call is routed to an agent.	

Sequence 2 events:

```
.
[Fri Jun 25 14:39:42 PDT 2004] - [device
device =  RoutePointDevice[7710]
object =  oracle.apps.cct.ccc.commonLaye
75e0
method = assignMediaItemId
mediaItemId =  {MI-11380760;}
callId =  61696
.

.[Fri Jun 25 14:39:42 PDT 2004] - [devic
device =  RoutePointDevice[7710]
object =
oracle.apps.cct.sdk.sample.advanced.Samp
event = updateMediaItemIdEvent
callId =  61696
mediaItemId =  {MI-11380760;}
.
```

Sequence 3 events:

```
.[Fri Jun 25 14:39:46 PDT 2004] - [devic
device =  RoutePointDevice[7710]
object =
oracle.apps.cct.sdk.sample.advanced.Samp
event = callDivertedEvent
callId =  61696
.
.
[Fri Jun 25 14:39:46 PDT 2004] - [device
d-18]
- [EVENT]
device =  TelesetDevice[7001,8001,9001]
object =
oracle.apps.cct.sdk.sample.advanced.Samp
event = beginCallEvent
mediaItemId =  {MI-11380760;}
mediaItemIdComplete =  true
lineIndex =  0
ani =  6501112222
dnis =  7710
data =  null
dataComplete =  true
.
.
[Fri Jun 25 14:39:46 PDT 2004] - [device
d-18]
- [EVENT]
device =  TelesetDevice[7001,8001,9001]
object =
oracle.apps.cct.sdk.sample.advanced.Samp
event = callRingingEvent
lineIndex =  0
.
```

Sequence	Description	Events

4	The agent answers the call.	`.`
		`[Fri Jun 25 14:40:00 PDT 2004] - [device:info]`
		`d-18]`
		`- [EVENT]`
		`device =  TelesetDevice[7001,8001,9001]`
		`object =`
		`oracle.apps.cct.sdk.sample.advanced.SampleTele`
		`event = callEstablishedEvent`
		`lineIndex =  0`
		`.`

5	The call is released.	`.`
		`[Fri Jun 25 14:43:35 PDT 2004] - [device:info]`
		`d-18]`
		`- [EVENT]`
		`device =  TelesetDevice[7001,8001,9001]`
		`object =`
		`oracle.apps.cct.sdk.sample.advanced.SampleTele`
		`event = callReleasedEvent`
		`lineIndex =  0`

Interruption of OTAS and CTI Connection

When the underlying connection between Oracle Telephony Adapter Server and the CTI middleware or switch is interrupted, you can expect the following behavior from the soft phone.

When the connection terminates, the soft phone displays an appropriate message, such as "Connection to telephony services is lost," and the soft phone buttons become disabled.

When the connection is reestablished, the soft phone displays an appropriate message such as "Connection to telephony services is restored," and the soft phone buttons are enabled. The soft phone does not reconnect with the physical telephone. The agent must complete any active calls manually by using the physical telephone.

If the expected messages do not appear, you can implement the custom adapter to trigger the appropriate soft phone behavior by doing the following:

- When a node goes down, throw a TeleDeviceException passing SWITCH_CONNECTION_LOST as the ErrorCode.
- When a node recovers or when the device is reconnected to another node, throw a TeleDeviceException passing SWITCH_RECONNECT_SUCCESS as the ErrorCode.

Common Adapter Errors

The following table lists common adapter errors that adapter implementers sometimes make, and the symptoms of those errors.

Common Adapter Errors

Adapter Error	Observed Symptom
SDK method not implemented	Softphone functions do not work.
SDK call events not properly implemented	Softphone out of synchronization.
RoutePoint Events not properly implemented	• Queue Count does not work in Enhanced Passive mode. • Missing screen pop data. • Reporting shows more than one media items for a single call. • Call data transfer does not work.
Media Item ID not properly propagated	
Memory not properly managed in C/C++ implementation	Oracle Telephony Adapter Server crashes.

Compiling Sample Code

Issue

Unable to compile Sample code.

Incompatible versions of Microsoft Visual Studio can cause problems when making a build by using the workspace file. To correct this issue, build the SampleAdapter.dll using the workspace (.dsw) file in the directory \oracle\apps\cct\c\sample.

Use the following procedure to build the SampleAdapter.dll.

1. Add only the following source files to the project.
    1. TeleDeviceFactory.c
    2. RoutePointDevice.c
    3. TelesetDevice.c
2. In Project Settings > Resource tab, add the following directory in the 'Additional resource include directories' field ../include, or the whole path to the include directory, such as D:\sdk\oracle\apps\cct\c\include.

A
SDK Scope Analysis

This appendix covers the following topics:

- Worksheet

Worksheet

Use the following questions to determine the SDK requirements of a customer site.

1. Does Oracle Advanced Inbound Development certify or support this PBX-middleware combination, or is the certification in progress?

   Yes

   No

2. If this PBX-middleware combination is not certified or supported by Oracle Advanced Inbound Development, is there an Oracle consultant or Oracle SSI-provided telephony adapter solution available, or is a solution in progress?

   To check if any solutions may be available, refer to the Oracle Solution ServicesePortal (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).

   Yes

   No

3. Is a consulting-based pipeline available for this PBX and CTI middleware?

   Consider contacting the Oracle professional community (such as the itccon_us mailing list) to ask whether there are opportunities similar to yours.

   Yes

   No

4. PBX product research and data collection

   1. Model:
   2. Current release:
   3. CTI interface:
   4. Route points, how routing works:
   5. Is a software CTI translator required for the proprietary PBX CTI interface?

      Yes

No

6. Does the PBX or CTI middleware permit routing to be controlled by an adjunct platform, specifically, Oracle Advanced Inbound?

   Yes

   No

7. The ACD/PBX is:

   TDM-based

   IP-based

8. Does this PBX provide ACD (automatic call distribution) functionality?

   Yes

   No

9. The platform is:

   CSTA CTI specification (the PBX is a CSTA switch)

   Propriety call model

10. Is a software CTI translator required for the proprietary PBX CTI interface?

    Yes

    No

5. CTI Middleware

   1. Is the manufacturer a leading brand of CTI middleware, and does the manufacturer still exist?

      Yes

      No

   2. The CTI middleware is:

      Intel CT Connect/NetMerge Call Processing Software

      Cisco ICM

      Genesys

      Other

   3. Does this CTI middleware support the customer's PBX platform?

      Yes

      No

   4. List available documentation:

   5. List available tools and simulators:

   6. Are developer licenses for API required?

      Yes

      No

   7. Are developer licenses for API available?

    Yes

    No

   8. Does the CTI middleware provide routing?

      Yes

      No

   9. Does the CTI middleware provide IVR integration?

      Yes

      No

   10. Does the CTI middleware support server-level integration?

       Yes, C-based API

       Yes, Java-based API

       No

6. Additional Information on Advanced Inbound Requirements:

   Log in, log out

   Ready, not ready

   Make call

   Answer a call

   Transfer a call:

   - Consultative transfer to route point
   - Blind transfer to agent
   - Blind transfer to route point

   Conference calls:

   - Consultative conference
   - Blind transfer to agent
   - Blind transfer to route point

   Hold

   Call and data transfer

7. What are the reporting requirements?

   PBX reports

   CTI middleware reports

   Interaction Center Intelligence reports

8. Is Advanced Outbound Predictive required?

   No

   Yes

   1. Does the PBX support a lineside T1 interface to the VDUs?

Yes

No

2. Does the CTI middleware support monitoring and control of the lineside analog PBX extensions?

   Yes

   No

B Telephony Adapter Test Cases

This appendix covers the following topics:

- Introduction

Introduction

Events, denoted by [<Agent>/<EventName>] are Oracle Telephony Manager events to check at the given time.

> **Note:** Not all possible Oracle Telephony Manager events are checked at all times.

The following table lists agent state test procedures.

Number	Test Procedure
A1	With agent A's actual phone logged out, log in agent A with the softphone.
A2	With agent A's actual phone logged in and not ready, log in agent A with the softphone; see if the phone is automatically logged out and then logged in.
A3	With agent A's actual phone logged in and ready, login agent A with the softphone; see if the phone is automatically logged out and then logged in.
A4	With agent A's actual phone logged in and on a call, log in agent A with the softphone; see if the proper error message is displayed; manually hang up any remaining calls on the phone; log in agent A with the softphone again; see if the phone is automatically logged out and then logged in.
A5	Log out agent A.
A6	Set agent A to ready state.
A7	Set agent A to not-ready state.

The following table lists make call/ answer call test procedures.

Number	Test Procedure
B1	A calls B; B does not answer; A hangs up.
B2	A calls B; B answers [NO B/ExternalWithData]; A hangs up.
B3	A calls B; B answers; B hangs up.
B4	A makes outbound call to X; X does not answer; A hangs up.
B5	A makes outbound call to X; X answers [NO A/ExternalWithData]; A hangs up.
B6	A makes outbound call to X; X answers; X hangs up.
B7	A receives inbound call from X [A/ ExternalWithData]; A does not answer; X hangs up.
B8	A receives inbound call from X [A/ ExternalWithData]; A answers; A hangs up.
B9	A receives inbound call from X; A answers; X hangs up.
B10	A calls B which is busy.
B11	A makes outbound call to X which is busy.
B12	A calls an invalid internal number.
B13	A makes outbound call to an invalid external number.

The following table lists hold/retrieve test procedures.

Number	Test Procedure
C1	A calls B; B answers; A puts call on hold; A retrieves the call.
C2	A calls B; B answers; B puts call on hold; B retrieves the call.
C3	A calls B; B answers; A puts call on hold; B puts call on hold; A retrieves the call; B retrieves the call.
C4	A calls B; B answers; A puts call on hold; B hangs up.
C5	A calls B; B answers; B puts call on hold; A hangs up.
C6	A makes outbound call to X; X answers; A puts call on hold; A retrieves the call.
C7	A makes outbound call to X; X answers; A puts call on hold; X hangs up.
C8	A receives inbound call from X [A/ExternalWithData]; A answers; A puts call on hold; A retrieves the call.
C9	A receives inbound call from X; A answers; A puts call on hold; X hangs up.
C10	A calls B; A puts call on hold; B answers; A retrieves the call.
C11	A makes outbound call to X; A puts call on hold; X answers; A retrieves the call.

The following table lists consultative transfer test procedures.

Number	Test Procedure
D1	A receives inbound call from X [A/ExternalWithData]; A answers; A makes consultation call to B; B answers [B/TransferWithData; call data should be same as A/ExternalWithData previously]; A completes transfer.
D2	A receives inbound call from X; A answers; A makes outbound consultation call to Y; Y answers; A completes transfer.
D3	A makes outbound call to X; X answers [NO A/ExternalWithData]; A makes consultation call to B; B answers [NO B/TransferWithData]; A completes transfer.
D4	A makes outbound call to X; X answers; A makes outbound consultation call to Y; Y answers; A completes transfer.
D5	A calls B; B answers; B makes consultation call to C; C answers [NO C/TransferWithData]; B completes transfer.
D6	A calls B; B answers; B makes outbound consultation call to X; X answers; B completes transfer.
D7	A receives inbound call from X; A answers; A makes consultation call to B; B answers; B puts consultation call on hold; A completes transfer; B retrieves transferred call.
D8	A makes outbound call to X; X answers; A makes consultation call to B; B answers; B puts consultation call on hold; A completes transfer; B retrieves transferred call.
D9	A calls B; B answers; B makes consultation call to C; C answers; C puts consultation call on hold; B completes transfer; C retrieves transferred call.
D10	A calls B; B answers; A puts call on hold; B makes consultation call to C; C answers [NO C/TransferWithData]; B completes transfer; A retrieves transferred call.
D11	A calls B; B answers; A puts call on hold; B makes outbound consultation call to X; X answers; B completes transfer; A retrieves transferred call.

The following table lists blind transfer test procedures.

Number	Test Procedure
E1	A receives inbound call from X [A/ ExternalWithData]; A answers; A makes consultation call to B; A completes transfer before B answers; B answers [B/TransferWithData].
E2	A receives inbound call from X; A answers; A makes outbound consultation call to Y; A completes transfer before Y answers; Y answers.
E3	A makes outbound call to X; X answers [NO A/ExternalWithData]; A makes consultation call to B; A completes transfer before B answers; B answers [NO B/TransferWithData].
E4	A makes outbound call to X; X answers; A makes outbound consultation call to Y; A completes transfer before Y answers; Y answers.
E5	A calls B; B answers; B makes consultation call to C;B completes transfer before C answers; C answers [NOC/TransferWithData].
E6	A calls B; B answers; B makes outbound consultation call to X; B completes transfer before X answers; X answers.
E7	A calls B; B answers; A puts call on hold; B makes consultation call to C; B completes transfer before C answers; C answers [NO C/TransferWithData]; A retrieves transferred call.
E8	A calls B; B answers; A puts call on hold; B makes outbound consultation call to X; B completes transfer before X answers; X answers; A retrieves transferred call.

The following table lists consultative conference test procedures.

Number	Test Procedure
F1	A receives inbound call from X [A/ ExternalWithData]; A answers; A makes consultation call to B; B answers [B/ ConferenceWithData; call data should be same as A/ExternalWithData previously]; A completes conference.
F2	A receives inbound call from X; A answers; A makes outbound consultation call to Y; Y answers; A completes conference.
F3	A makes outbound call to X; X answers [NO A/ ExternalWithData]; A makes consultation call to B; B answers [NO B/ConferenceWithData]; A completes conference.
F4	A makes outbound call to X; X answers; A makes outbound consultation call to Y; Y answers; A completes conference.
F5	A calls B; B answers; B makes consultation call to C; C answers [NO C/ConferenceWithData]; B completes conference.
F6	A calls B; B answers; B makes outbound consultation call to X; X answers; B completes conference.
F7	A receives inbound call from X; A answers; A makes consultation call to B; B answers; B puts consultation call on hold; A completes conference; B retrieves conferenced call.
F8	A makes outbound call to X; X answers; A makes consultation call to B; B answers; B puts consultation call on hold; A completes conference; B retrieves conferenced call.
F9	A calls B; B answers; B makes consultation call to C; C answers; C puts consultation call on hold; B completes conference; C retrieves conferenced call.
F10	A calls B; B answers; A puts call on hold; B makes consultation call to C; C answers [NO C/ConferenceWithData]; B completes conference; A retrieves conferenced call.
F11	A calls B; B answers; A puts call on hold; B makes outbound consultation call to X; X answers; B completes conference; A retrieves conferenced call.

The following table lists consultation (transfer/conference) cancel test procedures.

Number	Test Procedure
G1	A receives inbound call from X [A/ ExternalWithData]; A answers; A makes consultation call to B; X hangs up before B answers; B answers [B/TransferWithData].
G2	A receives inbound call from X; A answers; A makes consultation call to B; A hangs up consultation call before B answers; A reconnects to X.
G3	A receives inbound call from X [A/ ExternalWithData]; A answers; A makes consultation call to B; B answers [B/ TransferWithData]; X hangs up before A completes the transfer/conference.
G4	A receives inbound call from X; A answers; A makes consultation call to B; B answers; A hangs up consultation call; A reconnects to X.
G5	A receives inbound call from X; A answers; A makes consultation call to B; B answers; B hangs up consultation call; A reconnects to X.
G6	A receives inbound call from X; A answers; A makes outbound consultation call to Y; X hangs up before Y answers; Y answers.
G7	A receives inbound call from X; A answers; A makes outbound consultation call to Y; A hangs up consultation call before Y answers; A reconnects to X.
G8	A receives inbound call from X; A answers; A makes outbound consultation call to Y; Y answers; X hangs up before A completes the transfer/conference.
G9	A receives inbound call from X; A answers; A makes outbound consultation call to Y; Y answers; A hangs up consultation call; A reconnects to X. P
G10	A receives inbound call from X; A answers; A makes outbound consultation call to Y; Y answers; Y hangs up consultation call; A reconnects to X.
G11	A makes outbound call to X; X answers; A makes consultation call to B; X hangs up before B answers; B answers.
G12	A makes outbound call to X; X answers; A makes consultation call to B; A hangs up consultation call before B answers; A reconnects to X.

Number	Test Procedure
G13	A makes outbound call to X; X answers; A makes consultation call to B; B answers; X hangs up before A completes the transfer/conference.
G14	A makes outbound call to X; X answers; A makes consultation call to B; B answers; A hangs up consultation call; A reconnects to X.
G15	A makes outbound call to X; X answers; A makes consultation call to B; B answers; B hangs up consultation call; A reconnects to X.
G16	A makes outbound call to X; X answers; A makes outbound consultation call to Y; X hangs up before Y answers; Y answers.
G17	A makes outbound call to X; X answers; A makes outbound consultation call to Y; A hangs up consultation call before Y answers; A reconnects to X.
G18	A makes outbound call to X; X answers; A makes outbound consultation call to Y; Y answers; X hangs up before A completes the transfer/conference.
G19	A makes outbound call to X; X answers; A makes outbound consultation call to Y; Y answers; A hangs up consultation call; A reconnects to X.
G20	A makes outbound call to X; X answers; A makes outbound consultation call to Y; Y answers; Y hangs up consultation call; A reconnects to X.
G21	A calls B; B answers; B makes consultation call to C; A hangs up before C answers; C answers.
G22	A calls B; B answers; B makes consultation call to C; B hangs up consultation call before C answers; B reconnects to A.
G23	A calls B; B answers; B makes consultation call to C; C answers; A hangs up before B completes the transfer/conference.
G24	A calls B; B answers; B makes consultation call to C; C answers; B hangs up consultation call; B reconnects to A.
G25	A calls B; B answers; B makes consultation call to C; C answers; C hangs up consultation call; B reconnects to A.

Number	Test Procedure
G26	A calls B; B answers; B makes outbound consultation call to X; A hangs up before X answers; X answers.
G27	A calls B; B answers; B makes outbound consultation call to X; B hangs up consultation call before X answers; B reconnects to A.
G28	A calls B; B answers; B makes outbound consultation call to X; X answers; A hangs up before B completes the transfer/conference.
G29	A calls B; B answers; B makes outbound consultation call to X; X answers; B hangs up consultation call; B reconnects to A.
G30	A calls B; B answers; B makes outbound consultation call to X; X answers; X hangs up consultation call; B reconnects to A.
G31	A receives inbound call from X; A answers; A makes consultation call to B which is busy; A reconnects to X.
G32	A receives inbound call from X; A answers; A makes outbound consultation call to Y which is busy; A reconnects to X.
G33	A receives inbound call from X; A answers; A makes consultation call to an invalid internal number; A reconnects to X.
G34	A receives inbound call from X; A answers; A makes outbound consultation call to an invalid external number; A reconnects to X.
G35	A makes outbound call to X; X answers; A makes consultation call to B which is busy; A reconnects to X.
G36	A makes outbound call to X; X answers; A makes outbound consultation call to Y which is busy; A reconnects to X.
G37	A makes outbound call to X; X answers; A makes consultation call to an invalid internal number; A reconnects to X.
G38	A makes outbound call to X; X answers; A makes outbound consultation call to an invalid external number; A reconnects to X.
G39	A calls B; B answers; B makes consultation call to C which is busy; B reconnects to A.

Number	Test Procedure
G40	A calls B; B answers; B makes outbound consultation call to X which is busy; B reconnects to A.
G41	A calls B; B answers; B makes consultation call to an invalid internal number; B reconnects to A.
G42	A calls B; B answers; B makes outbound consultation call to an invalid external number; B reconnects to A.

The following table lists route point test procedures.

Number	Test Procedure
K1	A calls R; R routes to B [NO B/ ExternalWithData]; B answers; A makes consultation call to C; C answers [NO C/TransferWithData]; A completes transfer.
K2	A calls R; R routes to B; B answers; A makes consultation call to C; A completes transfer before C answers; C answers [NO C/TransferWithData].
K3	A calls R; R routes to B; B answers; A makes consultation call to C; C answers [NO C/ConferenceWithData]; A completes conference.
K4	A calls B; B answers; B makes consultation call to R; R routes to C; C answers [NO C/TransferWithData]; B completes transfer.
K5	A calls B; B answers; B makes consultation call to R; B completes transfer before R routes to C; R routes to C; C answers [NO C/TransferWithData].
K6	A calls B; B answers; B makes consultation call to R; R routes to C; B completes transfer before C answers; C answers [NO C/TransferWithData].
K7	A calls B; B answers; A makes consultation call to R; R routes to C; C answers [NO C/TransferWithData]; A completes transfer.
K8	A calls B; B answers; A makes consultation call to R; A completes transfer before R routes to C; R routes to C; C answers [NO C/TransferWithData].

Number	Test Procedure
K9	A calls B; B answers; A makes consultation call to R; R routes to C; A completes transfer before C answers; C answers [NO C/TransferWithData].
K10	A calls B; B answers; B makes consultation call to R; R routes to C; C answers [NO C/ConferenceWithData]; B completes conference.
K11	A calls R; R routes to B; B answers; A makes consultation call to C; B hangs up before C answers; C answers [NO C/ConferenceWithData].
K12	A calls R; R routes to B; B answers; A makes consultation call to C; A hangs up consultation call before C answers.
K13	A calls R; R routes to B; B answers; A makes consultation call to C; C answers; B hangs up.
K14	A calls R; R routes to B; B answers; A makes consultation call to C; C answers; A hangs up consultation call.
K15	A calls R; R routes to B; B answers; A makes consultation call to C; C answers; C hangs up consultation call.
K16	A calls B; B answers; B makes consultation call to R; A hangs up before R routes to C; R routes to C; C answers.
K17	A calls B; B answers; B makes consultation call to R; R routes to C; A hangs up before C answers; C answers.
K18	A calls B; B answers; B makes consultation call to R; B hangs up consultation call before R routes to C.
K19	A calls B; B answers; B makes consultation call to R; R routes to C; B hangs up consultation call before C answers.
K20	A calls B; B answers; B makes consultation call to R; R routes to C; C answers; A hangs up.
K21	A calls B; B answers; B makes consultation call to R; R routes to C; C answers; B hangs up consultation call.
K22	A calls B; B answers; B makes consultation call to R; R routes to C; C answers; C hangs up consultation call.

Number	Test Procedure
K23	A receives inbound call from X [A/ExternalWithData]; A answers; A makes consultation call to R; R routes to B; B answers [B/TransferWithData]; A completes transfer.
K24	A receives inbound call from X; A answers; A makes consultation call to R; A completes transfer before R routes to B; R routes to B; B answers [B/TransferWithData].
K25	A receives inbound call from X; A answers; A makes consultation call to R; R routes to B; A completes transfer before B answers; B answers [B/TransferWithData].
K26	A receives inbound call from X; A answers; A makes consultation call to R; R routes to B; B answers [B/ConferenceWithData]; A completes conference.
K27	A makes an outbound call to X; X answers; A makes consultation call to R; R routes to B; B answers [B/TransferWithData]; A completes transfer.
K28	A makes an outbound call to X; X answers; A makes consultation call to R; A completes transfer before R routes to B; R routes to B; B answers [B/TransferWithData].
K29	A makes an outbound call to X; X answers; A makes consultation call to R; R routes to B; A completes transfer before B answers; B answers [B/TransferWithData].
K30	A makes an outbound call to X; X answers; A makes consultation call to R; R routes to B; B answers [B/ConferenceWithData]; A completes conference.

The following table lists DTMF tones test procedures.

Number	Test Procedure
L1	A received inbound call from X; A answers; A presses some digits and clicks on "Dial" button; X hears DTMF tones of digits dialed by A.
L2	A makes outbound call to X; X answers; A presses some digits and clicks on "Dial" button; X hears DTMF tones of digits dialed by A.
L3	A calls B; B answers; A presses some digits and clicks on "Dial" button; B hears DTMF tones of digits dialed by A.
L4	A calls B; B answers; B presses some digits and clicks on "Dial" button; A hears DTMF tones of digits dialed by B.

C
Sample Code

This appendix covers the following topics:

- Sample Codes for Integration Levels and Extension
- Switch Simulator Overview
- Sample Adapter Design for Standard and Advanced Integration
- Sample Adapter Design for Basic Integration

Sample Codes for Integration Levels and Extension

The following tables describe the locations of the sample codes for Standard Integration, Advanced Integration and Advanced Outbound Extension, and the functions supported by the corresponding sample implementations.

Switch Simulator Overview

The switch simulator is one of the Interaction Center server processes that is used in demonstration environments where a real switch is not available. The switch simulator supports teleset and route point functions, and can be accessed programmatically using the OpenTel Bean API (a Java API, package oracle.apps.cct.openTel.bean). The switch simulator models each line of the teleset and route point as extension objects. Extension objects can be accessed using the OpenTelExtensionBean. Events are sent by the switch simulator as OpenTel events.

Sample Adapter Design for Standard and Advanced Integration

The Standard Integration and Advanced Integration sample adapter connects to the switch simulator and forwards Adapter SDK method calls to the OpenTel Bean API method calls and also process OpenTel events. The sample adapter then translates the OpenTel events into Adapter events.

The following example demonstrates the relationship among the Oracle Interaction Center servers.

Interaction Center <---> Oracle Telephony Adapter Server <--> SampleAdapter <---> Switch Simulator.

SampleTeleDeviceFactory

SampleTeleDeviceFactory reads the IP address and port from the Hashtable in its init() method.

SampleTelesetDevice

SampleTelesetDevice maintains one OpenTelExtensionBean object for each line. Adapter SDK methods are directed to the appropriate OpenTelExtensionBean object based on the line index specified. SampleTelesetDevice registers a dedicated OpenTelEventListener for each OpenTelExtensionBean object to process switch simulator events.

SampleRoutePointDevice

SampleRoutePointDevice maintains one OpenTelExtensionBean object. Adapter SDK methods are directed to the route point OpenTelExtensionBean object. SampleRoutePointDevice registers itself as OpenTelEventListener to process switch simulator events.

Sample Adapter Design for Basic Integration

For Basic Integration sample adapter design, see Sample Code.

D

SDK Qualification Worksheet

This appendix covers the following topics:

- What ACD/PBXs are supported?
- What, if any, CTI middlewares are supported?
- Does the customer already have CTI middleware or a preference for a particular CTI middleware?
- How many agents does the interaction center have?
- How many sites does the customer want to CTI enable?
- (Only if # 5 is yes) If the customer wants to enable multiple sites, they want:
- Is the customer an existing Oracle eBusiness Suite customer?
- In which Oracle Advanced Inbound features is the customer interested?
- Does the interaction center have automated outbound dialing? (outbound telesales campaigns)
- 10. Does the interaction center have an IVR?
- What is the $US total license revenue?
- What is the implementation project time line?
- What is the consulting budget?

What ACD/PBXs are supported?

Use the following questions to determine if the integration of Oracle Telephony Adapter SDK at a specific customer site is necessary and, if it is necessary, what the SDK requirements are.

- Name, version and release (current or legacy? If old, plan to upgrade?)
- Is there a CTI interface?
  - Yes
  - No
- What ACD teleset is used?
- How many extensions/line appearances are on each agent teleset?
- On which line are inbound calls received?

- On which line are outbound calls made?

What, if any, CTI middlewares are supported?

- Name, version and release
- Support for server-side integration:
  - C API
  - Java API

Does the customer already have CTI middleware or a preference for a particular CTI middleware?

- No
- Yes, the ACD/PBX is supported by the following CTI middleware:
  - CT Connect (Java or C API)
  - Cisco ICM, (C API only)
  - Genesys, (C API only)

How many agents does the interaction center have?

- 1000 agents per Oracle Telephony Adapter Server/Call Center Connectors on a single site
- Fewer than 1000 agents

How many sites does the customer want to CTI enable?

- Oracle Advanced Inbound Oracle Telephony Adapter SDK supports single site routing, queueing and distribution in this release.
- If the customer has multiple sites, multi-site routing and call and data transfer, the customer may want to consider a CTI middleware which provides multi-site functionality, such as Cisco ICM or Genesys.

(Only if # 5 is yes) If the customer wants to enable multiple sites, they want:

- Enterprise routing (arrive at any site and be distributed to any site)
- Enterprise call and data transfer

Is the customer an existing Oracle eBusiness Suite customer?

- Yes, they have the following applications:
  - TeleService
  - TeleSales
  - iSupport

    - iMeeting
    - Collections
    - iStore
    - Marketing Online
- No, they are buying the following applications:
    - TeleService
    - TeleSales
    - iSupport
    - iMeeting
    - Collections
    - iStore
    - Marketing Online

In which Oracle Advanced Inbound features is the customer interested?

- Call routing:
    - PBX skill-based
    - DNIS
    - Routing on business data
    - Rules-based routing
- Screen pops
- Other call data (If IVR, see #9)
- Call and data transfer
- They have or want a softphone GUI

Does the interaction center have automated outbound dialing? (outbound telesales campaigns)

- Yes. How do they want to dial?
    - Preview: Record delivered. An agent initiates an outbound call.
    - Progressive: Record delivered to an agent. Dialing is automatic.
    - Predictive: System dials customer. Only live contacts are transferred to the agent.
- No.

10. Does the interaction center have an IVR?

- Yes.
    - The IVR manufacturer is
    - CTI-enabled/intelligent (IVR maintains the call ID of the call exiting the IVR)

  - IVR connects (PL/SQL) to an Oracle database IVR updates the call's application data field
- No.

What is the $US total license revenue?

- eBusiness suite
- Interaction Center

What is the implementation project time line?

What is the consulting budget?

Glossary

active mode

A routing mode in which Oracle Advanced Inbound controls the routing and distribution of incoming calls to call center agents using business data and rules that are configured in Oracle Advanced Inbound. Specific ACD/PBX configurations are required to grant Oracle Advanced Inbound full control of an inbound call when it reaches a ACD/PBX route point monitored by Oracle Advanced Inbound.

adapter

A telephony driver of the Oracle Telephony Adapter Server developed specifically to integrate Oracle Interaction Center to a specific switch and CTI middleware platform. Oracle develops adapters for certified switch and middleware combinations. Third-parties can use the Oracle Telephony Adapter SDK to develop adapters for switch and middleware combinations that are not certified by Oracle. Typically, each adapter is developed to integrate only with the telephony system of a specific manufacturer.

ACD

Automatic Call Distribution, systems designed to automatically answer, queue and route incoming calls to interaction center agents. An ACD differs from a PBX in that while a PBX allows users to share a limited number of telephone lines, an ACD has at least one telephone line for each agent.

ANI

Automatic Number Identification, a service, similar to caller ID, that long distance carriers provides to identify the calling party's billing number.

API

Application Programming Interface, the calling conventions by which a software application accesses the operating system and other services.

blind transfer

A call transferred from one person to another without the first person telling the other the identity of the caller.

canonical phone number

A standardized phone number of the format:

+<country code> (<area code>) <local exchange>-<subscriber number>

For example, +1 (123) 456-7890.

Given a canonical phone number and an interaction center's local telephone network characteristics, such as local country and area codes, Oracle Advanced Inbound can automatically determine how to call a number.

CTI

Computer Telephony Integration, a system in which a computer is connected to a telephone switch, either PBX or ACD, so that the computer sends instructions to the switch about how to direct telephone calls.

DN

Directory Number, typically the phone number associated with a telephone line.

DNIS

Dialed Number Identification Service, a feature of 800 and 900 lines that identifies the called number to a telephony system, which routes the call to the correct extension.

DTMF

Dual Tone Multi-Frequency, also known as touch-tone dialing.

dynamic route

A route that is based on a PL/SQL query.

enhanced passive mode

A routing mode in which standard ACD/PBX routing and distribution of calls to call center agents occurs with Oracle Advanced Inbound monitoring ACD/PBX route points to allow classification of calls for targeted screen pops, inbound call queue counts and tracking of calls that are abandoned at the route point for reporting by Oracle Interaction Center Intelligence. Specific ACD/PBX configurations are required to ensure that inbound calls pass through a ACD/PBX route point that is monitored by Oracle Advanced Inbound.

IDE

Interactive Development Environment, a system for supporting the process of writing software. An IDE may include a syntax-directed editor, graphical tools for program entry, and integrated support for compiling and running the program and relating compilation errors back to the source. Examples of IDEs are Visual C++ and Visual Basic.

Interaction Center Server Manager

The only server process that is required to be explicitly started on each target machine, ICSM is responsible for starting, stopping and monitoring all the other Oracle Advanced Inbound server processes. The ICSM server processes are controlled by the Interaction Center Server HTML Administration.

Inbound Telephony Server

The Oracle Interaction Center server that handles inbound telephony interactions. ITS supports the following features:

- (Active mode only) ITS enables enterprise data-based routing by listening for route queries offered by the CTI middleware and responding to them to instruct the switch where to route the call.
- ITS monitors calls arriving at route point(s)
- ITS detects calls that are abandoned at route point(s)
- ITS receives IVR data packets from the IVR

interaction center server

Any interaction center server, such as Oracle Interaction Queuing and Distribution, Oracle Universal Work Queue, Oracle Routing server and Oracle Inbound Telephony Server. Same as mid-tier server process and server process.

IVR

Interactive Voice Response, an automated system that, in response to incoming telephone calls, plays a recorded message that gives callers the option of pressing telephone buttons to route the call to one or more extensions.

Javadoc

A facility provided within the Java Development Kit that produces HTML documentation from a program. Javadoc reads the source code and parses specially formatted and positioned comments into documentation.

JDBC

Java Database Connectivity, part of the Java Development Kit that defines an application programming interface for Java for standard SQL access from Java programs to databases.

Java Development Kit (JDK)

A Sun Microsystems product that provides the required environment for Java programming.

Java Native Interface (JNI)

A native programming interface for Java that allows Java code that is running inside a Java Virtual Machine to operate with applications and libraries written in other programming languages such as C and C++.

Java Virtual Machine (JVM)

A specification for software which interprets Java programs that have been compiled into byte-codes and usually stored in a ".class" file. JVMs support object-oriented programming directly by including instructions for object method invocation. JVMs are written in C and can be ported to run on most platforms.

media controller

Software that bridges other systems or software with the underlying media hardware, such as a PBX.

media queue

The interaction center component for queuing and distributing inbound media items. It stores inbound items such as telephone calls or e-mails in a queue and integrates with the routing module so that the items can be sent to a set of agents. The media queue provides an API to other modules, such as Oracle Universal Work Queue, for querying and manipulating items in the queue.

media item

A representation of a telephone call, e-mail, Web callback or other type of media.

mid-tier server process

Any interaction center server, such as Oracle Interaction Queuing and Distribution, Oracle Universal Work Queue, Oracle Routing Server, Oracle Inbound Telephony Server, and Oracle Telephony Media Controller. Same as server process and interaction center server.

monitoring

The ability to view server status.

multi-site

Interaction centers that work together across multiple physical locations.

multi-site routing

The ability to route a call to agents who are located across multiple sites.

multi-site queuing and distribution

A single system storing and maintaining agent queues across multiple sites.

multi-PBX

Support for multiple switches and middleware configurations by the same server.

Observer API

The API that the adapter uses to notify Oracle Interaction Center about incoming calls and to deliver call and caller data that are used for softphone displays, screen pops and call routing. Oracle Interaction Center implements the Observer API.

Oracle Advanced Inbound

The Oracle eBusiness application that is required to telephony enable business applications in the Oracle eBusiness suite. The server architecture of Oracle Advanced Inbound is scalable to run interaction centers with a single physical site or multiple sites. The Oracle Advanced Inbound bundle consists of the following products: Oracle Universal Work Queue, Oracle Telephony Manager, Oracle Telephony Adapter Server and Oracle Interaction Blending.

Oracle Advanced Outbound

The Oracle eBusiness application that provides the outbound telephony capability corresponding to Oracle Advanced Inbound.

Oracle Interaction Center

A group of server processes that serves as the telephony-enabling foundation of Oracle's eBusiness Suite applications.

Oracle Telephony Adapter Server

The CTI adapter server that substitutes for Oracle Call Center Connectors. Oracle Telephony Adapter Server encompasses one telephony adapter per switch.'

Oracle Telephony Manager

The Oracle Interaction Center application that performs queuing, routing and distribution of media items.

package

Groups of procedures, functions, variables and SQL statements grouped together into a single unit.

passive mode

A routing mode in which standard ACD/PBX routing and distribution of calls to call center agents occurs. Oracle Advanced Inbound becomes aware of the call through CTI when the call rings at the agent’s teleset. Oracle Advanced Inbound does not monitor or control any ACD/PBX route points in this mode.

PBX

Private Branch eXchange, a telephone system within a company or other organization that switches calls between the company’s users and allows them to share a number of outside telephone lines. In passive mode calls are routed by the PBX routes calls.

Provider API

The API that supports telephony and interaction center feature requests such as make call, transfer call, route call and change agent mode. The Provider API Provider is implemented by the adapter.

route point

The first point from which calls are queued and routed. An automatic call distributor (ACD). Route points are also called "pilots" by Alcatel.

scalability

A measure of how well a software or hardware product is able to adapt to future business needs.

screen pop

A user interface presentation of customer data and product and service information that appears on an interaction center agent’s monitor simultaneously with the customer’s incoming telephone call.

server process

Any interaction center server, such as Oracle Interaction Queuing and Distribution, Oracle Universal Work Queue, Routing server, Oracle Inbound Telephony Server, and Oracle Telephony Media Controller. Same as mid-tier server process and interaction center server.

server status

Information on whether the server process is running or not, how long the server has been running, and so on.

site

A single geographic location where an interaction center is located. A site typically has a PBX and CTI middleware installed.

skill-based routing

A dynamic call routing intelligence that delivers inbound calls to an agent who is appropriately skilled to meet the needs of the caller.

softphone

A functional GUI representation of a telephone that is displayed on interaction agents' monitors.

Software Development Kit

(SDK) Software that is provided by software vendors to allow their products to be used with the products of other software vendors.

static route

A route that is based on cached data.

super group

The topmost, parent server group in a hierarchy of server groups.

switch simulator

A process that uses Intel CT Connect middleware to simulate a Nortel switch and the connection and message behavior of the Oracle Call Center Connectors server. The switch simulator makes it possible to set up an interaction center without connecting to a real switch.

target machine

The machine where mid-tier server processes are run. Same as node.

telephony enabled

The ability of an application to communicate with a telephone system for inbound and outbound calls, or inbound or outbound calls, through the CTI middleware that handles the messaging between a telephone switch and the user's application.

telephony model

A scenario that describes the expected behavior of a call for any given telephony function. For example, in one telephony model, a transferred call has the same call ID as the original call. In another telephony model, a transferred call has the same call ID as the consultation call. In a third telephony model, a transferred call has a completely new call ID that differs from the original call and the consultation call.

telephony system

Any hardware and software components that provide telephony and CTI messaging, such as PBX, ACD, IVR, predictive dialer and CTI middleware.

VDU

Voice detection unit.

Index